N°= d'ordre : 07/2018-D/INF

REBUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique

THESE DE DOCTORAT

Présentée pour l'obtention du grade de **DOCTEUR**

En : INFORMATIQUE

Spécialité : Systèmes Informatiques

Par : KHEROUA LEILA

Sujet

Dissémination à base d'Agents Mobiles dans les Réseaux de Capteurs.

Soutenue publiquement, le 05 Décembre 2018, devant le jury composé de :

Mr. Y. AKLOUF	Professeur, à l'USTHB	Président
Mme. S. MOUSSAOUI	Professeur, à l'USTHB	Directrice de thèse
Mr. S.M. SENOUCI	Professeur, à l'U. Bourgogne	Examinateur
Mme. N. NOUALI	Directrice de Recherche, au CERIST	Examinatrice
Mr. S. BITAM	Maitre de Conférences /A, à l'U. Biskra	Examinateur
Mr. Y. ZAFFOUNE	Maitre de Conférences /A, à l'USTHB	Examinateur
Mr. M.GUERROUMI	Maitre de Conférences /A, à l'USTHB	Invité

Résumé

Le travail réalisé dans cette thèse s'inscrit dans le cadre de la recherche utilisant des agents de rumeur pour disséminer l'information dans les réseaux de capteurs sans fil (RCSF) ne disposant pas de systèmes de localisation géographique. L'utilisation des agents de rumeur dans les RCSF permet deux gains non négligeables : la réduction du trafic généré grâce à leur collaboration ainsi que l'optimisation de la consommation de la bande passante.

En l'absence de l'information de localisation, le parcoure de l'agent est aléatoire (a random walk path) et risque d'être concentré sur une partie restreinte du réseau créant ainsi des chemins avec boucles (spiral like routing paths). Dans cette thèse, nous proposons une nouvelle approche de construction de chemin en exploitant une information de voisinage à deux sauts qui permet à l'agent de maintenir une trajectoire aussi droite que possible. Le premier protocole proposé, FRA (Fast Rumor Agent), se base sur un choix sélectif du prochain saut de l'agent. Il exploite une information de voisinage chargée dans des structures de données réinitialisées évitant ainsi de surcharger l'agent lors de son parcoure. Pour une meilleure dissémination de l'information, nous avons proposé par la suite le protocole EDARD (Efficient Data Access based on Rumor Dissemination) qui implémente une nouvelle procédure de création d'agents fils (The forking procedure) afin d'uniformiser la distribution de l'information et d'améliorer le taux de délivrance des requêtes émises par les nœuds puits.

Pour les applications critiques (sauvetage, repérage en catastrophe) dans des environnements hostiles, nous avons proposé deux autres protocoles : CSR (Corridor Star Routing) et BWR (Backbone Web Routing) dont l'objectif est d'assurer un accès rapide et efficace à la donnée. Les résultats enregistrés montrent que ces protocoles permettent un gain appréciable en comparaison à d'autres approches de la littérature en termes de : taux de délivrances de requêtes, temps d'accès à la donnée ainsi que le trafic généré dans le réseau.

Mots clés : Dissémination, Agents de rumeurs, Routage, Rumor Routing (RR), Energie, Requête (query), Evènement (event), et Réseau de capteurs sans fil (RCSF).

Abstract

In the context of Wireless Sensor Networks (WSN) where position information is not assumed, we focus our research on improving data dissemination using rumor agents. The use of rumor agents allows in one hand to reduce traffic generated by agents' collaboration and on the other hand the optimization of bandwidth consumption.

In environments without position information, agents progress in random walk paths concentrated in one part of the network creating spiral like routing paths. This thesis proposes a new path construction approach based on two hops nodes neighboring information, which allows agent to maintain straightest path. The first protocol, Fast Rumor Agents (FRA) uses a neighboring information embedded in frequently reinitialized data structures to create straight paths. Later , and in order to optimize and homogenize data dissemination, we propose the Efficient Data Access based on Rumor Dissemination protocol (EDARD) which adapts a new forking procedure to uniform data distribution and improve query delivery rate.

For critical applications like saving, catastrophic detection in hostile environments, we propose two protocols: Corridor Star Routing (CSR) and Backbone Web Routing (BWR) that aim at a quick and efficient data access. Compared to other relative, the results show a considerable benefits in terms of query delivery rate, data access time and generated traffic in the network.

Key words: Dissemination, Rumor agents, Routing, Rumor Routing (RR), Energy, Query, Event, and Wireless Sensor Networks (WSN).

To my lovely mother …

Acknowledgments

Over time, I learned how to execute, process and synchronize real-life tasks to achieve this chapter of my life. But, I also realized that this achievement would be impossible without our genius human body resources: a brain to think, a body to act and a heart to still believe… Consequently, I would like to express my sincere gratitude to my supervisor Prof. Moussaoui Samira that have accepted – years ago- to conduct and supervise this work. I also thank her for its remarkable pedagogical skill, continuous guidance and encouragements. My special thanks goes to my colleagues and team researchers members: Dr. Guerroumi Mohammed, Dr. Doukha Zouina and Dr. Al-Sakib Khan Pathan for their useful comments to enhance the quality of this work.

It is my pleasure to thank Prof. Aklouf Youcef that has accepted to chair the evaluation committee of this thesis. I also thank Prof Senouci Sidi Mohamed from university of Bourgogne, Dr. Nouali Nadia from CERIST, Dr. Bitam Salim from university of Biskra and Dr. Zaffoune Youcef from USTHB for their spent time and valuable comments.

I also wish to express my thanks to my parents, my husband, and my siblings for their invaluable support.

Table des matières

Liste des figures

Liste des tableaux

Glossaire

ACK	Acknowledgment
ACO	Ant Colony Optimisation
ADC/DAC	Digital to Analogical Converter/Digital to Analogical Converter
AlAc	Along & Across algorithm
AoA	Angle of Arrival
ARR	Appointment-based Rumor Routing
BWR	Backbone Web Routing
COG	Center Of Gravity
CRR	Clustering Rumor Routing
CSR	Corridor Star Routing
DD	Directed Diffusion
DRR	Directional Rumor Routing
EDARD	Efficient Data Access based on Rumor Dissemination
FA	Forking Agents
FRA	Fast Rumor Agent
GAF	Geographic Adaptive Fidelity
GEAR	Geographic and Energy Aware Routing
GPS	Global Positioning System
LBDD	A Line-Based Data Dissemination protocol for Wireless Sensor Networks
LEACH	Low-Energy Adaptive Clustering Hierarchy
MECN	Minimum Energy Communication Network
MEMS	Micro-Electro-Mechanical Systems
MWE	Multiple Winner Algorithm
PEGASIS	Power- Efficient GAthering in Sensor Information Systems
QoS	Quality of Service
RCRR	Relative Coordinate Rumor Routing
RCSF	Reseau de Capteurs Sans Fil
RER	Real time Energy efficient Routing protocol
RR	Rumor Routing
SLR	Straight Line Routing
SPEED	Spatiotemporal Communication Protocol for Sensor Networks
SPIN	Sensor Protocol for Information via Negociation
SWE	Single Winner Algorithm
TBF	Trajectory-Based Forwarding
TEEN	Threshold sensitive Energy Efficient sensor Network protocol
TTDD	A Two Tire Data Dissemination approach for Wireless Sensor Networks
TTF	Time To Fork
TTL	Time To Live
WSN	Wireless Sensor Network
ZEU	Zone Estimation Unit
ZRR	Zonal Rumor Routing

INTRODUCTION GENERALE

Les avancées récentes dans la technologie des capteurs, des systèmes « MEMS » (Micro-Electro-Mechanical Systems) et des communications sans fil ont participé de manière significative à l'émergence et au développement des Réseaux de Capteurs Sans Fil (RCSF), en anglais : Wireless Sensor Networks (WSNs). Les RCFS élargissent l'accès des populations à l'information, communiquent l'information physique des environnements de recherches avec les réseaux de transmission, permettant ainsi aux réseaux de nouvelles générations de donner à la population un accès direct, effectif et authentique à l'information.

Les capteurs communément appelés nœuds sont des microcomposants qui intègrent des dispositifs d'interception et de communication sans fil dans un seul circuit, à dimension réduite, et avec un coût raisonnable. Un RCSF est donc une collection d'un très grand nombre de ces nœuds capteurs déployés dans de vastes régions pour la collecte et le traitement de la donnée ainsi que pour le monitoring et la détection de certains événements spéciaux. Un RCSF établit des connexions avec le monde extérieur à travers un ou plusieurs nœuds puits (appelés aussi passerelles ou sink en anglais). Toute donnée collectée par les nœuds capteurs est envoyée vers le nœud puits qui la transmet à son tour à des serveurs et des utilisateurs terminaux via Internet, satellite ou un medium viable. [1]

Les nœuds d'un réseau de capteur sont caractérisés par des capacités limitées de stockage, de traitement et d'énergie. Un RCSF est caractérisé par une durée de vie limitée à cause de la contrainte d'énergie. Généralement, un nœud capteur détecte des événements, procède au traitement de la donnée captée et transmet cette dernière au nœud puits. Mais parfois, il peut générer des requêtes pour rechercher des événements, localiser une donnée ou un service particulier à travers le réseau. Initialement, l'utilisation des RCSF était réservée uniquement pour le domaine militaire. De nos jours, leur utilisation est pluridisciplinaires, elle touche différents secteurs industriels, académiques ainsi que la télémédecine et le biomédical. Les RCSF sont principalement déployés dans des environnements hostiles, difficilement accessibles à cause des catastrophes (feu, tremblement de terre,…etc.)[1,2]

Une issue fondamentale des applications dans les RCSF, est de définir comment qu'est qu'un événement (une information) détecté par un ensemble de nœuds soit relayé saut par saut jusqu'au nœud originaire de la requête. Ce problème est en effet un problème de dissémination de données. De ce fait, la conception de protocoles de routage et de dissémination dans les RCSF doit prendre en considération les ressource en : (a) énergie des batteries non rechargeable, (b) capacités de traitement et de stockage limitées. Par conséquent, ces protocoles doivent être simples, avoir de bonnes performances en terme de conservation d'énergie, de mise à l'échelle (scalability) et doivent assurer une longévité du réseau.

Différent protocoles de routage et de dissémination ont été proposé pour les RCSF. Un des premiers paradigmes utilisés est l'inondation (*Flooding)* [12] [13]. Ce paradigme offre la possibilité de construire de plus courts chemins en effectuant plusieurs diffusions des événements et des requêtes sur le réseau. Par contre, il consomme beaucoup de ressources énergétiques et de bande passante. De par les contraintes sévères des RCSF surtout celle de l'énergie, les protocoles de routage utilisant les agents de rumeurs [41-47] semblent être mieux

adaptés et plus intéressants en terme de performances. Au lieu d'avoir recours à la diffusion du premier sort, de simples paquets appelés agents peuvent être utilisé pour disséminer des descriptions sur les données environnementales captées. L'utilisation des agents de rumeur dans les RCSF permet deux gains non négligeables : (a) la réduction du trafic généré grâce à leur collaboration ; qui fait que lorsqu'un agent croise le chemin d'autres agents, il peut disséminer autant d'information que le nombre d'agents sur un seul chemin minimisant ainsi le nombre de transmissions. (b) l'optimisation de la consommation de la bande passante vu que la taille du code transporté par l'agent se résume à quelques lignes de code. Ces principales motivations ont guidées le travail de recherche présenté dans ce document. Notre objectif est donc de concevoir un protocole de dissémination de données à base d'agents pour les applications critiques (non tolerable delay applications). Trois principales métriques parmi d'autres sont utilisées pour mesurer les performances des protocoles proposés à savoir: le trafic généré (comme mesure de l'énergie consommée), le temps de réponse (la latence) et le taux de délivrance de requêtes.

1. Problématique

Un des premiers protocoles de routage et de dissémination de données qui a été largement utilisé dans les réseaux de capteurs est le célèbre : *Directed Diffusion (DD)* [12]. Ce protocole se base sur la technique de diffusion dirigée pour construire des chemins optimaux entre deux nœuds capteurs du réseau. La diffusion consiste à inonder le réseau de messages relayés par l'ensemble des nœuds afin de construire les meilleurs chemins d'une source à une destination donnée. Les inconvénients majeurs de cette technique sont l'épuisement énergétique du réseau ainsi que la consommation de la bande passante résultats de ce qu'on appel dans la littérature : tempête de diffusion (*the brodcasting strom*). En conséquence, un immense trafic (*overhead*) est généré à travers le réseau. Le protocole *DD* a été vite abandonné pour des techniques moins coûteuses qui se basent sur des transmissions sélectives telles que le Gossiping [30] [31] et le Rumor Routing [41].

Rumor Routing (RR) est un protocole de routage à base d'agents de rumeurs, spécialement conçu pour les environnements qui ne disposent pas de système de localisation géographique. La problématique soulevée dans ce type d'environnement, est la distribution non uniforme de l'information disséminée ainsi que les chemins avec boucles (*spiral like routing paths*). En effet, en l'absence d'une information de localisation et vu que le chemin de l'agent est aléatoire (a *random walk path*), la donnée risque d'être restreinte non loin des nœuds où l'événement a été détecté. Depuis son introduction en 2002 et jusqu'au jour d'aujourd'hui, plusieurs variantes et extensions du protocole *Rumor Routing* ont été proposées dans la littérature [42-47]. Leurs principal objectif, est d'améliorer la trajectoire du chemin emprunté par l'agent et la requête. Néanmoins, ils présentent toujours certaines insuffisances: 1. Le risque que les chemins des agents et des requêtes ne se croisent pas. 2. souvent les chemins construits pour l'écoulement de la donnée de la source vers le puits ne sont pas optimaux (en termes de nombre de sauts) et enfin en 3. Ceux sont des protocoles très dépendant de la localisation géographique des nœuds et de la connaissance topologique globale du réseau.

2. Objectif et principales contributions de la thèse

L'objectif du travail présenté dans cette thèse est d'aboutir à des protocoles de dissémination de données à base d'agents de rumeurs pour les réseaux de capteurs sans fil. Les protocoles proposés doivent remédier aux différentes lacunes des protocoles existants dans la littérature tout en garantissant la contrainte énergétique, le passage à l'échelle ainsi que des délais de réponse tolérables. Dans ce qui suit, nous résumons les principales contributions de cette thèse:

- A travers l'étude de divers protocoles de routage et de dissémination utilisant les agents de rumeurs dans les RCSF (unicast agent based routing protocols) [41-47] [51-60], nous proposons une nouvelle classification de ces derniers présentée dans le chapitre 2 (section II.3) où nous proposons de classer ces protocoles selon l'utilisation de l'information de localisation des nœuds ainsi que la stratégie adoptée pour le choix du prochain saut de l'agent.

- Dans un but de palier au problème des chemins avec boucles et de la distribution non uniforme de la donnée disséminée, nous commençons par proposer les protocoles: « Fast Rumor Agent » (FRA) [48-50] et « Efficient Data Access based on Rumor Dissemination » (EDARD) [51] (voir. Chapitre III). FRA et EDARD sont conçus pour des environnements ne disposant d'aucun système de localisation géographique. Ils implémentent une technique de dissémination des agents et des requêtes en ligne droite sur la base d'un choix sélectif du prochain saut. Leurs principales procédures sont: (a) le partitionnement du réseau en zones, (b) le routage des agents et des requêtes. Pour cela, ils exploitent une information de voisinage à deux sauts. Les structures de données proposées sont périodiquement réinitialisées pour faire face à la contrainte du passage à l'échelle. Dans EDARD, nous proposons une troisième procédure : (c) la création d'agents fils qui permet d'uniformiser la dissémination de la donnée et d'atteindre de nouvelles régions dans le réseau.

- En exploitant l'information locale sur le positionnement des nœuds capteurs, nous proposons le protocole Corridor Star Routing (CSR) [52]. CSR a pour objectif de procéder à une dissémination efficace et homogène de la donnée à travers le réseau. Sur la base de construction progressive de couloirs, il établit des chemins en réduisant considérablement le temps de réponse et le trafic généré. De ce fait, sont utilisation est appréciable dans les applications critiques (non tolerable delay applications).

- Enfin, afin de pallier au problème de coût engendré par l'utilisation des systèmes de positionnement géographique tel que le GPS (Global Positioning System), nous proposons le protocole Backbone Web Routing (BWR) [52]. BWR propose la construction d'un backbone sous forme de toile dans le réseau. Une région centrale sera définie comme région de rendez-vous pour les agents et les requêtes. Une étude de la complexité de l'algorithme de construction de la toile ainsi qu'une série de test sous environnement de simulation ont été conduits pour démontrer les performances de BWR.

3. Organisation du document

Le présent document est organisé en quatre chapitres :

- Le chapitre I introduit les réseaux de capteurs sans fil en définissant leurs principes de fonctionnement, leurs caractéristiques intrinsèques et quelques concepts liés au routage et à la dissémination de données. Il présente également un état de l'art couvrant les principaux protocoles de routage et de dissémination existants dans la littérature.

- Le chapitre II introduit la classe : « Unicast agent based routing protocols » ainsi que le protcole Rumor Routing (RR)[41] connu comme étant l'un des premiers travaux utilisant la notion d'agents de rumeurs pour améliorer la dissémination de données dans les RCSF. Dans ce chapitre, nous définissons également les concepts liés au routage à marche aléatoire (random walk routing) et proposons une nouvelle classification des protocoles de cette classe en présentant un nombre importants des extensions et variantes du protocole RR.

- Le chapitre III présente la conception des protocoles proposés afin d'améliorer la trajectoire des protocoles à marche aléatoire classiques. Il définit les structures de données utilisées ainsi que la stratégie de routage adoptées dans des environnements dépourvus de tout système de localisation géographique des nœuds. Enfin, une analyse des performances des protocoles proposés est présentée afin de les évaluer.

- Le chapitre IV présente la suite de nos contributions. Les protocoles qui y sont proposés considèrent des applications critiques qui ne tolèrent pas des retards dans le délai de réponse et qui s'exécutent sur des environnements hostiles et réelles. Dans ce chapitre, une description détaillée de ces protocoles est présentée et une analyse de leurs performances est réalisée par le biais d'outil de simulation.

CHAPITRE I. LA DISSÉMINATION DE DONNÉES ET LE ROUTAGE DANS LES RCSF.

Ce chapitre est une introduction à la thématique présentée dans les travaux de cette thèse. Il présente des généralités sur les Réseaux de Capteurs Sans Fil (RCSF) et quelques définitions et concepts liés à la dissémination de données et au routage. Une classification des protocoles de routage et de dissémination de données -la plus répondue dans la littérature- est par la suite présentée.

I.1 INTRODUCTION

L'émergence des Réseaux de Capteurs Sans Fil (RCSF) a inspiré beaucoup de sujets de recherche au cours de ces dernières années. Beaucoup de travaux ont été consacrés à l'organisation, la communication, la gestion de l'énergie, et à l'acheminement des données récoltées au sein d'un réseau de capteurs.

Développer un protocole de routage à faible consommation d'énergie est l'un des défis importants des réseaux de capteurs sans fil. Dans les réseaux sans fil classiques, les protocoles de routage permettent d'établir des routes entre les nœuds pour acheminer les paquets entre eux. Cependant, dans les réseaux de capteurs, les protocoles de routage établissent des routes entre tout nœud du réseau et la station de base pour assurer un routage performant.

A travers ce chapitre, nous allons présenter les principaux protocoles de routage et de dissémination de données des RCSF proposés dans la littérature.

I.2 DESCRIPTION D'UN RESEAU DE CAPTEUR SANS FIL

Un réseau de capteurs est constitué de milliers de nœuds appelés nœuds capteurs ou tout simplement capteurs, permettant de capter et de collecter des événements environnementales tels que : la température, le son, le mouvement d'objets. Ces nœuds peuvent avoir des positions fixes ou bien être déployés aléatoirement pour surveiller l'environnement. Les communications dans un réseau de capteurs se font souvent d'une manière multi-saut. L'écoulement des données se termine vers des nœuds spéciaux appelés nœuds puits ou stations de base « *sink* ». Ces nœuds-puits sont des bases de contrôle qui possèdent plus de ressources matérielles et permettent de collecter et stocker les informations issues des capteurs. En d'autres termes le fonctionnement d'un réseau de capteurs se déroule de la manière suivante : les nœuds sont déployés dans une zone appelée zone d'intérêt pour la surveiller. Lorsqu'un nœud détecte un évènement, il le traite localement et l'achemine vers la station de base via une communication multi-saut. Ce processus est illustré dans la figure I.1 [2].

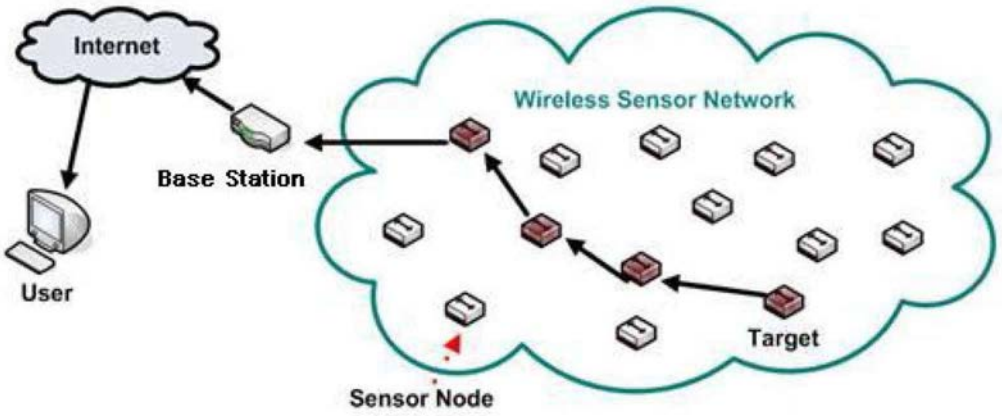


FIG I.1. Architecture d'un réseau de capteurs [2].

Les capteurs sont caractérisés par de courtes portées de communication afin de minimiser le chevauchement des signaux et de préserver la longévité du réseau. Dans certaines applications, l'accès aux capteurs est très difficile, les nœuds capteurs doivent pouvoir fonctionner longtemps sans avoir recours à des moyens de remplacement de leurs batteries. Ces conditions font que le paramètre d'énergie dans les RCSF est crucial surtout que la durée de vie du réseau y dépend fortement. L'ensemble des nœuds capteurs doivent collaborer afin de prolonger la durée de vie du réseau, réduire la consommation de la bande passante et maximiser la qualité du réseau.

Les différentes parties d'un nœud capteur peuvent être classées en six principales unités (voir figure I.2): 1. L'unité de transmission, 2. L'unité de traitement, 3. L'unité d'acquisition, 4. Les convertisseurs de signal ADC/DAC, 5. L'unité de contrôle d'énergie, 6. L'unité de stockage temporaire (mémoire) [3]. Il peut contenir également, suivant son domaine d'application, des modules supplémentaires tels qu'un système de localisation (GPS), ou bien un système générateur d'énergie (cellule solaire). On peut même trouver des micro-capteurs, un peu plus volumineux, dotés d'un système mobilisateur chargé de déplacer le micro-capteur en cas de nécessité. Il existe dans le monde plusieurs fabricants de capteurs. Nous citerons Crossbow, Cisco, Dalsa, EuroTherm, et Sens2B. Parmi ces capteurs, il existe quelques-uns qui sont capables de varier la puissance du signal émis afin d'élargir/réduire le rayon de communication et en conséquence la zone de communication.

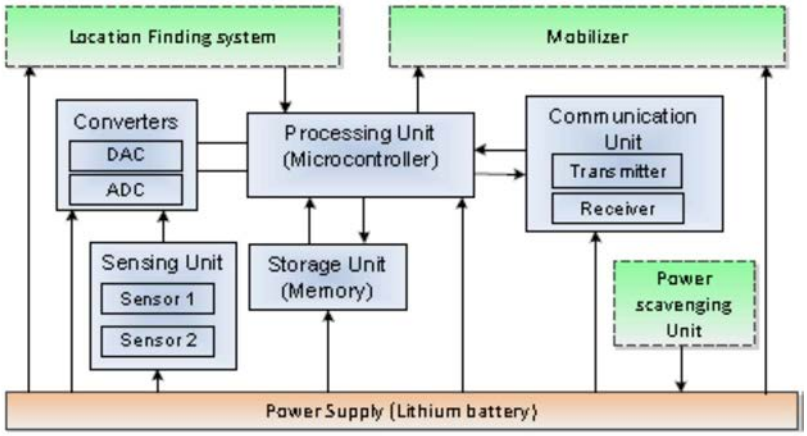


FIG I.2. Les différents composants d'un capteur.

Les capteurs dans l'unité d'acquisition interagissent physiquement avec l'environnement déployé. La donnée collectée est par la suite transférée vers les convertisseurs ADC/DAC pour la transformation : analogique vers le digital. Le microcontrôleur de l'unité de traitement reçoit la donnée digitale et y exécute un traitement en utilisant la mémoire. La donnée traitée est ensuite transférée vers l'unité de transmission pour être envoyée vers la station de base ou le prochain saut. La source d'énergie utilisée par un nœud capteur est une petite batterie en lithium. Les principales unités gourmandes en énergie sont l'unité de transmission et de traitement. La vie d'un capteur dépend fortement de son niveau d'énergie, il meurt immédiatement après épuisement de sa batterie. Pour faire face à la contrainte énergétique, le capteur peut changer son état parmi les trois modes suivants [3]: "active", "idle" et "sleep". Dans le mode actif, le capteur consomme beaucoup d'énergie car il est en mesure de recevoir et de transmettre des paquets de données. Le mode "idle" est défini lorsque le capteur exécute des opérations d'acquisition de données sans aucune transmission, il en consomme donc moins d'énergie par rapport au mode actif.

Dans le mode "sleep", le capteur éteint complètement son unité de transmission, l'énergie consommée dans ce mode est beaucoup moindre que dans les modes "idle" et "active".
Conventionnellement, en plus des changements d'états ou de modes que peut subir un capteur, il existe d'autres techniques pour la conservation d'énergie par exemple:

- L'utilisation de protocoles de routage qui consomment de l'énergie efficacement durant la transmission de données.
- Réduire-le plus possible- la puissance de transmission.
- Eviter l'overhearing (la sur-écoute), les collisions ainsi que l'écoute inutile.

I.3 DEFINITIONS ET CONCEPTS

Dans ce qui suit nous allons définir le routage dans les RCSF, la dissémination de données ainsi que tous les concepts qui y sont relatifs tel que la diffusion, l'agrégation et la fusion de données.

I.3.1 Le routage

Le routage est la méthode d'acheminement des données à la bonne destination à travers un réseau de connexion. Le but du routage est de trouver l'investissement de moindre coût en capacités nominales et de réserve qui assure le bon acheminement de l'information et garantit sa survie devant toute panne de lien ou de nœud [4]. Dans le cas des réseaux de petites tailles, le routage est généralement assuré par la technique d''inondation (la diffusion pure qui fait propager un paquet dans le réseau entier). Par contre, dans un réseau volumineux tel qu'un RCSF, le manque de données de routage concernant les destinations peut impliquer une large diffusion dans le réseau ce qui peut dégrader considérablement les performances du système caractérisé principalement par une faible bande passante et une capacité énergétique limitée. Dans un RCSF, chaque nœud est susceptible d'être mis à contribution pour participer au routage et pour retransmettre les paquets d'un nœud qui n'est pas en mesure d'atteindre sa destination directement ; tout nœud joue ainsi le rôle de poste de travail et de routeur [5].

Dans le cas où le nœud destination se trouve dans la portée de communication du nœud source, le routage devient évident et aucun protocole de routage n'est initié, ce qu'on appelle : envoi direct ou à un seul saut. Mais ce cas est généralement rare dans les réseaux Ad-hoc et les réseaux de capteurs. Un nœud source peut avoir besoin de transférer des données à un autre nœud qui ne se trouve pas dans sa portée de communication. La figure I.3 montre un exemple d'un réseau constitué de quatre nœuds. Le nœud A envoie directement un paquet à B sans besoin de routage puisque B est dans la portée de communication de A (envoi direct). Par contre, si le nœud A veut envoyer un paquet au nœud D, il doit utiliser les « services » des nœuds intermédiaires B et C puisque le nœud D n'est pas dans la portée de A. A envoie le paquet à B ; B transmet le paquet à C ; C, à son tour, transmet le paquet au nœud D. Cette technique est appelée routage multi-sauts.

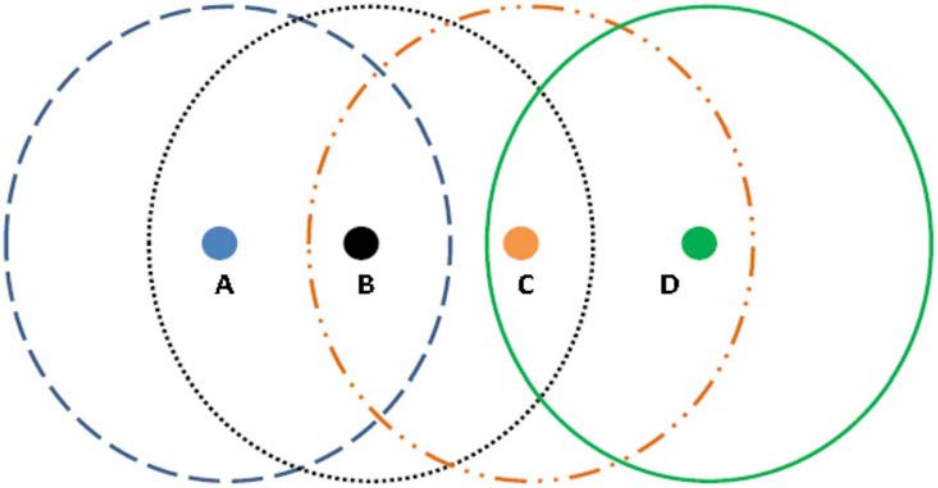


FIG I.3 : Communication multi-sauts entre A et D [5].

I.3.2 La Dissémination de données

La dissémination de donnée consiste à acheminer l'information depuis le nœud puits vers le reste des nœuds à travers les liens sans fils du réseau. C'est un processus vital pour un bon nombre de services dans les RCSF tel que la reprogrammation (transport de code) et la mise à jour des paramètres du système. En général, afin d'assurer la qualité de service (QoS) lors de la dissémination de donnée, quatre principales contraintes doivent être assurées [6] :

1. La fiabilité: afin d'assurer un réseau avec 100% de fiabilité, deux contraintes doivent être respectées : 1) la couverture de tous les nœuds du réseau ; 2) chaque nœud doit recevoir le bloc complet de données. Dans le cas contraire, l'inconsistance des données et la défaillance du réseau peuvent être causées.

2. Un délai d'acheminement réduit : lors de la dissémination de données, un nombre important de paquets occupent le canal de transmission ce qui peut bloquer le trafic ou rendre le réseau inefficace. De ce fait, le délai d'acheminement de données doit être le plus court que possible.

3. Une meilleure utilisation des ressources (énergie): l'énergie consommée lors de la dissémination de données consiste en la lecture/écriture des mémoires flash, la transmission radio ainsi que l'écoute du signal lors de l'état "idle" du capteur. Comme l'opération de lecture /écriture est inévitable, la transmission radio doit être minimisée et contrôlée.

4. La mise à l'échelle (scalability): le nombre ainsi que la densité des nœuds dans un RCSF ne doivent en aucun cas alterné le processus de dissémination de données. La mise à l'échelle d'un protocole de dissémination de données est dite satisfaisante si le temps de l'accomplissement de cette opération est linéaire par rapport aux nombre et à la densité des nœuds.

I.3.3 L'inondation (flooding)

C'est un mécanisme classique de transmission de donnée dans un réseau de capteur sans fil sans algorithme de routage ni algorithme de maintenance de topologie. Dans cette approche, chaque nœud recevant une donnée ou un paquet de contrôle le diffuse à tous les nœuds voisins, jusqu'à ce que le nombre maximum de sauts pour ce paquet soit atteint ou le paquet arrive à destination.

L'inondation est une technique facile à implémenter, ne nécessite pas une maintenance coûteuse de la topologie du réseau, ni des algorithmes complexes pour la découverte des routes, mais elle ne prend pas en considération les ressources énergétiques du réseau ainsi que le traitement de la redondance des données. Parmi ces inconvénients nous citons : [7]

• L'implosion: c'est l'envoi de messages dupliqués au même nœud. Par exemple, le nœud A diffuse son message a tous ces voisins. Le nœud D, recevra une copie dupliquée du même massage.

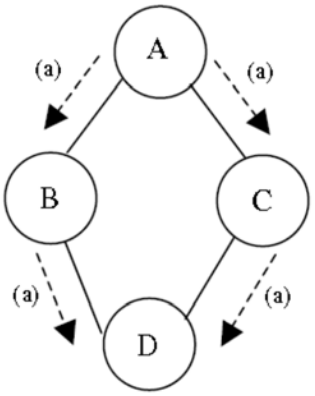


FIG I.4. Le problème d'implosion.

• L'overlap: Si deux nœuds capteurs assez proches l'un de l'autre partagent la même zone de captage, à un instant t donnée, ils peuvent transmettre la même information à leurs voisins ce qui crée une duplication de messages (au niveau du nœud C).

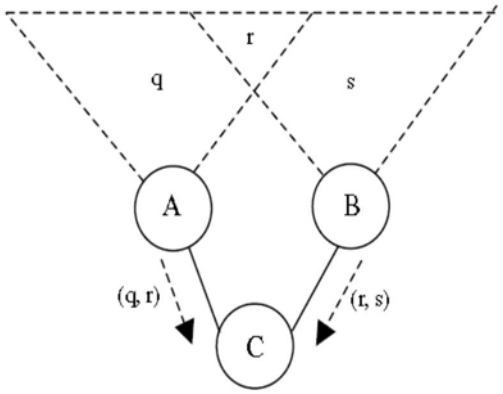


FIG I.5. Le problème d'overlap.

I.3.4 Le gossiping

C'est un mécanisme de diffusion plus souple que l'inondation. Un nœud ne diffuse pas les messages reçus à tous ses voisins, mais il les transmet à un seul, sélectionné aléatoirement. Les voisins font suivre le paquet de la même manière jusqu'à l'arrivée de l'information à destination. Le « gossiping » réduit les redondances de données mais l'arrivée des données à destination prend beaucoup plus de temps. [7]

I.3.5 L'agrégation ou la fusion de données

Dans un RCSF, plusieurs sources peuvent envoyer des données similaires. De plus, selon l'application, l'information captée par un seul nœud peut ne pas être très significative. Par exemple, pour un utilisateur désirant obtenir la température d'une région donnée, l'information fournie par un seul nœud n'est pas très représentative. La température globale de la région nécessite la collecte de toutes les données captées par les nœuds de cette région. Soulignons que dans un RCSF, le traitement de données est moins coûteux en consommation d'énergie que la communication. Il est donc plus avantageux de privilégier le traitement local des données en les agrégeant de sorte à réduire le nombre de transmissions et avoir des données plus significatives. Les données captées sont agrégées, si elles atteignent le même nœud au moment de leur acheminement vers le nœud puits. L'agrégation des données peut se faire en employant une fonction de combinaison des données telle que la suppression des redondances, le calcul du minimum, du maximum ou de la moyenne. Dans l'exemple de la figure I.6, le nœud E agrège les données provenant des nœuds A et C, alors que le nœud G agrège les données provenant des nœuds D et F. Par la suite, le nœud H agrège les données de G et F. [7]

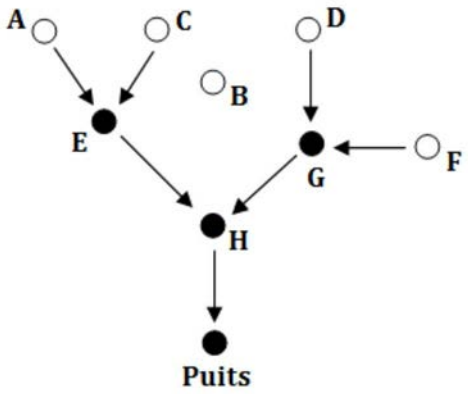
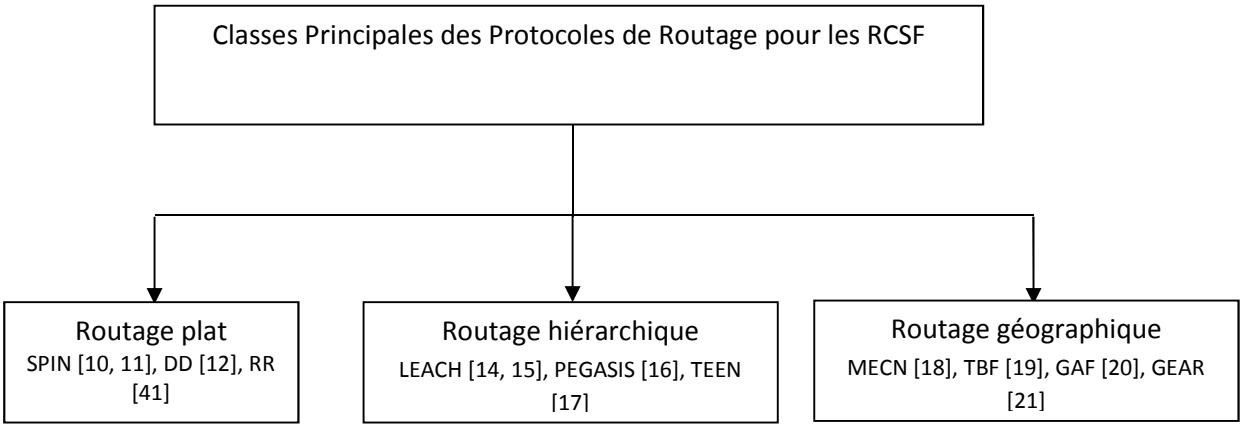


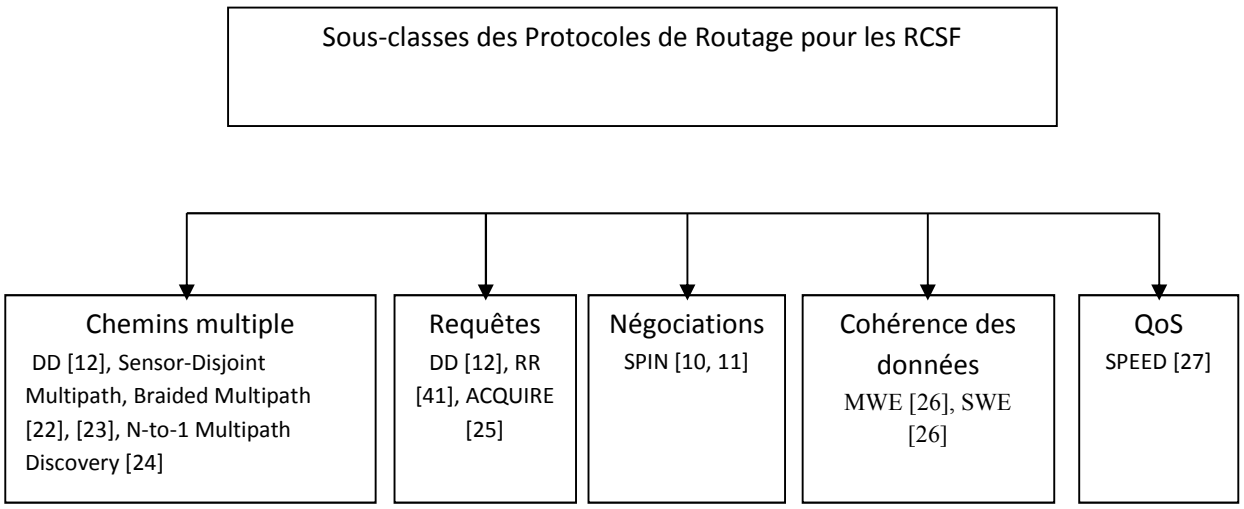
FIG I.6. Exemple d'agrégation de données [7].

I.4 CLASSIFICATION DES PROTOCOLES DE DISSEMINATION ET DE ROUTAGE DANS LES RCSF

Les protocoles de dissémination et de routage dédiés aux RCSF peuvent être classés selon différents critères par exemple: la structure ou la topologie du réseau, la stratégie de routage du protocole et la destination du paquet [3]. La plus part des travaux de classification dans la littérature [6-9] classifient les protocoles de routage et de dissémination de données selon la structure du réseau en trois classes principales : routage plat, hiérarchique et géographique. Dans chacune de ces trois principales classes, cinq sous-classes peuvent exister tel que proposé dans [4]. Cependant, un protocole de routage donné peut appartenir à une des trois principales classes et à une autre sous-classe à la fois. La figure I.7 suivante détaille cette classification.



(a) Classification selon la structure du réseau



(b) Classification selon la stratégie de routage du protocole

FIG I.7.Classification des protocoles de dissémination et de routage dans les RCSF [4].

I.4.1. Classification sur la base de la structure du réseau

La structure topologique du réseau joue un rôle important dans les opérations des protocoles de routage. Dans cette classe, les protocoles de routage sont classés comme suit: les protocoles plats, hiérarchiques et les protocoles géographiques [8].

I.4.1.1 Routage plat (centré-donnée)

Dans le routage plat, chaque nœud joue typiquement le même rôle et les nœuds capteurs collaborent pour accomplir la tâche globale du réseau [4]. A la différence des réseaux ad hoc classiques, les nœuds d'un réseau de capteurs sont dépourvus d'une identification globale (tel que l'adressage IP). Cette absence d'identification explicite est due à certaines caractéristiques des réseaux de capteurs. Tout d'abord, l'obtention et la gestion d'adresse nécessitent une étape de configuration qui peut être complexe, à cause du nombre très élevé de capteurs qui peuvent être déployés aléatoirement. De plus, étant donné la nature des applications des réseaux de capteurs, l'utilisateur ne s'intéresse pas à communiquer avec un nœud particulier du réseau mais envoie des requêtes, à tous les nœuds du réseau ou à une partie de celui-ci, afin d'identifier les nœuds concernés. Ces caractéristiques ont mené à la conception d'un nouveau type de protocoles appelé centré-données. Dans ces derniers, le routage ne se fait pas en fonction d'une adresse de destination mais suivant les données disponibles au niveau des capteurs. Les protocoles SPIN (Sensor Protocol for Information via Negociation) [10] [11] et DD (Directed Diffusion) [12] ont été les premiers travaux sur le routage centré données. Plus tard, d'autres protocoles basés sur DD, où suivant un concept similaire, ont été proposés tel que le Rumor Routing [41].

I.4.1.2 Routage Hiérarchiques

Ce type de routage procède à un groupement logique des nœuds du réseau (*clustering*) qui consiste à élire un chef de groupe puis à déterminer les nœuds qui feront partie du groupe. Dans chaque groupe (*cluster*), les nœuds envoient leurs données au chef de groupe (cluster Head) (voir figure I.8) qui envoi, à son tour la donnée au nœud puits (par l'intermédiaire des autres chefs de groupe, si nécessaire). Les chefs de groupe sont parfois supposés avoir un niveau d'énergie supérieur et une grande capacité de stockage et de calcul. Ils sont responsables du traitement, de l'agrégation et de la transmission de données. Alors que les nœuds les plus contraints en ressources énergétiques se consacrent uniquement au captage. De cette façon, le *clustering* peut contribuer considérablement à l'économie d'énergie, et à une meilleure mise à l'échelle (*scalability*) du système. En effet, le routage hiérarchique facilite l'agrégation des données, ce qui conduit à diminuer les redondances et les transmissions inutiles vers le nœud puits. D'autre part, le *clustering* permet aux nœuds d'effectuer des transmissions sur de courtes distances avec leur chef de groupe, ce qui minimise la consommation d'énergie.

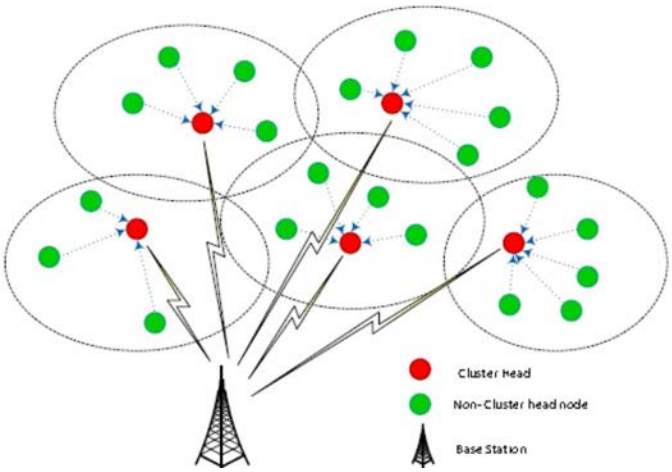


FIG I.8. Clustering hiérarchique.

LEACH (Low-Energy Adaptive Clustering Hierarchy) [14] [15] a été l'un des premiers protocoles de routage hiérarchique proposés pour les réseaux de capteurs. Il propose une élection périodique du chef de groupe à chaque période de temps (round) afin d'assurer une consommation uniforme de l'énergie par l'ensemble des nœuds du réseau. LEACH a inspiré la conception de plusieurs protocoles hiérarchiques tels que PEGASIS (Power- Efficient GAthering in Sensor Information Systems) [16] et TEEN (Threshold sensitive Energy Efficient sensor Network protocol) [17].

I.4.1.3 Routage géographiques

Certaines applications des réseaux de capteurs requièrent un adressage géographique pour spécifier les requêtes émises par le nœud puits. Ces applications nécessitent que les nœuds aient connaissance de leurs positions géographiques. L'idée des protocoles de routage géographique dans les réseaux de capteurs est de tirer profit des positions des nœuds pour prendre des décisions qui minimisent le nombre de messages transmis pendant le routage afin d'assurer une meilleure consommation d'énergie. Par exemple, ayant connaissance de la région concernée par la requête de l'utilisateur, les capteurs intermédiaires peuvent diffuser cette requête uniquement vers cette région, ce qui réduirait sensiblement le nombre de transmissions. Aussi, les informations sur les positions des nœuds permettent de calculer la distance séparant deux nœuds particuliers, de telle sorte à avoir une estimation de l'énergie nécessaire pour une communication entre ces nœuds. De cette manière, les informations sur les positions des nœuds peuvent être utilisées pour réaliser un routage efficace en énergie. Un nœud capteur peut obtenir sa position à l'aide des communications satellites dans le cas où les capteurs seraient équipés d'un récepteur GPS (Global Positioning System) ou en utilisant des algorithmes de localisation [62-63]. Parmi les protocoles géographiques utilisés dans les réseaux ad hoc et les réseaux de capteurs nous citons: MECN (Minimum Energy Communication Network) [18], TBF (Trajectory-Based Forwarding) [19], GAF (Geographic Adaptive Fidelity) [20], et GEAR (Geographic and Energy Aware Routing) [21].

I.4.2. Classification sur la base de la stratégie de routage du protocole

Les protocoles de routage dans cette classe sont classés comme suit : routage basé sur les chemins multiples, routage basé sur les requêtes, routage basé sur la négociation entre les nœuds, routage basé sur la cohérence des données, et routage avec qualité de service (QoS).

I.4.2.1 Routage basé sur les chemins multiples

Les protocoles de cette sous-classe utilisent des chemins de routage multiples au lieu d'un chemin unique entre une source et une destination. Ainsi, ils sont considérés comme tolérants aux fautes puisque ils assurent l'existence de chemins alternatifs lorsque le chemin principal est défaillant. Cette tolérance aux pannes est souvent la cause d'une consommation énergétique et d'un trafic de contrôle supplémentaires. Directed Diffusion (DD) [12] ainsi que Sensor-Disjoint Multipath [22] [23], Braided Multipath [22] [23] et N-to-1 Multipath Discovery [24] sont des exemples de protocole basé sur les chemins multiples.

I.4.2.2 Routage basé sur les requêtes

Dans le routage à base de requêtes, les nœuds à la recherche d'une donnée particulière ou d'un service créent et propagent des requêtes dans le réseau. Généralement, ces requêtes sont décrites dans un langage naturel ou des langages d'interrogation de niveau élevé. Rumor Routing [41] ainsi que Directed Diffusion [12] et ACQUIRE [25] sont des protocoles de cette classe.

I.4.2.3 Routage basé sur la négociation entre nœuds

L'idée principale du routage via la négociation est de supprimer l'information double qui résulte de la diffusion et d'empêcher l'envoi des données redondantes au prochain capteur ou à la station de base. Avant la transmission effective des données, ces protocoles échangeant une série de messages de négociation entre les nœuds en utilisant des descripteurs de données à un niveau élevé. SPIN [10], [11] est l'un des protocoles de cette classe.

I.4.2.4 Routage basé sur la cohérence des données

Différentes techniques pour traiter la cohérence des données circulant dans le réseau sont utilisées dans cette classe afin de minimiser la consommation en énergie d'un nœud capteur. Le principe du routage basé sur la cohérence ou la non-cohérence des données, est de décider sur quel nœud capteur doit s'exécuter l'agrégation des données selon le type d'applications utilisées. Dans beaucoup d'applications, la donnée captée est caractérisée par une corrélation et une redondance. Dans ce cas, il serait préférable que le puits reçoive l'essentiel de la donnée afin qu'il puisse la traiter rapidement et efficacement. C'est pour cela, que la donnée doit être traitée localement à chaque nœud intermédiaire. L'intérêt de ce processus de traitement tel que l'agrégation ou la fusion de donnée apparait surtout lorsque la donnée transmise est vectoriel et non scalaire (scalar). Par exemple, s'il s'agit d'envoyer une grande quantité de données par chaque nœud capteur comme des images, il serait préférable qu'un seul nœud responsable d'agrégation (*the aggregateor*) fusionne ces dernières et les envois au puits. Ceci dit, une contrainte de délais ajoutée à l'énergie consommée pour le traitement des données en locale sont enregistrées. De manière générale, les protocoles de cette sous-classe doivent assurer une balance entre la communication et la fusion de données. Multiple Winner Algorithm (MWE) [26] et Single Winner Algorithm (SWE) [26], sont des exemples de protocoles de routage de cette sous-classe.

I.4.2.5 Routage avec qualité de service

Les algorithmes utilisés par les protocoles de cette classe doivent satisfaire des critères de qualité de service lors du transfert de la donnée comme par exemple: la fiabilité, la latence et la bande passante. Ces algorithmes doivent assurer un équilibre entre l'assurance de la qualité de service requise et la consommation énergétique du réseau. SPEED [27] [28] est un bon exemple de protocole de routage avec QoS pour les RCSF qui fournit des garanties temps-réel de bout-en-bout.

Notons que d'autres sous-classes ont été aussi proposées dans la littérature [3] [29]. Par exemple dans [3], en se basant sur le critère de la destination du paquet c.à.d. le nombre de nœuds pour lesquels le paquet est destiné (un seul nœud (*unicast forwarding*), un ensemble de nœuds (*multicast forwarding*) ou tous les nœuds du réseau (*diffusion*)), l'auteur à identifier les sous-classes suivantes :

- *Gossiping and agent-based unicast forwarding:* Le principe de cette sous-classe consiste à éviter l'utilisation des tables de routage -souvent construites sur la base de diffusion de messages-. Autrement dit, au lieu que les messages soient diffusés à l'ensemble de leur voisins, ils sont envoyés à un nœud choisit aléatoirement via la technique du Gossiping ou bien un agent est créé et envoyé en unicast. Les protocoles de routage de cette classe réduisent considérablement le trafic réseau et présentent une solution au problème de l'implosion causé par la technique de diffusion. Néanmoins, ils engendrent une latence lors de la dissémination de la donnée. Gossip [30], Smart gossip [31] [32] et Rumor Routing [41] sont des exemples de ces protocoles.

- *Energy-efficient unicast forwarding*: Dans cette technique, le coût de transmission est calculé à chaque saut entre chaque paire de nœuds. Le prochain saut est choisi par rapport à l'énergie résiduelle du nœud. Cette technique améliore considérablement la durée de vie du réseau en choisissant les plus courts chemins consommant le moins d'énergie possible [33].

- *Mobility-aware routing*: Les protocoles de routage et de dissémination qui considèrent la mobilité des nœuds (capteur ou station de base) apportent de nouveaux challenges au RCSF. En effet, la mobilité peut être un facteur recherché pour résoudre certains problèmes de couverture et de connectivité. Par contre, la mobilité des nœuds comme celle du sink par exemple, nécessite des protocoles efficaces en énergie afin de garantir la délivrance des données depuis le nœud source vers le sink. Joint Mobility and Routing [33], Data MULES [34], TTDD [35] [36], SEAD [37], Dynamic Proxy Tree-Based Data Dissemination [38] sont des exemples de ces protocoles.

- *Heterogeneity-Based Protocols*: La plus part des protocoles de routage et de dissémination conçus pour les RCSF présument que tous les nœuds du réseau possèdent les mêmes capacités en termes de stockage, traitement, captage et communication. Dans de tels réseaux dits homogènes, où les liens de communication sont symétriques, une paire de nœud aura la même durée de vie si elle consomme la même quantité d'énergie. Cela dit, certaines applications réelles dans les RCSF utilisent des capteurs avec différentes capacités formant un réseau hétérogène. Cette hétérogénéité des nœuds permet d'améliorer la fiabilité et d'augmenter la durée de vie du réseau. [39][40] sont des exemples de ces protocoles.

I.5 CONCLUSION

A travers ce premier chapitre, nous avons introduit les réseaux de capteurs sans fil en définissant leurs principes de fonctionnement, leurs caractéristiques intrinsèques et quelques concepts liés au routage et à la dissémination de données. Aussi, nous avons présenté un état de l'art couvrant les principaux protocoles de routage et de dissémination qui existent dans la littérature.
Il est à noter, que les premiers protocoles de routage basés sur la diffusion ne sont plus appropriés à ce type de réseau puisque cette dernière est une opération très coûteuse. Les émissions fréquentes épuisent rapidement la batterie du capteur difficile à recharger. De nos jours, les travaux de recherche pour la conception de protocoles de routage efficaces en énergie s'intéressent à d'autres techniques telles que le gossiping ou exploitent de nouvelles architectures du réseau. Ils considèrent aussi des environnements plus réalistes où la contrainte de mobilité des nœuds existe et où la localisation de ces derniers est parfois difficile à obtenir.

De ce fait, la classe des protocoles à marche aléatoires (random walk routing protocols) ou « Unicast Agent based Routing protocols » a suscité l'intérêt d'un grand nombre de travaux de recherche. Le prochain chapitre sera consacré à l'étude détaillée des protocoles de cette classe et de leur apport pour améliorer la dissémination dans les RCSF.

CHAPITRE II. LES PROTOCOLES DE ROUTAGE À BASE D'AGENTS DE RUMEUR DANS LES RCSF (UNICAST AGENT BASED ROUTING PROTOCOLS).

Ce chapitre présente un état de l'art sur les protocoles de routage à marches aléatoires dits aussi : « Unicast agent based routing protocols ». Pour mieux les étudier, nous proposons de les classer en deux grandes classes : ceux qui routent les évènements et les requêtes sans avoir recours à un système de localisation géographique (free-location based routing protocols) et ceux dont le fonctionnement nécessite une information de localisation des nœuds (location based routing protocols). Une étude détaillée du traditionnel protocole Rumor Routing (RR) ainsi que ces différentes variantes et extensions est aussi présentée en raison de sa relation étroite avec nos contributions.

II.1 INTRODUCTION

Le principe du routage à base d'agents est très proche de la technique du *Gossiping* qui contrairement à celle de la diffusion, consiste à transmettre la donnée à un seul nœud voisin sélectionné aléatoirement ou selon une certaine probabilité. Les voisins font suivre le paquet de la même manière jusqu'à l'arrivé de l'information à destination [30]. Dans les protocoles de routage à base d'agent, dès qu'un évènement est capté dans le réseau, un paquet nommé agent *(agent)* est créé et propagé dans le réseau selon la technique du gossiping. Le long de son parcours, l'agent propage l'information sur l'évènement (méta data) telle qu'une rumeur. Lorsque le nœud puits est à la recherche d'une information ou d'un service particulier dans le réseau, il crée un paquet nommé requête *(query)* qui progressera d'une manière similaire à l'agent jusqu'à ce que leurs deux chemins se croisent. L'approche de routage utilisée dans cette classe de protocole réduit le trafic ainsi que les redondances de données mais l'arrivée des données à destination prend beaucoup plus de temps. Néanmoins, deux gains non négligeables pour les RCSF sont enregistrés : le premier, est la collaboration entre agents qui permet une synchronisation de leurs listes d'évènements réduisant ainsi le nombre de transmissions. Autrement dit, lorsque *n* agents collaborent entre eux, il en résulte un seul agent qui disséminera les *n* évènements captés à la fois. Le second, est la taille réduite du code transporté par l'agent qui consiste en quelques lignes de synchronisation de listes ce qui n'affecte pas la bande passante limitée des communications sans fil dans les RCSF.

Dans ce chapitre, nous présentons les concepts fondamentaux liés au routage à base d'agent et nous y proposons une nouvelle classification. Nous détaillons également l'un des premiers protocoles: « Rumor Routing » proposé par *Braginsky et Estrin en 2000* ainsi que ces différentes variantes et extensions.

II.2 CONCEPTS FONDAMENTAUX SUR LE ROUTAGE A MARCHE ALEATOIRE (THE RANDOM WALK ROUTING)

Beaucoup de protocoles de routage et de dissémination ont été proposés pour les réseaux de capteurs sans fil. Dans l'approche « Query driven » ou « Query based », la donnée est disséminée depuis le noud source vers l'ensemble des nœuds du réseau à la recherche d'une information particulière. Le protocole Directed Diffusion [12] [13] fut l'un des premiers protocoles utilisant cette approche. Par contre, la technique de diffusion sur laquelle il se base produit une surconsommation énergétique et affecte considérablement la bande passante du réseau. De ce fait, ses performances se dégradent vite face à la contrainte de mise à l'échelle et le temps de réponse augmente significativement de telle sorte que l'information recherchée devient inutile.

Dans l'approche « Event driven » « Event based », la dissémination de la donnée est déclenchée périodiquement - ou suite à l'apparition d'un évènement dans le réseau- depuis le(s) nœud(s) source(s) vers le nœud puits. Les protocoles hiérarchiques [16] [64-66] sont un exemple de cette

approche de routage. Dans ces derniers, le réseau est composé de clusters ou groupes et la donnée est agrégée et disséminée d'un cluster-head (chef de groupe) à un autre jusqu'à atteindre le sink. En reconsidérant l'architecture du réseau, ces protocoles visent à améliorer la consommation d'énergie. Par contre, l'étape d'initialisation du réseau peut parfois consommée beaucoup de ressources en énergie ce qui risque de diminuer la durée de vie du réseau.

L'approche hybride «Event based query-driven » fut alors proposée pour pallier aux problèmes des deux approches précédentes. Le protocole de routage « Rumor Routing » (RR) [41] est l'un des premiers protocoles dit à marche aléatoire proposés dans cette approche. C'est un protocole hybride qui se base sur la dissémination d'évènements et de requêtes dans le réseau. Dans RR, de petits paquets appelés *agents* propagent de manière aléatoire une information donnée suite à la détection d'un évènement dans le réseau. De manière similaire, le nœud puits propage des requêtes (*query)* à la recherche d'une information particulière dans le réseau. Lorsque le chemin d'une requête croise celui de l'*agent* correspondant, une route depuis le nœud source vers le puits est alors établie. Dans ce qui suit, nous allons définir les concepts et terminologies nécessaires afin de détailler le principe de fonctionnement du protocole RR.

II.2.1 Terminologies

- **Nœud** : fait référence à un dispositif statique doté d'une capacité énergétique limitée, capable d'accomplir des tâches telles que la détection d'évènements, l'envoie de requêtes et de notifications.
- **Evènement *(event)*:** est n'importe quel phénomène qui apparait dans une région du réseau, détecté par les nœuds sources.
- **Agent** : est un paquet responsable de la dispersion de la rumeur sur l'évènement capté, il possède un TTL *(Time To Live)* qui correspond au nombre de sauts qu'il est en mesure d'effectuer à travers le réseau. Chaque agent maintient une liste d'évènements et un historique de son parcours.
- **Requête *(query)*** : est une demande d'information ou un ordre de collecte d'information concernant un évènement. Elle est générée par le puits et possède aussi un TTL. Une requête est considérée comme non délivrée lorsqu'elle n'atteint pas sa destination avant l'expiration de son TTL.
- **Liste des voisins *(neighbor list)*:** la liste des voisins contient les identificateurs (IDs) des nœuds voisins pour chaque nœud. Cette table est construite via l'envoi des messages « Hello» ou *(Beacons)*.
- **Liste d'évènements *(event list)* :** dans une liste d'évènements, on retrouve le nom de chaque évènement capté ainsi que le nombre de sauts pour l'atteindre. Les nœuds et les agents possèdent leurs propres listes d'évènements.
- **Liste historique *(historical list)* :** elle contient un historique sur les nœuds préalablement visités par l'agent ou la requête.
- **Chemin d'évènement :** est le chemin créé par l'agent lors de la dispersion de la rumeur. Il représente le plus court chemin qui mène vers l'évènement.

II.2.2 Le protocole Rumor Routing (RR)

Le concept du routage à marche aléatoire consiste en l'acheminement des requêtes émises par le puits vers les nœuds source qui ont observés un évènement particulier. Un évènement est n'importe quel phénomène, détecté par les nœuds du réseau. Dès qu'un évènement apparait dans le réseau, le nœud source qui capte avec une certaine précision cet évènement crée un agent. Le rôle de l'agent consiste à faire propager une rumeur sur l'existence d'une donnée relative à l'évènement capté le plus loin possible dans le réseau. Notons que les agents sont considérés comme des messages actifs avec un petit code de synchronisation de liste; ils ne sont donc pas similaire aux agents mobiles utilisés dans les RCSF pour la collecte de données ou pour le déploiement dynamique des applications comme dans certains travaux tel que [67][68].

Lorsque le puits recherche une donnée particulière, il crée une requête. La requête est un paquet similaire à l'agent qui parcourt le réseau dans le seul but de rechercher un chemin qui mène vers l'évènement en question. Si le TTL de la requête expire avant que cette dernière rencontre le chemin vers sa destination alors elle est considérée comme non délivrée. Pour remédier à ce problème, on peut avoir recours dans certain cas à la technique de diffusion.

II.2.2.1 Les étapes d'exécution du protocole RR

Dans RR [41], chaque nœud maintient une liste de ses voisins, et également une table d'évènements. La liste des voisins peut être créée et maintenue d'une manière active en diffusant une requête ou d'une façon passive, par l'écoute des émissions des autres nœuds. Lorsqu' un nœud distingue un évènement, il l'ajoute à sa table d'évènements, avec une distance égale à zéro et crée par la suite un agent. Pour déterminer son prochain saut, l'agent devra choisir un nœud voisin non déjà visité. Si ce nœud n'existe pas, son choix sera aléatoire. Une fois arrivé sur un nœud, l'agent procède à une synchronisation entre sa liste d'événements et celle du nœud. Tous les voisins qui reçoivent ce broadcast vont aussi synchroniser leurs listes d'évènement. L'agent continu son voyage à travers le réseau de saut en saut jusqu'à l'épuisement de son TTL.

Lorsqu'un nœud puits s'intéresse à un évènement particulier, il crée une requête. Le puits transmet directement la requête s'il connaît le chemin menant à l'évènement. Sinon, il l'enverra dans une direction aléatoire. La requête est disséminée dans le réseau jusqu'à ce qu'elle croise le chemin d'un agent qui a observé l'évènement recherché. Si le nœud originaire de la requête (puits) conclut que sa requête n'a pas atteint la destination (après expiration du TTL), il peut la retransmettre, l'inonder ou même l'abandonner. En résumé, le processus d'exécution du protocole RR passe par les quatre étapes suivantes :

a. La détection d'évènements

Le protocole RR est conçu pour des réseaux avec des nœuds capteurs homogènes et une architecture plate. Lorsqu'un évènement se produit dans une région d'intérêt, il sera capté par un ensemble de nœuds assez proches formant la région à partir de laquelle une notification sera envoyée.

b. La notification

Lorsqu'un capteur détecte un évènement avec un certain degré de puissance, il génère immédiatement un agent qui parcourt le réseau en utilisant une marche aléatoire. Lors de son parcours, l'agent va notifier tous les nœuds sur son chemin afin que ces derniers gardent trace de l'évènement capté ainsi que de sa direction. Le TTL de l'agent créé sera défini par rapport à la taille du réseau. A la fin de cette étape, un ensemble de nœuds dans le réseau sera notifiés telle qu'illustré par la figure II.1

c. La génération de requêtes

Lorsqu'un nœud puits cherche une donnée relative à un évènement capté. Il crée une requête qui parcourt le réseau en utilisant une marche aléatoire et en interrogeant chaque nœud sur son chemin sur l'évènement recherché. Si la requête passe par un nœud déjà notifié, elle s'arrête et termine son parcours (voir figure II.1).

d. La transmission de données

Lorsque une route est construite entre la source et le puits à travers le croisement des chemins des agents et des requêtes, les données enregistrées dans le cache du nœud source peuvent alors s'écouler jusqu'au nœud puits en empruntant ce chemin.

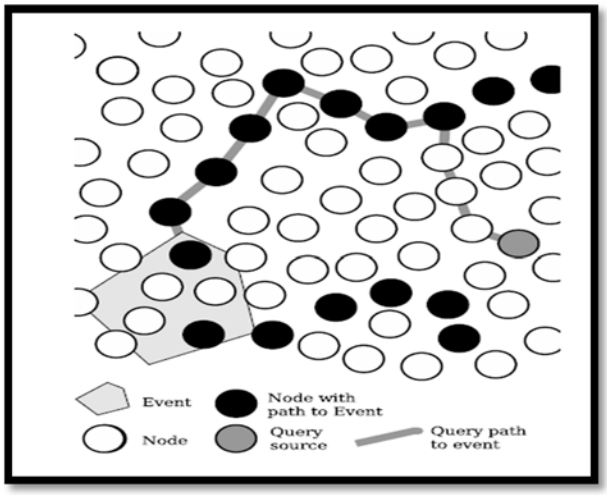


FIG II.1.Les étapes d'exécution du protocole RR.

II.2.2.2 La coopération entre agents dans RR

Dans les protocoles de routage à marche aléatoire, la coopération ou collaboration entre agents est très importante pour maintenir des chemins optimaux vers chaque évènement. Le principal rôle d'un agent est d'informer l'ensemble des nœuds qu'il rencontre dans son chemin de la liste d'évènement dont il est en possession. Pour cela, à chaque passage par un nœud, l'agent procède à une synchronisation entre sa liste d'évènements et la table de routage du nœud. Par exemple, comme illustré dans la figure II.2, lorsqu'un agent arrive sur le nœud A à partir du nœud B, il met à jour la route vers l'évènement (event1) qui était à quatre sauts en direction du nœud C par la nouvelle route qui consiste en trois sauts uniquement en direction du nœud B. Aussi, comme

un ancien agent propageant l'évènement (event2) est déjà passé par le nœud A, le nouvel agent, procède à une agrégation de rumeurs et propage les évènements (event1) et (event2) à la fois.

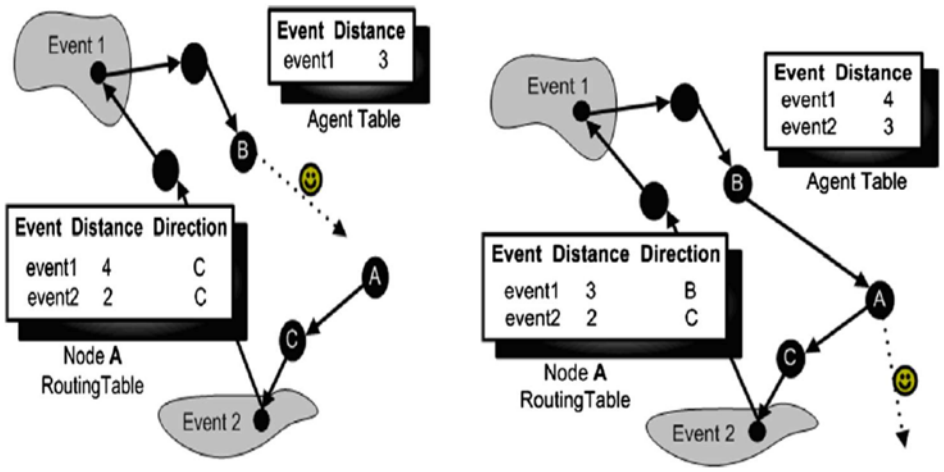


FIG II.2. Synchronisation des listes d'évènements et des tables de routage dans RR.

II.2.2.3 Problématique liée à la marche aléatoire

Un des problèmes majeur des algorithmes de routage à marche aléatoire est l'absence de tout critère de sélection du prochain saut lors de la dissémination des agents et des requêtes dans le réseau. Par conséquent, l'instabilité de la longueur des routes créées est un risque potentiel qui peut causer une surconsommation énergétique surtout dans les réseaux à grande échelle. De plus, un bon nombre de routes qui ne favorisent pas une dissémination uniforme de la rumeur peuvent être créées telle que les routes en spirales ou en zigzag (figure II.3 I et II). Aussi, dans certains cas, les chemins des agents et des requêtes peuvent ne pas se croiser si les routes sont parallèle ou concentriques tel qu'illustré par la (figure II.3 III).

En résumé, le protocole RR présente les limites suivantes :

- ✓ Rumor Routing est dédié aux réseaux où le nombre d'évènements est petit par rapport au nombre de requêtes. Autrement dit, il ne pourra pas assurer une conservation d'énergie pour un grand nombre d'évènements car le coût en énergie pour maintenir les agents et les table des évènements ne peut pas être amorti.
- ✓ La taille de l'agent risque de croître due à la liste historique et celle des événements.
- ✓ Absence de maintenance du chemin découvert : l'agent dans RR, maintient un seul chemin de la source vers le puits pour chaque évènement. Si plusieurs requêtes sont générées pour un même évènement, la majorité de ces requêtes devront parcourir au moins une partie de ce chemin ce qui épuisera rapidement les nœuds le constituant.
- ✓ Dans RR, l'agent choisit son prochain saut aléatoirement. Donc, les voisins proches ont la même probabilité d'être sélectionnés que ceux qui sont distants. Cela risque de faire circuler la rumeur autour d'une région très restreinte du réseau.

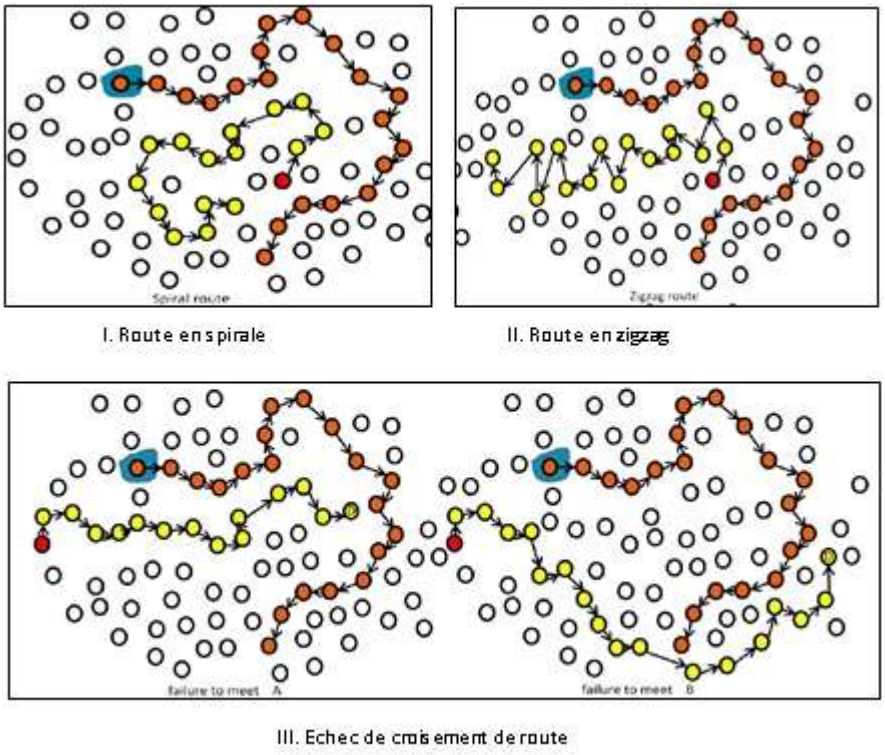


FIG II.3. Cas extrêmes des chemins générés dans RR.

Afin de pallier à ces inconvénients et minimiser la consommation d'énergie dans le réseau par les mêmes moyens, une approche consiste à contrôler le comportement de la marche aléatoire, de raccourcir les chemins créés et d'assurer leurs intersections. Dans la section suivante, nous présentons les principales extensions et variantes du protocole RR qui tentent de résoudre cette problématique.

II.3 LES PROTCOLES DE ROUTAGE A BASE D'AGENTS DE RUMEURS

Afin d'améliorer la qualité des chemins créés dans RR, beaucoup de ses extensions ont été proposées dans la littérature [42-61] [69-75]. Un de leur principal objectif est de disséminer l'information sur des chemins aussi droits que possible pour minimiser la consommation d'énergie et accélérer le processus de rencontre entre les agent et les requêtes. Par ailleurs, il est évident que construire des chemins droits est une tâche simple dans les réseaux avec une distribution uniforme des nœuds et où l'information de localisation géographique des nœuds est disponible. Par contre, certains travaux récents par exemples [62] et [63], affirment que l'utilisation des algorithmes et des systèmes de localisation telle que le GPS, engendre des erreurs d'estimation et un surcoût énergétique. Pour cela, nous proposons dans cette section de classer les protocoles de routage à base d'agent *(agent based unicast forwarding)* en deux principales classes : 1. Ceux qui routent les évènements et les requêtes sans avoir recours à un système de localisation géographique (*free-location based routing protocols)*. Dans cette première classe, nous nous somme basé sur la stratégie de sélection du prochain saut adoptée par l'agent comme critère de classification afin d'identifier quatre autres sous-classes et 2. Ceux dont le fonctionnement nécessite une information de localisation des nœuds *(location based*

routing protocols). La classification proposée est illustrée par la figure II.4, elle est suivie d'une présentation des protocoles de routage représentatifs de chacune de ces classes.

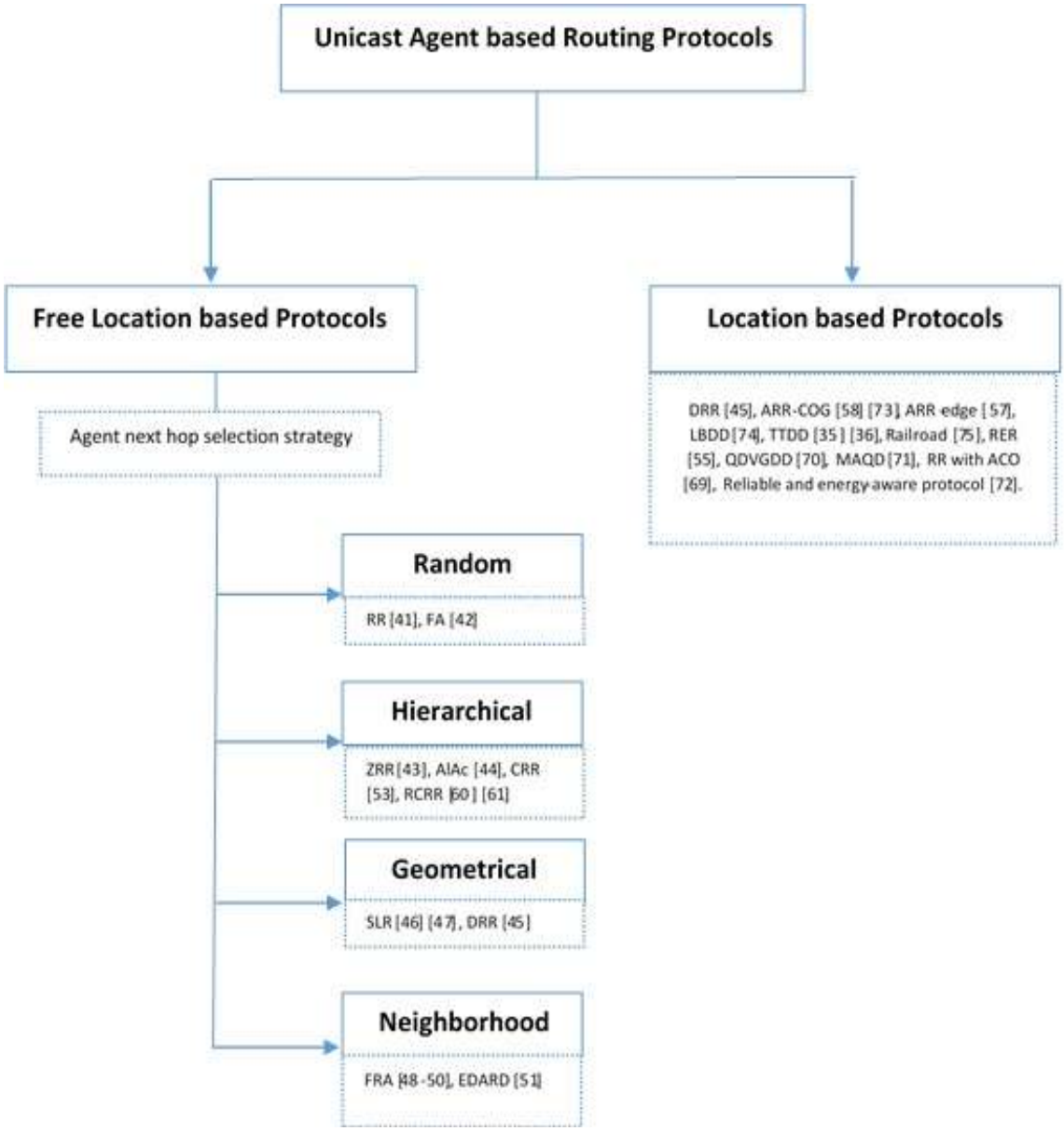


FIG II.4. Classification des protocoles de routage à base d'agents proposée.

II.3.1 Les protcoles de routage indépendants des système de localisation géographique (Free location based routing protocols)

Les protocoles de routage de cette classe disséminent les agents et les requêtes dans le réseau sans avoir recours à un système de localisation géographique. Dans les premiers travaux, le choix du prochain saut était tout simplement aléatoire, ce qui a causé plusieurs problèmes tels que la distribution non uniforme de la donnée ainsi que les chemins avec boucle. Plus tard, certains travaux ont tenté de guider le parcours des agents et des requêtes afin d'optimiser la contrainte d'énergie et de latence. Dans cette section, nous détaillons le principe de fonctionnement de ces protocoles selon la stratégie de sélection du prochain saut adoptée par l'agent.

II.3.1.1 Stratégies aléatoires

Dans cette sous classe, le choix du prochain saut lors de la dissémination d'agents et de requêtes se fait de manière aléatoire. Le protocole Rumor Routing (RR) [41] et Forking Agents (FA) [42] sont deux protocoles de cette sous classe. Dans [42], Haenselmann et Effelsberg ont proposé un protocole qui assure une meilleure distribution de l'information à travers le réseau. Ils utilisent pour cela la notion d'agents maitres (*master agents*), qui sont créés immédiatement après la détection d'évènements dans le réseau. Parmi les champs contenus dans le paquet de l'agent maitre, on retrouve un champ décrivant l'information à propager et un champ accusé de réception (*ACKnowledgment ou ACK*) pour une transmission sans perte de paquets. Les agents maitres ont la propriété de se diviser et de se multiplier en plusieurs agents fils *(child agents)* via le *forking process*. La transmission des agents fils se fait sans l'utilisation d'ACK et leur parcours est aléatoire tout comme ceux des agents maitres.

L'idée du protocole FA est de propager l'information sur un évènement particulier le plus loin possible de la région où il a été capté. De plus, son processus de création d'agents fils réduit considérablement le nombre de transmissions. En effet, transmettre *n* agents depuis un nœud source génère plus de trafic que transmettre un seul agent maitre qui créera à son tour (n-1) agents fils après un certain nombre de sauts. Les résultats de simulation du protocole FA montrent une dispersion plus large et mieux équitable de l'information couvrant des régions plus vastes avec une réduction du trafic généré (voir figure II.5). Cependant, aucune stratégie de dissémination n'est définie quant au choix du prochain saut des agents.

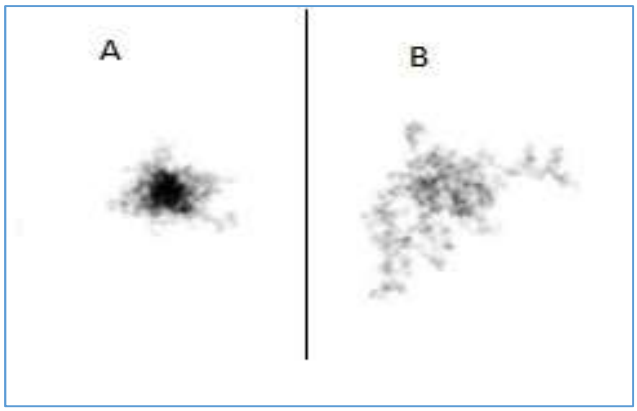


FIG II.5. Etendue de la dissémination des agents dans les protocoles RR (**A**) et FA (**B**).

II.3.1.2 Stratégies hiérarchiques ou basées sur la topologie du réseau

La topologie du réseau dans les protocoles de cette sous classe est reconsidérée puisqu'il ne s'agit plus de la topologie plate. La stratégie de sélection du prochain saut de l'agent est définie selon l'architecture topologique du réseau organisée en clusters ou en arbre. Parmi les protocoles de cette sous classe, nous citons: *Zonal Rumor Routing protocol(ZRR) [43], Along & Across algorithm (AlAc) [44], Clustering Rumor Routing protocol (CRR) [53] et Relative Coordinate Rumor Routing protocol (RCRR) [60] [61]*.

A. Le protocole ZRR : Zonal Rumor Routing for Wireless Sensor Networks (2005)

Un des inconvénients du protocole *RR* est la possibilité qu'une rumeur reste concentrée dans une petite région du réseau à cause du choix aléatoire du prochain saut de l'agent. Dans [43], les auteurs tentent de remédier à cette problématique en partitionnant le réseau en zones.

Le pseudo algorithme du protocole ZRR passe par trois grandes étapes :

a) La création de zones

Chaque nœud à une probabilité d'être sélectionné comme le « leader de la zone ». Une fois sélectionné, ce dernier diffuse via un broadcast, l'annonce de création de sa zone à ses voisins. Si le nœud récepteur appartient déjà à une autre zone, il répond par un broadcast contenant la paire (id zone, id nœud). Sinon, il se joindra à la nouvelle zone et invite à son tour ses voisins à rejoindre sa zone. Les messages de création de zones vont se propager dans le réseau jusqu'à ce que les nœuds arrêtent de recevoir des invitations. Après un certain temps, chaque nœud devient exactement membre d'une seule zone.

b) Le routage des agents

Lorsqu'un nœud détecte un évènement, il génère un agent, et enregistre l'évènement avec une distance égale à zéro. L'agent créé, fait de même et enregistre l'évènement qu'il devra propager. Afin d'éviter de repasser par les zones déjà visitées, l'agent insère l'Id de zone de son nœud

créateur dans une liste historique. Pour déterminer son prochain saut, l'agent donnera toujours la priorité aux nœuds voisins appartenant à une zone différente de celle où il se trouve. Si ce nœud est trouvé, l'agent lui sera envoyé via un broadcast. Sinon, le choix sera aléatoire comme dans le protocole *RR*. Lorsqu'un agent arrive au niveau d'un nœud, il synchronise sa liste d'événements avec celle du nœud; tous les nœuds voisins qui reçoivent ce broadcast font de même.

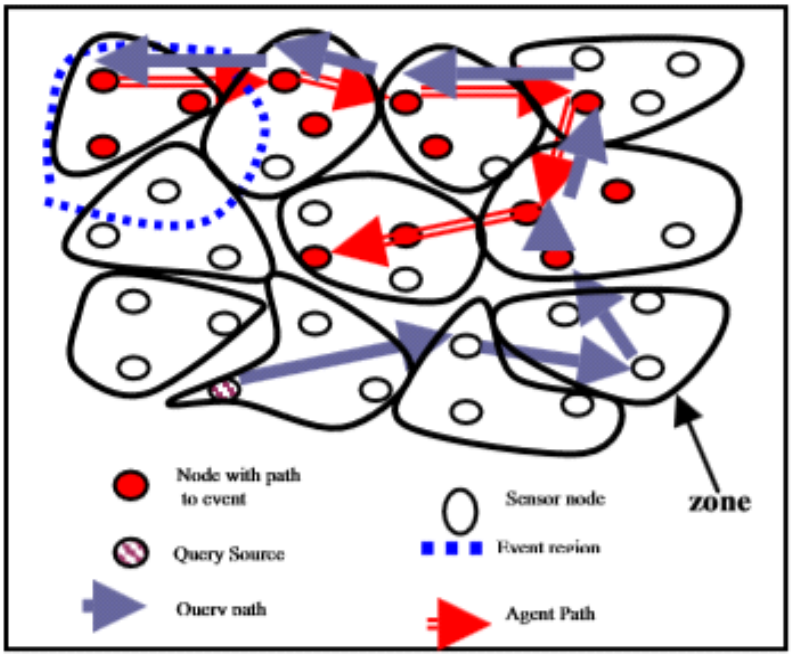


FIG II.6. Stratégie de routage des agents et des requêtes dans ZRR.

c) Le routage des requêtes

Lorsqu'un nœud s'intéresse à un événement particulier, il génère une requête. Le principe de routage des requêtes est similaire à celui des agents, sauf que la requête consulte la liste d'évènements de chaque nœud dans son parcours afin de trouver un chemin vers l'événement recherché. Dans le cas contraire, elle continue son parcours jusqu'à l'épuisement de son *TTL*. Les résultats de simulation montrent que le protocole ZRR atteint ses meilleures performances en termes de pourcentage et de coût de délivrance de requêtes autour d'un certain nombre de zones appelé nombre optimal de zones noté, Nz :

$$Nz = N/Zo \ldots\ldots [43].$$

Où N est le nombre de nœuds du réseau et *Zo* est la taille optimale d'une zone, donnée par la formule suivante:

$$Zo = c^2 * d \ldots\ldots\ldots [43]$$

Où *c* est le rang de communication du nœud et *d* est la densité du réseau.

Discussion

Le protocole ZRR semble avoir une bonne approche pour disséminer les agents et les requêtes vers des régions lointaines du réseau. Cependant, ses performances peuvent se dégrader face au critère de la mise à l'échelle. En effet, lorsque le nombre de nœuds du réseau devient important, les structures utilisées ne pourront pas stocker en mémoire tout l'historique du parcours des agents et des requêtes. De plus, ZRR génère un coût supplémentaire en trafic et énergie qui devrait être évalué lors de la phase de création de zones.

B. Le protocole AlAc : Along & Across algorithm for routing events and queries in wireless sensor networks (2005)

Along & Across algorithm (AlAc) [44] a été conçu pour les réseaux de capteurs à forte densité. Dans AlAc, le réseau est organisé en arbre à niveaux de telle sorte que chaque niveau « *i* » représente les nœuds qui sont à une distance de *i* sauts de la racine. Il comporte trois grandes étapes: la construction de l'arbre à niveaux, la distribution des évènements et enfin la propagation des requêtes.

a) La construction de l'arbre

La première étape de l'algorithme consiste à organiser le réseau en arbre à niveaux initié par un nœud racine aléatoire. Le nœud racine génère un paquet *Tree-build* <*SenderID, Hop*>, où *SenderID* et *Hop* sont initialisés à l'ID du nœud racine et 0 respectivement. Ce paquet est ensuite diffusé à tous les voisins de la racine. Lorsqu'un nœud reçoit un premier paquet *Tree-build*, il enregistre la valeur du champ *Hop* puis l'incrémente, remplace *SenderID* par son propre identificateur et retransmet le paquet à tous ses voisins via le broadcast. Si un deuxième paquet *Tree-build* est reçu par le même nœud, il compare la valeur du champ *Hop* avec celle qui est stockée localement. Si la valeur mémorisée est supérieure à celle reçue, le nœud met à jour la valeur stockée, incrémente la valeur reçue et retransmet à son tour le paquet. Sinon, il l'ignore. La diffusion du paquet continue à travers le réseau jusqu'à ce que chaque nœud construise sa liste de voisin avec leurs niveaux respectifs. Un exemple d'un arbre à 6 niveaux est représenté sur la figure II.7 où le nombre dans chaque nœud représente son niveau par rapport au nœud racine (0).

b) La distribution des évènements

Lorsque un nœud détecte un évènement, il l'ajoute dans sa table d'évènements et forme un paquet *Event-Dist* < *EventAttributes, EventSrcLevel*> où *EventAttributes* est une description de l'évènement capté et *EventSrcLevel* est le niveau du nœud source qui a capté l'évènement. Pour choisir le prochain saut du paquet *Event-Dist*, le nœud source accorde une priorité à ces voisins de même niveau afin de distribuer l'évènement en cercle. Si ce voisin n'existe pas, il transmet le paquet à des voisins de niveaux inférieurs *(EventSrcLevel -1)* ou supérieurs *(EventSrcLevel + 1)* pour tenter de le distribuer sur ces derniers. A la réception d'un paquet *Event-Dist*, chaque nœud met à jour le champ *EventAttributes* dans sa table d'événement, définit le chemin vers cet évènement et le transmet de manière similaire que la source. La figure II.7 illustre la distribution d'un évènement capté par un nœud de niveau 4 en cercle à travers les nœuds du même niveau.

c) La propagation des requêtes

Lorsqu' un nœud reçoit une requête, il vérifie d'abord sa table d'évènement pour voir s'il y a une correspondance. Sinon, il forme un paquet *Query-Prop <QueriedEventAttributes, HopList>* où *QueriedEventAttributes* est une description de l'évènement recherché par la requête et *HopList* est utilisé pour enregistrer le chemin traversé par le paquet. Toute requête générée par le nœud puits est distribuée à travers le réseau selon une direction (soit vers les niveaux inférieurs ou supérieurs). Lorsqu'un nœud reçoit un paquet *Query-Prop*, il met à jour le champ *HopList* puis vérifie sa table d'évènement pour voir s'il y a une correspondance. Si oui, il dirige le paquet vers le voisin qui conduit à l'événement en fonction de sa table d'événements. Dans le cas contraire, il transmet le paquet à un voisin dans une direction opposée à celle de l'expéditeur. Par exemple, il transmet un paquet reçu à partir d'un voisin du niveau supérieur à un voisin de niveau inférieur. S'il n'y a pas de voisin (par exemple, le nœud a déjà le niveau le plus bas ou le plus élevé), cela signifie que la requête n'aura pas de réponse dans la direction choisie. Dans ce cas, le nœud tenant le paquet forme un paquet *Query-Failed* qui est identique au paquet *Query-Prop* et le transmet au nœud puit en utilisant le chemin inverse. Lorsque le puits reçoit le paquet *Query-Failed*, il forme un nouveau paquet *Query-Prop* et le transmet dans une direction opposée à celle qui a été sélectionnée auparavant. Si le puits reçoit des paquets *Query-Failed* des niveaux supérieurs et inférieurs à la fois, cela signifie que la requête n'aura pas de réponse. Dans ce cas, la requête est inondée sur l'ensemble du réseau de capteurs. La figure II.7 illustre un exemple d'une requête initiée à partir du niveau 2 qui se propage initialement vers les niveaux inférieurs jusqu'au nœud racine puis rebrousse son chemin vers les niveaux supérieurs pour trouver un chemin vers l'évènement recherché.

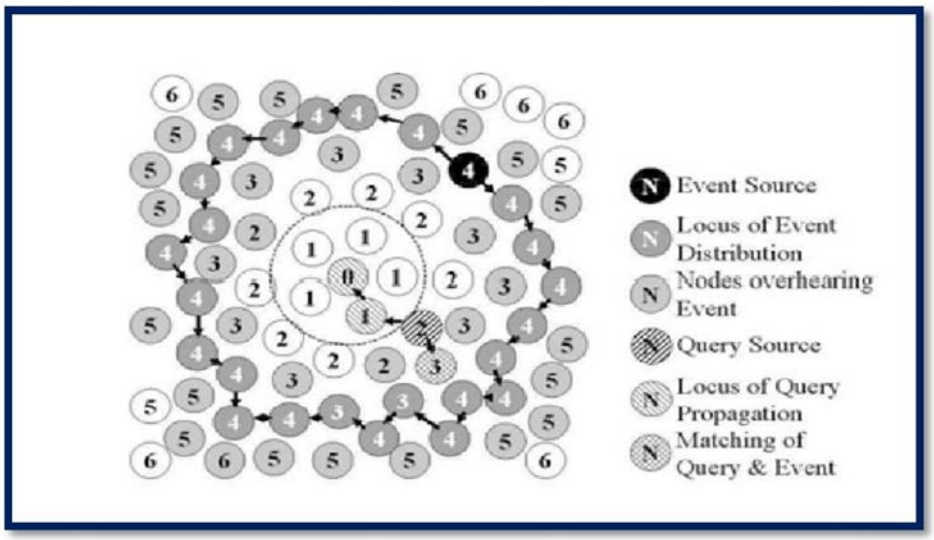


FIG II.7. Stratégie de routage des évènements et des requêtes dans l'algorithme AlAc.

Discussion

Les performances de l'algorithme AlAc ont montré un gain considérable en terme de nombres de transmissions et donc en consommation énergétique (jusqu'à moins de 72.6 %) par rapport au protocole RR. L'inconvénient majeur de cet algorithme réside dans le fait que les chemins

créés sont plus longs que nécessaire ; ils ne sont pas donc optimaux. Par conséquent, l'algorithme AlAc a connu des optimisations pour permettre de raccourcir les chemins inverses dans certains cas.

C. Le protocole CRR : Energy-Efficient Clustering Rumor Routing Protocol (2010)

Le protocole CRR [53] a été proposé dans un objectif d'économie d'énergie. Son principe de fonctionnement se base sur les trois caractéristiques suivantes: a) un réseau structuré en clusters; b) une stratégie pour un choix sélectif du prochain saut et c) un mécanisme de rotation locale et globale pour la réélection du chef de cluster ou ClusterHead (CH). CRR considère un réseau de capteurs où les nœuds sont distribués aléatoirement et où chaque nœud est en mesure de connaitre son taux d'énergie résiduelle. Dans CRR, le réseau est organisé en plusieurs clusters. Chaque cluster a un seul ClusterHead (CH) et plusieurs membres. Les membres d'un cluster ne peuvent communiquer qu'avec leurs CHs respectifs. Le fonctionnement du protocole CRR est décrit selon les étapes suivantes:

a) La construction de clusters

Dans CRR, les nœuds CHs sont choisis aléatoirement. Une fois sélectionnés, ils commencent la diffusion de messages pour la construction de clusters. Si le nœud recevant ce message n'est pas encore leader d'un cluster (CH), alors il va décider quel cluster rejoindre selon la puissance du signal reçu. Le nouveau nœud membre envoie par la suite un ACK au CH contenant son identificateur. Ce dernier sera stocké dans la liste des membres du CH et l'identifiant (ID_{CH}) sera aussi stocké au niveau de ce nœud. Par contre, si le nœud recevant le message de construction de cluster est déjà un CH alors il sauvegardera seulement l'ID du CH du message dans sa liste de voisinage (*Neighboring Clusterhead Queue*). Cette première étape du protocole CRR se termine lorsque chaque nœud se trouve membre d'un cluster.

b) Le routage d'évènements et de requêtes

De manière similaire aux protocoles RR et ZRR, dès qu'un évènement particulier est détecté par un nœud, ce dernier génère un agent et met à jour sa liste d'évènement. La principale différence réside dans le choix du prochain saut de l'agent. Dans CRR, les agents ainsi que les requêtes sont immédiatement transmis vers le CH du cluster dans lequel ils sont créés. En consultant sa liste (*Neighboring Clusterhead Queue*), le CH les transmet à leur tour vers d'autres CHs jusqu'à l'épuisement de leurs TTL respectifs. La figure II.8 illustre le principe de routage des agents et des requêtes dans le protocole CRR.

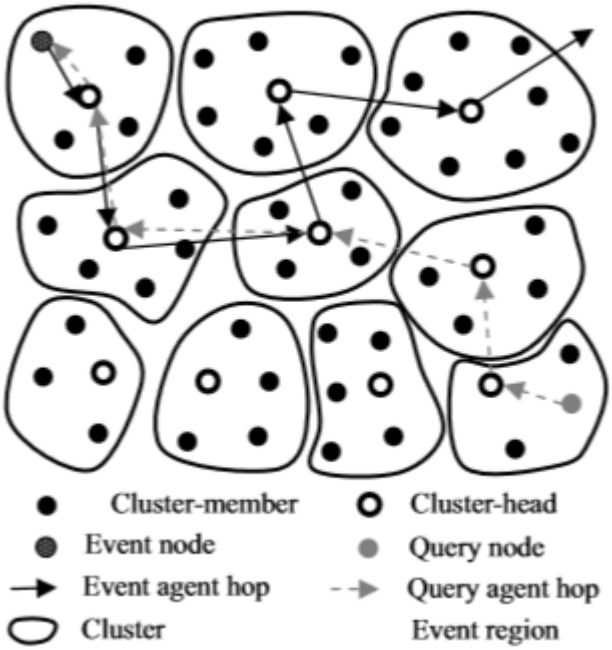


FIG II.8. La stratégie de routage entre CHs implémentée dans CRR.

c) La maintenance de ClusterHeads

Dans le protocole CRR, la tâche d'un CH est le traitement de tous les messages envoyés par les nœuds membres ainsi que la réélection d'un nouveau CH lorsque son niveau d'énergie est en dessous d'un certain seuil. Pour cela, deux types de rotations pour le remplacement de CH sont proposés : 1. un schéma de remplacement local et 2. un autre global. Dans le schéma de remplacement local, tous les membres d'un cluster envoient leur niveau d'énergie résiduelle à leur CH. Ce dernier calcul un certain seuil T_e donné par la formule suivante [53]:

$$T_e = \frac{p}{m}\sum_{i=1}^{m} E_{curr}(i), 0 < p < 1.$$

Où : $E_{curr}(i)$ est la valeur de l'énergie courante du nœud i, m est le nombre de nœuds dans le cluster et p est un paramètre qui varie selon le type d'application.

Lorsque l'énergie résiduelle du CH est inférieure à ce seuil, alors il sera remplacé par un autre nœud du cluster ayant le meilleur niveau d'énergie résiduelle et satisfaisant la formule suivante [53]:

$$E_{new-head} = Max\ E_{curr}(i), i = 1, 2, \ldots, m.$$

Si un nœud CH tombe en panne d'énergie avant le processus de rotation locale, le premier nœud voisin qui détecte ce problème s'élie comme nouveau CH afin de lancer une nouvelle réélection parmi les nœuds du cluster. Dans le schéma de rotation globale, tous les CHs envoient leurs énergies résiduelles à un CH sélectionné aléatoirement. Ce dernier calcule une moyenne et un seuil. Si la moyenne de l'énergie résiduelle des CHs est inférieure à ce seuil, alors la rotation

globale est lancée.

Discussion

La comparaison des performances du protocole CRR avec les protocoles conçus pour des réseau ayant une architecture plate tel que RR a montré une augmentation significative de l'énergie résiduelle sur l'ensemble des nœuds et donc une meilleur durée de vie du réseau. Les chemins établis, sont optimaux en termes de nombre de sauts et donc le taux de délivrance de données est amélioré. Ceci s'explique par la nature des transmissions exclusives entre CHs. Cela dit, CRR suppose que tous les CHs sont à portée de communication ce qui n'est pas toujours le cas dans les réseaux avec une distribution aléatoire des nœuds. Souvent, dans les architectures en clusters, les CHs doivent passer par plusieurs nœuds passerelles pour pouvoir établir une communication ce qui va influencer sur le nombre de saut des chemins établis.

D. Le protocole RCRR : Relative Coordinate Rumor Routing (2010)

Dans [60] [61], les auteurs présentent un protocole de routage hybride nommé : Relative Coordinate Rumor Routing (RCRR) dérivé du traditionnel RR [41]. Dans RCRR, un nouveau concept de coordonnées relatifs est proposé afin de router les agents et les requêtes sur la base de l'information topologique du réseau. Pour pouvoir utiliser les coordonnées relatives, RCRR nécessite une phase d'initialisation initiée lors du déploiement des nœuds. En résumé, les étapes d'exécution du protocole RCRR sont les suivantes :

a) Déploiement des nœuds

Tous les nœuds du réseau sont déployés aléatoirement mis à part quatre nœuds balises placés sur les extrémités nord, sud, est et ouest du réseau. Les deux nœuds balises nord et sud représentent les extrémités d'une ligne verticale traversant le réseau, alors que les deux autres nœuds est et ouest représentent les extrémités d'une ligne horizontale tel qu'illustré par la figure II.9 suivante :

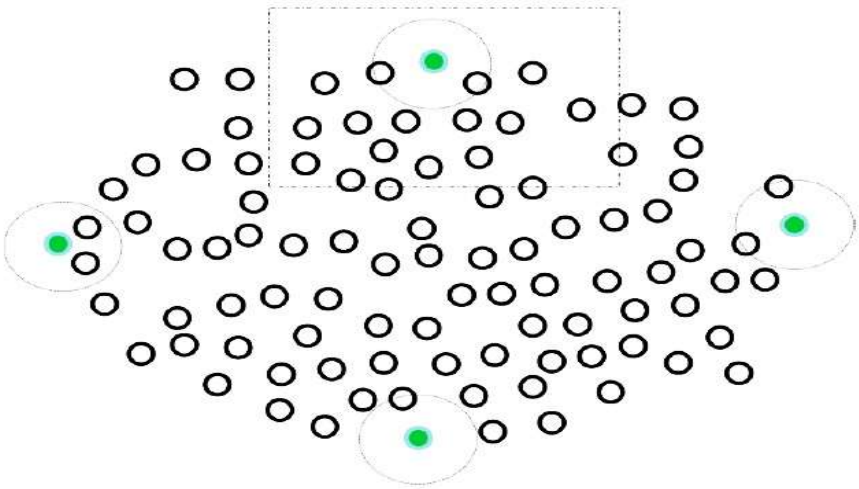


FIG II.9. Déploiement des nœuds dans RCRR.

b) Initialisation

Une fois tous les nœuds du réseau déployés, les deux nœuds balises du nord et du sud définissent leurs propres coordonnées à (x, 0) où x représente la coordonnée horizontale et 0 la coordonnée verticale et ceux de l'est et de l'ouest à (0, y) où 0 représente la coordonnée horizontale et y la coordonnée verticale. Par la suite, les nœuds balises procèdent à des diffusions dans le réseau de proches en proches afin d'établir le système de coordonnées relatives. Pour décrire en détail la procédure, on peut considérer le rectangle pointillé de la figure II.9, agrandi à la figure II.10. Le nœud balise (x, 0) diffuse à ses deux voisins directes ses coordonnées. Les deux voisins calculent alors leurs propres coordonnés en ajoutant 1 à la coordonnée verticale, ainsi de suite jusqu'à ce que chaque nœuds du réseau possède sa propre coordonnée relative. Notons, que lorsqu'un nœud reçoit un message avec une coordonnée verticale plus grande que la sienne, le message est simplement ignoré.

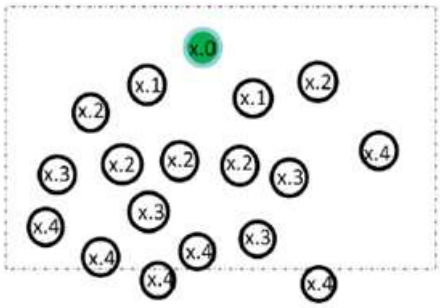


FIG II.10. La diffusion des coordonnées relatives lors de l'initialisation dans RCRR.

c) Détection et notification des événements

Lorsqu'un événement se produit dans le réseau, le nœud proche de l'événement va générer deux agents de notification. Les agents sans destinations spécifiques, devrons suivre respectivement le chemin de coordonnées croissant et le chemin de coordonnées décroissant, puis s'arrêter aux deux nœuds balises nord et sud. Après cette étape de notification, une ligne verticale de nœuds notifiés (en jaune) émerge tel qu'illustré par la figure II.11.

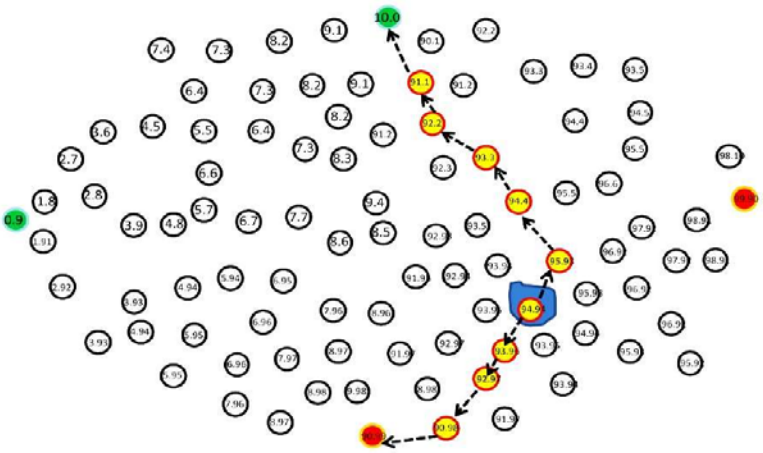


FIG II.11. La notification des agents dans RCRR.

d) Routage des requêtes et acheminement des données

Lorsqu'un nœud est à la recherche d'une donnée, il génère deux requêtes qui devront se déplacer dans les deux directions horizontales opposées. Dans la figure II.12, le nœud avec les coordonnées (5, 5) génère deux requêtes. Les chemins des requêtes suivent les coordonnées horizontales croissantes et décroissantes pour atteindre les deux nœuds balises horizontaux. Le requête se déplaçant dans la direction décroissante atteint le nœud balise gauche et s'arrête. L'autre, atteint le nœud notifié de coordonnées (92, 2) et s'arrête. Ainsi, le nœud notifié envoie une réponse au nœud source à travers le chemin inverse pour que ce dernier commence l'envoi des données. La ligne en pointillé (en rouge) de la figure II.12 est la route établie de la source au nœud puits de coordonnées (5.5).

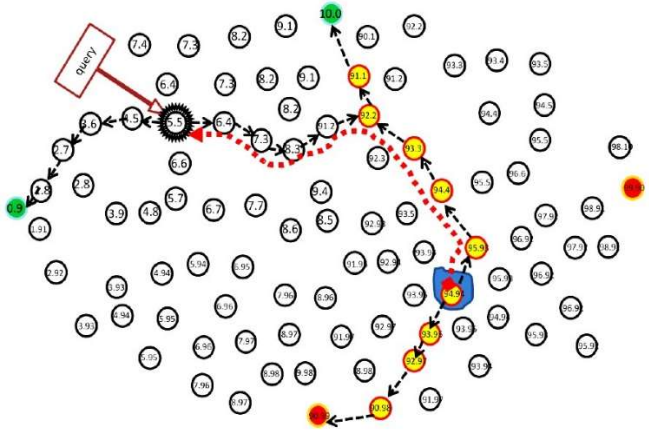


FIG II.12. Routage des requêtes et construction des chemins dans RCRR.

Discussion

RCRR est un protocole qui utilise le principe de coordonnées relatives afin d'optimiser les performances des protocoles de routage à marche aléatoire tel que RR. RCCR propose une approche qui garantit le croisement des chemins des agents et des requêtes. Néanmoins, la

qualité des chemins construits (en termes de nombres de sauts) reste un défi à soulever face à la contrainte de mise à l'échelle. Aussi, les résultats de simulation dans [60] ont montré que le nombre ainsi que la position des nœuds balises affectent considérablement les performances de ce protocole.

II.3.1.3 Stratégies basées sur le calcul géométrique

La stratégie de sélection du prochain saut de l'agent proposée dans cette sous classe, consiste à choisir la meilleure direction afin de construire des trajectoires aussi droites que possible. Si la trajectoire de l'agent se rapproche d'une ligne droite, moins de déviations sont créées sur le chemin et donc moins de transmissions sont générées. Ainsi, la quantité d'énergie sauvegardée sur plusieurs sauts peut être très significative. Les protocoles de routage de cette sous classe sont différents des protocoles de routage géographiques classiques où les coordonnées de tous les nœuds du réseau doivent être connues y compris celle de la destination. Dans ces protocoles, chaque nœud doit être en mesure de calculer la déviation de son voisin par rapport à une ligne droite, soit par le biais de coordonnées géographique soit en procédant à un calcul géométrique. Dans ce qui suit, nous allons présenter deux extensions du protocole *RR* appartenant à cette sous classe à savoir le protocole *Straight Line Routing (SLR) [46] [47]* et le protocole *Directional Rumor Routing (DRR) [45]*.

A. Le protocole SLR : Straight Line Routing for Wireless Sensor Networks (2005)

Le protocole Straight Line Routing (SLR) [46] [47] a été proposé afin de remédier à deux principales problématiques liées aux protocoles à marche aléatoire. La première, est la construction de chemin avec boucle (Spiral like path problem) à cause du choix aléatoire du prochain saut de l'agent et la deuxième, est la perte d'énergie et de bande passante enregistrée lors du transport et traitement des structures de données volumineuses qui mémorisent l'information de voisinage des nœuds. Pour ces raisons, SLR propose d'éliminer les structures de donnée et d'implémenter une stratégie de sélection du prochain saut de l'agent sur la base de calculs géométriques. Dans SLR, les auteurs supposent que tous les nœuds ont le même rayon de transmission R et que lorsqu'un nœud reçoit un signal donné, il est capable d'en mesurer la puissance ainsi que la distance.

L'idée principale du protocole SLR est illustrée par la figure II.13 où on suppose qu'un agent s'est déplacé depuis le nœud *B* vers le nœud *A*. SLR défini alors deux concepts : la bande extérieure d'un nœud *n* (***Outside Band)*** qui fait référence au cercle de rayon R depuis le nœud *n* et la bande intérieure d'un nœud *n* (***Inside Band)*** qui fait référence au cercle de rayon R/2 depuis le nœud *n*. Les nœuds candidats à être sélectionnés comme prochain saut sont ceux qui se trouvent dans l'intersection : ***Outside Band(A)*** ∩ ***Inside Band(B).***

L'algorithme du protocole SLR est défini selon les deux étapes suivantes: 1. la définition de la région candidate *(candidate region)* et 2. La sélection du prochain saut à partir de cette région. Lors de la première étape, le nœud courant (sur lequel se trouve l'agent) et le nœud précédent (à partir duquel l'agent a été transmis) renseignent deux variables : $flag_{in}$ et $flag_{out}$. Tous les nœuds qui appartiennent à la bande intérieure du nœud courant positionnent leur $flag_{in}$ à 1 et

tous les nœuds qui appartiennent à la bande extérieure du nœud précédent positionnent leur $flag_{out}$ à 1. Les nœuds de la région candidate sont ceux qui ont leurs deux variables $flag_{in}$ et $flag_{out}$ à 1. Lors de la deuxième étape, chaque nœud dans la région candidate active un timer (Twait) défini selon sa distance par rapport au nœud courant et précédent de telle sorte que le timer du nœud le plus proche de la ligne droite s'expire en premier. Ce nœud sera donc choisi comme prochain saut.

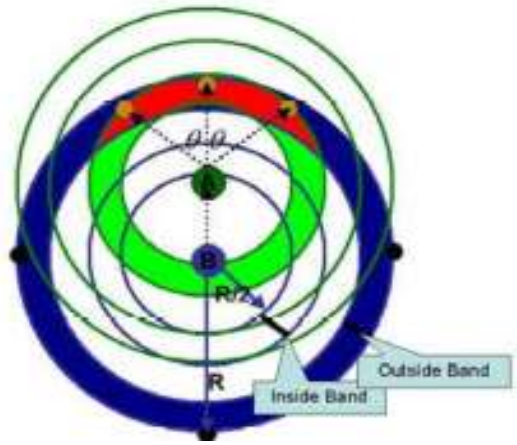


FIG II.13. Le concept graphique du protocole SLR, Un agent s'est déplacé depuis le nœud B vers le nœud A. R et R/2 sont les rayons de la région extérieure de B et la région intérieure de A respectivement.

Discussion

L'algorithme SLR a été amélioré pour faire face à de nombreuses situations et contraintes des réseaux de capteurs. Par exemple, dans les réseaux à faible densité, il y a une forte probabilité qu'aucun nœud n'appartienne à la région candidate, SLR propose alors d'y ajuster les largeurs de la bande intérieure et extérieure. Les résultats de simulation ont montré que le protocole SLR a réussi de diminuer le nombre de transmissions générées par les protocoles classiques (RR) ainsi que le nombre de saut des chemins établis surtout dans les réseaux à grande échelle. Par contre, l'absence de l'historique des chemins créés peut dans certains cas causé le problème de distribution uniforme de l'information.

B. Le protocole DRR : Directional Rumor Routing for Wireless Sensor Networks (2009)

Le protocole *Directional Rumor Routing (DRR)* [45] a été proposé pour les environnements disposant de systèmes de positionnement géographique. Son objectif est d'améliorer la latence des algorithmes classiques tels que *RR* en créant des chemins optimaux par le biais de routage géographique. L'algorithme *DRR* a connu des améliorations en proposant de remplacer les dispositifs d'information de localisation telle que le GPS (Global Positioning System) par des équipements moins chers et plus disponibles comme les antennes (AoA) « Angle of Arrival », les boussoles électroniques et les détecteurs d'intensité du signal. Ainsi, chaque nœud peut enregistrer la distance et la direction des nœuds voisins dans sa table de routage. Pour la sélection du prochain saut, l'agent calcule pour chaque voisin (i), la valeur:

$$D(i) = D(i-1) + d * \sin(a) \text{..... [45]}$$

Où '*d'* est la distance du voisin et *'a'* l'angle de déviation du nœud voisin. Le voisin ayant le plus petit écart est alors sélectionné comme prochain saut tel qu'illustré par la figure II.14.

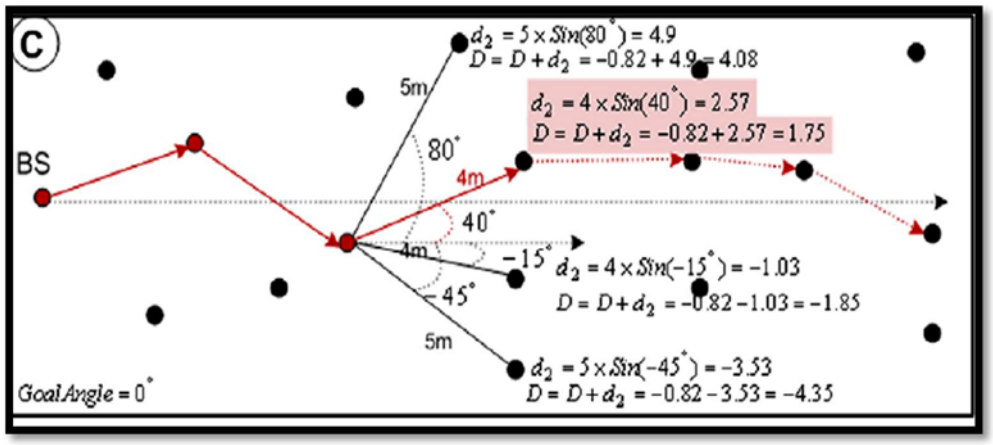


FIG II.14. Stratégie de sélection du prochain saut dans DRR sans GPS.

Discussion

Les résultats de simulations montrent que comparé au protocole RR, *DRR* construit des chemins optimaux en terme de nombre de sauts. Néanmoins, ces résultats ne sont maintenus que lorsque le taux d'erreur sur les angles et les distances est presque nul. *DRR* est un protocole qui traite le cas des vides dans le réseau, il est tolérant aux fautes puisque les tables de voisinage sont périodiquement mises à jour.

II.3.1.4 Stratégies basées sur le voisinage

Les protocoles de routage de cette sous classe utilisent l'information de voisinage des nœuds afin d'assurer des trajectoires droites pour les chemins des agents et des requêtes. L'information de voisinage transportée par chaque agent et requête est stockées dans des structures de données réinitialisées et mises à jour. Ainsi, le problème de débordement des structures de données est évité et les performances du réseau sont préservées face à la contrainte de mise à l'échelle. Les protocoles Fast Rumor Agents (FRA) [48-50] et Efficient Data Access based on Rumor Dissemination (EDARD) [51] sont deux exemples de ces protocoles. FRA et EDARD sont deux extensions directes du protocole ZRR dans lesquels nous essayons d'améliorer le taux de délivrance de requêtes ainsi que le délai de construction des chemins, leurs principes de fonctionnement sont présentés en détail dans le troisième chapitre de ce document.

II.3.2 Les protocoles de routage dépendants des systèmes de localisation géographique(Location based routing protocols)

Dans cette classe de protocoles de routage, les agents et les requêtes sont disséminés dans le réseau sur la base de l'information de localisation géographique des nœuds. Dans la version initiale du protocole Directional Rumor Routing (DRR) [45], les auteurs proposent d'implémenter une technique de routage en ligne droite. Pour cela, l'information de localisation des nœuds est nécessaire pour construire une ligne droite depuis le nœud source jusqu'au puits. Le prochain saut choisi, est le voisin ayant la plus petite déviation par rapport à cette ligne. Plus récemment, dans [54] [55], les auteurs proposent un protocole basé requête (query-driven based) dans lequel les chemins sont établis en utilisant le principe de la logique flous et des automates d'apprentissage ; les voisins sélectionnés sont ceux satisfaisants une certaine relation (énergie-distance). Bien que le protocole assure un équilibre entre consommation énergétique et équilibrage de charge, sont efficacité reste très dépendante des algorithmes de localisations afin de pouvoir estimer la destination de la requête à un certain temps *t*. Son utilisation est conseillée dans les réseaux denses avec un degré de voisinage des nœuds important. Dans le cas contraire, le protocole serait contraint d'avoir recours à la technique de diffusion sur tout le réseau et de ce fait, les chances de trouver des chemins optimaux à moindre coûts risquent de diminuer significativement.

D'autres protocoles de cette classe considèrent le facteur d'énergie comme métrique pour le choix du prochain saut comme dans [69], où un protocole basé les colonies de fourmis ACO (Ant Colony Optimisation) est proposé afin de construire des chemins efficaces en énergie. Les nœuds candidats sont ceux satisfaisant une certaine relation « énergie-distance-saut ». Aussi, dans QDVGDD [70] et MAQD [71], un nouveau mécanisme de dissémination de données avec puits mobile a été proposé. Contrairement aux protocoles basé évènement (event-driven based) tel que [35] [36], le protocole QDVGDD est basé requête, il exploite un réseau structuré en grille virtuelle. En dépit du coût généré par une telle structure, le trafic induit est réduit significativement mais le challenge réside dans le fait que la dimension du réseau et le nombre de nœuds doivent être connu au préalable. Le protocole MAQD [71], propose l'utilisation d'un système d'inférence basé sur la logique floue pour la sélection de chemins. Il considère quatre niveaux de connaissance : la durée de vie du capteur (énergie), le délai de la transmission de données, le coût énergétique du réseau et le plus court chemin de transmission. Dans [72], les auteurs proposent un protocole basé sur la distance euclidienne entre les nœuds qui améliore la durée de vie du réseau en sélectionnant les routes les plus optimales (en termes de nombre de sauts).

Notons que les protocoles de routage cités précédemment, n'assurent pas des points de croisement entre les chemins des agents et des requêtes. C'est pour cela, que nous nous intéressons dans cette section à une catégorie particulière des protocoles de routage utilisant l'information de localisation des nœuds. Dans cette catégorie, l'information de localisation est exploitée pour définir une région ou un ensemble de nœuds où les agents et les requêtes se rencontrent. Une telle approche permet de consommer moins d'énergie et de réduire les chemins

en nombre de sauts surtout lorsque la région de rendez-vous est située autour du centre du réseau. Cependant, l'efficacité de cette approche dépend fortement de l'utilisation de système de localisation géographique tel que le GPS et présente également l'inconvénient d'épuiser rapidement l'énergie des nœuds des régions de RDV. Dans ce qui suit, nous présentons quelques algorithmes qui se basent sur ce type de routage.

II.3.2.1 Le protocole ARR : Appointment-based Rumor Routing in Wireless Sensor Networks

Dans le protocole Appointment Rumor Routing (ARR) [57] [58], deux algorithmes de routage sont proposés : dans le premier, l'agent et la requête se rencontrent autour de la frontière du réseau alors que dans le second, ils se rencontrent au niveau du centre de gravité (COG) du réseau. L'exécution du protocole ARR nécessite une phase de calibrage du réseau dans laquelle les nœuds frontières sont supposés être correctement déterminés; leurs coordonnées géographiques vont aider pour définir le COG.

A. Le protocole APP-in-Edge (Appointment Rumor Routing in Edge of the network)

Dans *APP-in-Edge* [57], les agents et les requêtes se donnent rendez-vous sur le bord (la périphérie) de la topologie du réseau. Dans cet algorithme, lorsqu'un évènement se produit en un point aléatoire, un seul agent de notification est créé. Ce dernier, va se propager en direction de la frontière de la topologie et s'arrête lorsqu'il atteint un nœud frontière. Lorsque le nœud puits veut obtenir des informations sur l'évènement, il crée une requête qui se dirige aussi vers la frontière. Contrairement à l'agent, la requête parcours tous les nœuds frontières jusqu'à ce qu'elle croise l'agent en question. En ce point, elle suit le chemin de ce dernier pour atteindre la source. Concernant la rotation de l'agent autour des nœuds frontière, *APP-in-Edge* propose une trajectoire qui se base sur le calcul d'un angle de déviation α par rapport aux coordonnées du système polaire. Pour chaque voisin frontière, un angle βi est calculé et le prochain saut sélectionné est celui du voisin ayant la plus petite valeur $|\alpha - \beta i|$.

Même si l'approche de routage de l'algorithme *APP-in-Edge* consomme moins d'énergie que d'autres protocoles tel que DRR, elle présente l'inconvénient majeur de la longueur des chemins finaux (en terme de nombre de sauts) ainsi que l'épuisement rapide de l'énergie des nœuds frontières. Aussi, c'est une approche qui dépend étroitement de l'efficacité des techniques utilisées pour la détermination des nœuds frontière. Dans le cas d'une erreur de détection des nœuds frontières, l'agent peut dévié de sa trajectoire et épuisé son TTL en vain.

B. Le protocole APP-in-COG (Appointment in the Center Of Gravity)

En raison des inconvénients de l'approche de routage utilisée dans *APP-in-Edge*, cet algorithme a été révisé entièrement de telle sorte à ce que le nouveau point de rendez-vous soit fixé autour du centre de gravité du réseau. Cette nouvelle approche est appelée *APP-in-COG* [58]. Etant

donné les positions géographiques des nœuds du réseau, la position du point du centre de gravité (COG) se calcule selon la formule suivante:

$$(X_{cog}, Y_{cog}) = \left\{\frac{1}{n}\sum_{i=1}^{n} Xei\ ,\ \frac{1}{n}\sum_{i=1}^{n} Yei\right\} \text{..... } [58]$$

Où *n* est le nombre de nœuds frontières et (*Xei, Yei)* sont les positions du nœud frontière *i*. Lorsqu'un nœud détecte un évènement dans le réseau, il crée un agent et le transmet en direction du centre de gravité (COG) du réseau. Lors de son parcours, l'agent met à jour sa table de routage et les tables de routage des nœuds sur son chemin. Le processus de routage de la requête générée par le nœud puits est similaire à celui de l'agent et donc les deux paquets se donnent rendez-vous au niveau du COG du réseau.

Il convient de mentionner que l'inconvénient majeur de cet algorithme est la surconsommation énergétique au centre du réseau ainsi qu'un taux de perte de nœud important. Pour résoudre ce problème, les auteurs proposent de redéfinir périodiquement le COG (*Appointment in Distributed Center of Gravity)*. Cette idée améliore l'équilibrage de charge dans le réseau et rend le modèle de la consommation d'énergie semi-uniforme. Pour atteindre cet objectif, le COG du réseau est sélectionné selon une distribution normale aléatoire avec (numeral-COG) comme principal COG [73]. Un paramètre de déviation est déterminé à partir d'un générateur aléatoire afin de définir la répartition géographique du point COG dans le réseau. Lorsque le paramètre de déviation vaut zéro, le point de RDV sera le (numeral-COG). Si la valeur du paramètre de déviation est petite, le point COG sera sélectionné près du (numeral-COG) ce qui permet de créer des chemins optimaux, mais cela provoque une forte consommation d'énergie autour des points centraux du réseau. Par contre, la sélection d'une grande valeur pour le paramètre de déviation va permettre d'avoir le point du COG loin du centre du réseau. Dans cette situation, les nœuds vont perdre leur énergie de manière plus uniforme et auront un meilleur équilibrage de charge, mais les chemins créés seront plus longs en terme de nombre de sauts. Il est à noter également, que les deux algorithmes du protocole ARR ne traitent pas le cas des pannes des nœuds le long du chemin qui mène de la source vers le puits. La figure II.15 illustre le principe de fonctionnement de ces deux algorithmes.

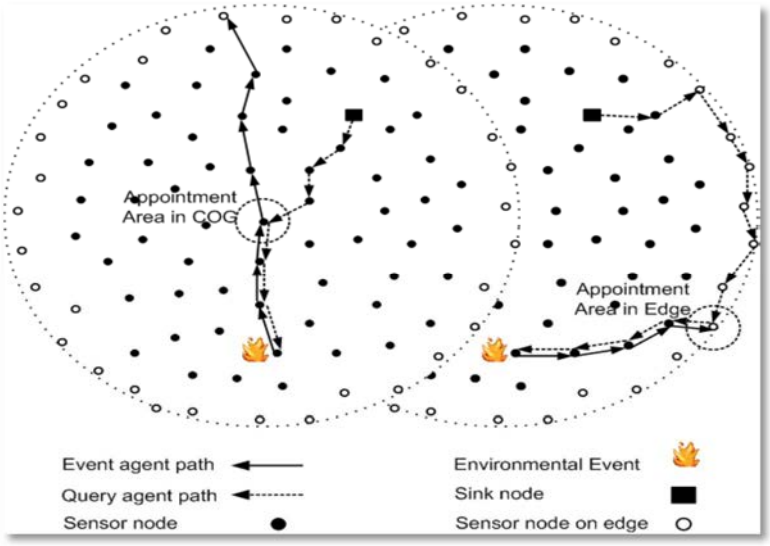


FIG II.15. L'algorithme *APP-in-Edge* (à droite) et l'algorithme *APP-in-COG* (à gauche).

II.3.2.2 Le protocole LBDD : A Line-Based Data Dissemination protocol for Wireless Sensor Networks

Le protocole LBDD [74] a été proposé pour les réseaux de capteurs avec des nœuds puits mobiles. Son objectif est de séparer la phase de collecte de celle de la dissémination de données. Les puits mobiles améliorent le temps de vie du réseau en évitant les transmissions excessives vers les nœuds proches des puits statiques. Ces derniers vont plutôt se déplacer et acquérir les données recherchées. LBDD définit une ligne virtuelle de largeur *w* qui divise le réseau en deux parties comme illustré par la figure II.16. Cette ligne est également divisée en groupes de taille *G*. Les nœuds à l'intérieur de la ligne sont appelés *inline-nodes* (nœuds internes), les autres sont appelés *ordinary nodes* (nœuds ordinaires). La ligne est une région de rencontre pour le stockage et la recherche de données. Le fonctionnement du protocole LBDD à recours à l'utilisation d'un système de localisation géographique tel que le GPS ou d'autres systèmes de coordonnées virtuelles pour obtenir les positions des nœuds dans le réseau. Les positions des nœuds frontières sont supposées aussi être déterminées. Le protocole LBDD fonctionne selon les deux étapes suivantes:

a) la dissémination de données : lorsqu' un nœud génère une nouvelle donnée, il l'achemine en direction de la ligne vers le plus proche *inline-node (fig II.16 (a))*.

b) la collecte de données : dans le but de retrouver une donnée spécifique, un puits envoie une requête vers la ligne dans un sens perpendiculaire *(fig II.16 (b))*; le premier *inline-node* qui reçoit la requête la propage dans les deux directions le long de la ligne jusqu'à atteindre le *inline-node* détenant la donnée. Cette dernière est ensuite transmise directement vers le puits *(fig II.16 (c))* en utilisant un routage géographique.

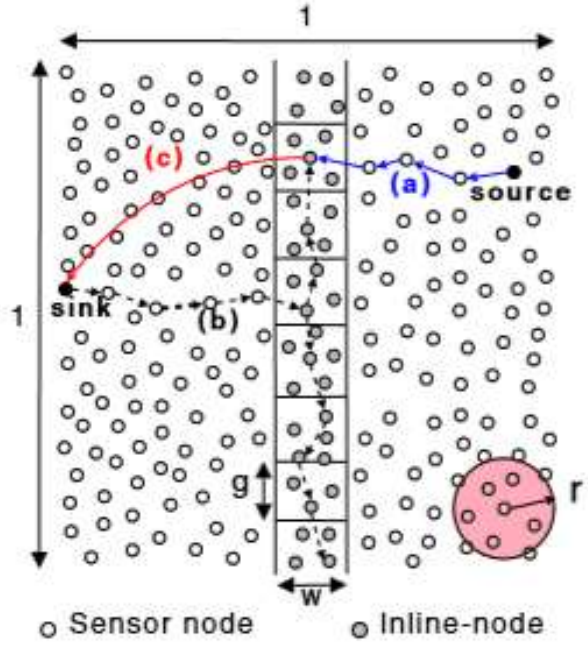


FIG II.16. Principe de fonctionnement du protocole LBDD.

Plusieurs optimisations du protocole LBDD sont possibles afin d'améliorer son efficacité: 1. dans les scénarios où le nombre de requêtes est supérieur au nombre d'évènements, LBDD avec réplication de données (LBDD-R) est proposé. La donnée est alors répliquée dans la ligne virtuelle entière afin de diminuer le coût de recherche. 2. dans les scénarios de recherche de cible *(target tracking),* une fois que le puits reçoit un rapport de donnée à partir d'un certain *inline-node*, la prochaine requête du puits sera envoyée à ce même *inline-node* au lieu de parcourir l'infrastructure virtuelle en entier. Cela permet la diminution de l'overhead et minimise la consommation d'énergie à l'intérieur de cette infrastructure.

Grace à son concept de ligne virtuelle, le protocole LBDD assure le traitement de n'importe quelle requête en minimisant le coût de communication en termes de trafic et en optimisant le temps de réponse. Cependant, c'est un protocole dont le fonctionnement dépend de l'utilisation de système de localisation souvent très couteux. Aussi, il présente l'inconvénient du problème de la persistance des données face à la panne et la sécurité des nœuds à l'intérieure de la ligne virtuelle.

II.3.2.3 Le protocole TTDD : A Two Tire Data Dissemination approach for Wireless Sensor Networks

Le principe de fonctionnement du protocole TTDD [35] [36] est de proposer une dissémination de données sur la base d'une structure de grille virtuelle construite suite à l'apparition d'un stimulus dans le réseau. Lors de la détection de ce dernier, au lieu d'attendre passivement les requêtes provenant des puits, la source construit de manière proactive une grille couvrant le réseau et met en place les informations d'acheminement au niveau des capteurs les plus proches des points de grille (appelés nœuds de diffusion). Avec une telle structure, une requête traverse

deux niveaux pour atteindre une source. Le niveau inférieur appelé cellules, et le niveau supérieur constitué des nœuds de diffusion. Ce processus se poursuit jusqu'à ce que le message atteigne la limite du réseau.
Sur la figure II.17, la source B divise le champ en une grille de cellules. Chaque cellule est un carré de dimension α. Le nœud puits S, va procéder à deux niveaux de dissémination afin d'atteindre la source B.

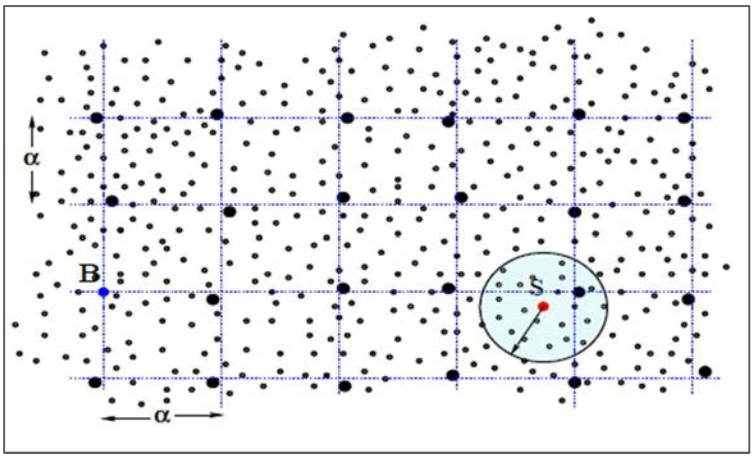


FIG II.17. Construction de la structure de grille dans TTDD.

Le principal avantage du protocole TTDD est qu'il s'adapte à la mobilité des puits dans les réseaux de capteurs à grande échelle. Cependant il enregistre une grande latence et une faible efficacité énergétique. Son principal inconvénient réside dans le coût énergétique nécessaire pour la construction et la maintenance de la structure de grille proposée.

II.3.2.4 Le protocole Railroad

RAILROAD [75] est un protocole de dissémination de données qui utilise le principe de « Rails » ou « chemin ferroviaire ». Sur la base de la forme topologique du réseau, il construit un seul Rail définit comme un intermédiaire de communication entre les nœuds, les observateurs, et les cibles. Pour ce faire, Le Rail devrait être placé dans une zone auquel chaque nœud peut facilement accéder. En utilisant le système Rail, Railroad met en œuvre une architecture de diffusion de données évolutive (face à la contrainte de mise à l'échelle) et économe en énergie dans des conditions dynamiques avec plusieurs cibles mobiles et des observateurs. Dans le système Railroad, lorsqu'une source génère un événement, les données détectées sont stockées localement et les métadonnées correspondantes sont transmises à la station la plus proche dans le Rail. Les récepteurs peuvent récupérer des informations en envoyant des requêtes. Cette dernière est transmise aux sources en trois phases : envoi de requête locale au Rail (2-1), interrogation de la circulation autour du Rail (2-2) et demande d'une notification aux sources (2-3) telle qu'illustré par la figure II.18.

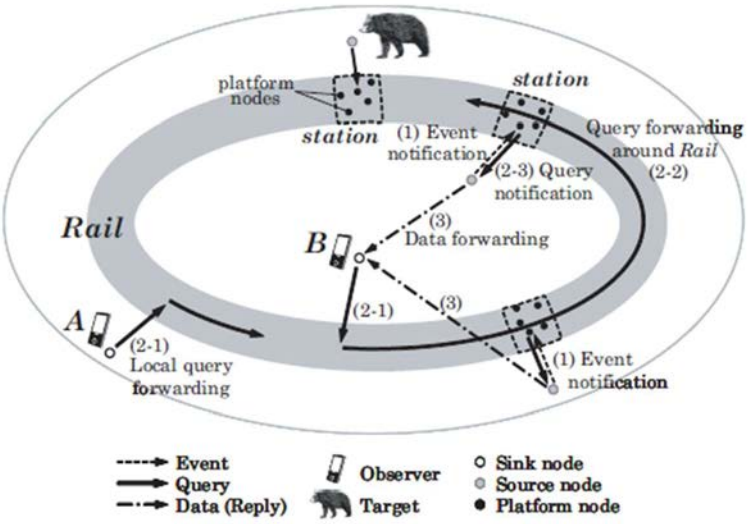


FIG II.18. Fonctionnement globale du protocole Railroad : Chaque nœud sur le chemin transfert la requête au nœud le plus proche du Rail. Une fois que la requête entre dans le Rail, elle est acheminée le long du Rail jusqu'à atteindre le point d'entrée. Ceci est possible en utilisant des champs de direction dans les listes de voisins des nœuds ferroviaires. La requête examine les stations pour trouver les données correspondantes pendant le voyage. S'il existe une station qui possède des données pertinentes, les nœuds de plate-forme génèrent un message de notification de requête et le transmettent au nœud source.

Le protocole Railroad assure les caractéristiques souhaitées pour les réseaux de capteurs sans fil tels que l'évolutivité et l'efficacité énergétique. Les résultats de simulations ont montrés que Railroad supporte les objets mobiles sans sacrifice de l'efficacité énergétique et de l'évolutivité.

Le tableau II.1 suivant résume les principales extensions du protocole RR présentées dans ce chapitre. La plus parts de ces protocoles tentent d'assurer une balance entre la qualité du chemin construit du puits vers le sink et la consommation d'énergie. En d'autres termes, ceux qui construisent des chemins optimaux peuvent consommer beaucoup d'énergie et ceux qui sont « energy-efficient », ne trouvent pas toujours des chemins optimaux et engendrent une grande latence. Le challenge des travaux présentés dans les prochains chapitres vise à assurer cette balance.

Tableau II.1. Principales extensions du protocole RR.

Protocole	Localisation des nœuds	Topologie	Stratégie de sélection du prochain saut de l'agent	mode de dissémination
FA [42]	Non	Plate	Aléatoire	Hybride
ZRR [43]	Non	Zones	Basée topologie	Hybride
CRR [53]	Non	Clusters	Basée topologie	Hybride
AlAc [44]	Non	Arbre	Basée topologie	Hybride
RCRR [60, 61]	Non	Coordonnés relatifs	Basée topologie	Hybride
SLR [46, 47]	Non	Plate	Calcul géométrique	Hybride

FRA [48-50]	Non	Zones	Voisinage local	Hybride
DRR [45]	Disponible	Plate	Basée ligne droite	Hybride
ARR-edge [57]	Disponible	Plate	Basée rendez-vous	Hybride
ARR-COG [58] [73]	Disponible	Plate	Basée rendez-vous	Hybride
RER [55]	Disponible	Plate	Energie-distance-saut	Basé requête
QDVGDD [70]	Disponible	Grille Virtuelle	Basée topologie	Basé requête
TTDD [35] [36]	Disponible	Grille Virtuelle	Basée topologie	Basé requête
MAQD [71]	Disponible (sink)	Plate	Basée logique floue	Basé évènement
RR with ACO [69]	Disponible	Plate	Basée colonies de fourmi	Hybride
Reliable and energy-aware protocol [72]	Disponible	Plate	Basée distance euclidienne	Basé requête
RAILROAD [75]	Disponible	Rail	Basée topologie	Hybride
LBDD [74]	Disponible	Line	Basée topologie	Hybride

II.4 CONCLUSION

Dans ce chapitre, nous avons introduit la classe des protocoles de routage à marche aléatoire dite : « Unicast agent based routing protocols ». Nous avons détaillé par la suite le protocole Rumor Routing connu comme étant un des premiers travaux exploitants la notion d'agents de rumeurs pour améliorer la dissémination de données dans les RCSF. Afin de mieux présenter toutes les extensions et variantes existantes du protocole RR, nous avons proposé une nouvelle classification des protocoles de cette classe sur la base de la disponibilité de l'information de localisation géographique des nœuds. A travers l'étude de l'existant, nous avons constaté que la plus parts des protocoles qui minimisent la consommation énergétique du réseau sont fortement dépendants des systèmes de positionnement géographiques et des algorithmes de localisation. D'autre part, certains protocoles ne font pas recours à ce genre de systèmes mais nécessitent beaucoup d'énergie pour la construction des topologies et structures requises pour les mécanismes de routage et de dissémination adoptés. De ce fait, développer des protocoles de dissémination et de routage pratiques sous des contraintes réelles reste un défi à relever par la recherche.

Dans les chapitres 3 et 4 suivants nous proposons quelques améliorations des travaux existants ainsi que de nouveaux protocoles conçus pour être appliqués dans des environnements réelles de réseaux de capteurs.

CHAPITRE III. PROTOCOLES POUR LA DISSÉMINATION DE RUMEURS DANS LES RCSF.

Dans ce chapitre, nous proposons deux protocoles de dissémination de données dans les RCSF. Dans le premier protocole : Fast Rumor Agent (FRA), nous proposons d'améliorer la trajectoire des agents et des requêtes des protocoles classiques tels que RR et ZRR. Dans une version plus optimisée, nous proposons le protocole Efficient Data Access based on Rumor Dissemination (EDARD) dans lequel nous implémentons une procédure de création d'agents fils (forking process) afin de mieux uniformiser la distribution de données dans le réseau. A la fin de ce chapitre, une série de tests de simulations est effectuée pour les protocoles proposés afin d'étudier leurs performances.

III.1 INTRODUCTION

Les protocoles à marches aléatoires (Unicast-agent based routing protocols) présentés dans le chapitre 2 précédent tentent de résoudre le problème de l'overhead généré dans le réseau en proposant un compromis logique entre la diffusion d'évènements et de requêtes. Néanmoins, leurs stratégies aléatoires implémentées lors du choix du prochain saut des agents présentent le problème de la distribution non uniforme de la donnée. Ceci est due principalement aux trajectoires des chemins crées en spirales ou en zigzag *(zigzag and spiral-like routing path problem)*. De ce fait, l'accès à la donnée devient difficile vue que cette dernière reste concentrée uniquement sur certaines régions du réseau.

Les travaux de *Banka et al (*2005) ont tenté de résoudre ce problème en proposant le protocole Zonal Rumor Routing (ZRR) [43]. ZRR exploitent une architecture topologique du réseau en clusters ou zones où chaque nœud capteurs est membre d'une seule zone. L'objectif d'une telle architecture est de disséminer les agents et les requêtes de zone en zone afin de leurs assurer une profonde progression dans le réseau. Ainsi, l'information est uniformément disséminée à travers le réseau et les chances de croisement entre les agents et les requêtes augmentent significativement.

Dans ce chapitre, nous commençons par proposer le protocole ***Fast Rumor Agents (FRA)*** [48-50] qui améliore les trajectoires des agents et des requêtes du protocole ZRR en proposant une technique de routage en ligne droite. Par la suite, et dans un souci de mieux uniformiser la distribution de l'information à travers le réseau, nous avons amélioré le protocole *FRA* en proposant un "forking process" implémenté lors de la dissémination de la rumeur dans le réseau. Le protocole résultant est nommée : ***Efficient Data Access based on Rumor Dissemination (EDARD)*** [51].

Dans le reste de ce chapitre nous décrivont le fonctionnement global des deux protocoles *FRA* et *EDARD* en présentant les hypothèses de travail, leurs principales procédures ainsi qu'une analyse de leurs performances faite à l'aide du simulateur NS2.34 [76].

III.2 PROTOCOLE FRA : FAST RUMOR AGENTS

L'approche de routage utilisée dans le protocole ZRR assure sans doute une meilleure dissémination de l'information comparé aux protocoles classiques à marche aléatoire tel que RR [41]. Par contre, à cause de l'information partielle de l'historique des agents ainsi que la taille limitée des structures de données utilisées, les chemins créés par les agents et les requêtes risquent parfois de revenir vers leurs régions initiales ou simplement de s'éloigner l'un de l'autre. L'objectif du protocole FRA [48-50] proposé et de réduire au maximum se risque en implémentant une technique de routage en ligne droite exploitant une information de voisinage à deux sauts contenue dans de nouvelles structures de données réinitialisées, ainsi qu'un historique complet des parcours des agents et des requêtes.

III.2.1 Environnement et hypothèses

Vu que les protocoles FRA et EDARD proposés sont des extensions du protocole ZRR [43], nous reprenons les principales hypothèses de ce dernier :

- Nous considérons un réseau de capteurs statique avec une distribution uniforme des nœuds capteurs;
- Nous supposons un environnement dépourvu de tout système de localisation géographique. De ce fait, les nœuds capteurs ne disposent d'aucune information de positionnement;
- Le réseau est partitionné en zones;
- Chaque nœud est exactement membre d'une seule zone;
- Les pannes des nœuds dans le réseau ne sont pas gérées;
- Les liens de communication sont fiables (pas de perte de paquets transmis).

III.2.2 Structures de données

Les principales structures de données utilisées dans le protocole ZRR [43] sont les suivantes :

- ***La liste des voisins :*** contient la paire {id nœud, id zone} de chaque nœud voisin.
- ***La liste des évènements (EvtList) :*** contient le triplet {évènement, direction, distance} de chaque évènement détecté. Cette liste permet aux nœuds ainsi qu'aux agents de garder une trace sur les évènements observés.

En plus des structures décrites ci-dessus, nous proposons dans FRA et EDARD les trois nouvelles structures suivantes:

- ***La liste de zones temporaires (TempZL):*** c'est une structure transportée par l'agent (resp. la requête), elle a pour rôle de garder l'historique des zones voisines de la dernière zone visitée afin d'éviter à l'agent (resp. la requête) les retours arrières. Elle est construite à partir de la liste des voisins du nœud émetteur à chaque fois que l'agent (resp. la requête) change de zone. La figure III.1 suivante explicite un exemple de cette structure.

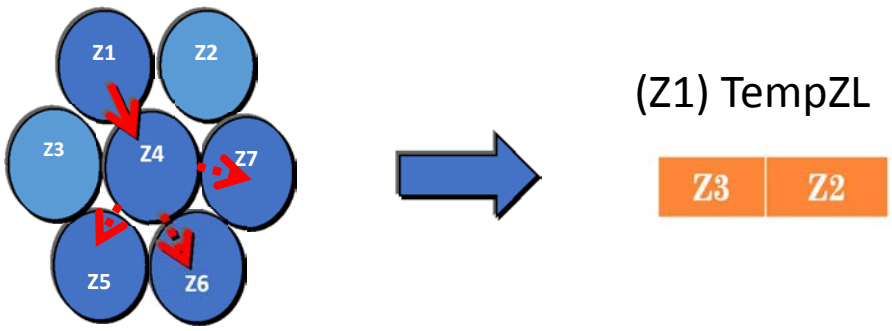


FIG III.1. La structure Temporary Zone List : sur l'exemple de cette figure un agent se trouvant en zone 4 et venant de la zone 1 va explicitement éviter lors de son prochain saut, les nœuds se trouvant dans les zones 2 et 3 vu qu'elles sont répertoriées comme zones voisines de la dernière zone visitée.

- ***La liste des nœuds temporaires (TempNL):*** c'est une structure transportée par l'agent (resp. la requête), elle a pour rôle de garder l'historique des nœuds voisins du dernier nœud visité par l'agent (resp. la requête). Elle est réinitialisée à chaque fois que l'agent (resp. la requête) change de zone. Cette liste sert à éviter d'envoyer l'agent (resp. la requête) vers des nœuds déjà visités et lui permet d'avancer toute en évitant les retours arrière à l'intérieur d'une même zone. La figure III.2 suivante explicite cette structure pour le nœud (N1). L'agent qui se trouve au niveau du nœud N4 ne pourra pas revenir en arrière via les nœuds N3 ou N2.

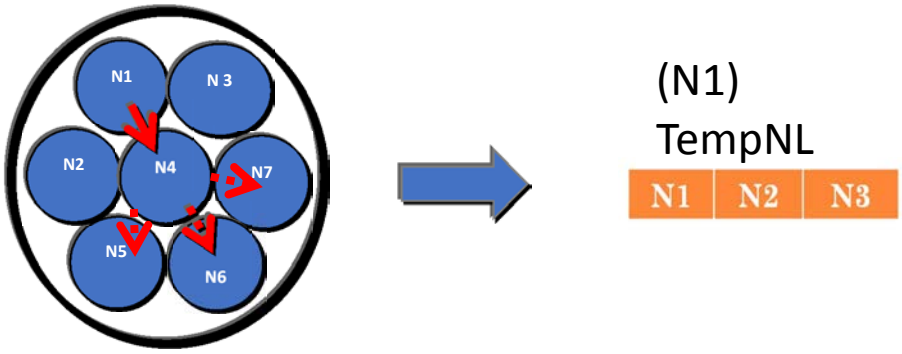


FIG III.2. La structure Temporary Node List.

- ***le vecteur historique de zones (HistoZ) :*** c'est un vecteur de bits de taille égale aux nombres de zones du réseau, initialisé à 0. A chaque fois que l'agent visite une zone, l'entrée correspondante à cette dernière dans le vecteur historique est mise à 1. Cette structure permet d'alléger la taille de l'agent (resp. la requête) et de lui donner une vue globale sur son historique afin de tracer un chemin sans boucle à travers le réseau. La figure III.3 suivante décrit cette structure pour un réseau partitionné en 10 zones dont les zones 2, 6 et 8 ont été déjà visitées par un agent.

Zone	1	2	3	4	5	6	7	8	9	10
Etat	0	1	0	0	0	1	0	1	0	0

FIG III.3. Le vecteur historique de zones.

Par ailleurs, la figure III.4 suivante décrit la structure des paquets AGENT et REQUETE.

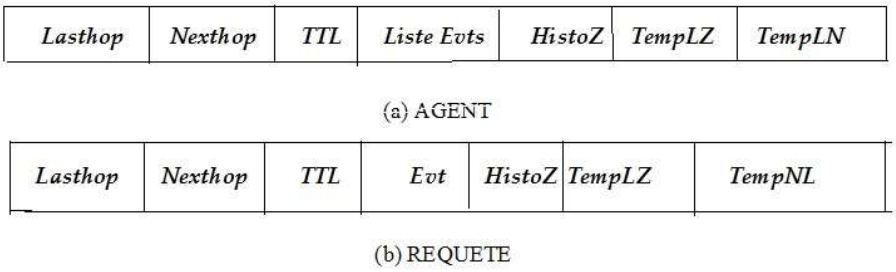


FIG III.4. La structure des paquets (a) AGENT et (b) REQUETE.

Avec,

- ***Lasthop :*** l'id du nœud émetteur de l'agent (resp. la requête).
- ***Nexthop :*** l'id du nœud destination de l'agent (resp. la requête).
- ***TTL :*** Time To Live, c'est la durée de vie de l'agent (resp. la requête) en nombre de sauts, décrémenté à chaque saut.
- ***EvtList :*** liste des évènements détectés.
- ***HistoZ :*** vecteur historique de zones.
- ***TempZL :*** la liste de zones temporaires.
- ***TempNL :*** la liste de nœuds temporaires
- ***Evt :*** l'évènement recherché par la requête.

III.2.3 Principe de fonctionnement

Le fonctionnement du protocole FRA se base sur l'exécution des deux principales procédures suivantes:

III.2.3.1 Création de zones

La création de zones dans le protocole FRA s'exécute comme suit :

- Chaque nœud à une probabilité d'être sélectionné comme « leader de la zone »; Ce dernier diffuse via un broadcast, l'annonce de création de sa zone à ses voisins via le message ***Msg_invitation (id-inv, idzone)*** ou ***id-inv*** est l'identificateur du nœud qui envoie l'invitation et ***idzone*** est l'identificateur de sa zone.

- Si le nœud récepteur appartient déjà à une autre zone, il répond par un broadcast contenant le message ***Reponse_invitation (id-rec, id-inv, idzone)*** ou ***id-rec*** est l'identificateur du nœud qui reçoit le message d'invitation et ***idzone*** est l'identificateur de sa zone. Sinon, il joindra la zone et invite à son tour ses voisins à la rejoindre. Dans les deux cas, les nœuds mettent à jours leurs listes des voisins respectifs.

- Les messages de création de zones vont se propager dans le réseau jusqu'à ce que les nœuds arrêtent de recevoir des invitations de nouvelles zones. Après un certain temps, le réseau devient stable, et chaque nœud devient exactement membre d'une seule zone.

III.2.3.2 Disséminations des agents et des requêtes

La dissémination des agents et des requêtes dans FRA se fait en trajectoire rectiligne sur la base d'un choix sélectif du prochain saut. L'information de voisinage est maintenue par l'envoi des messages « Hello » ou « beacons ». Dans ce qui suit, nous détaillons le processus de dissémination adopté par FRA :

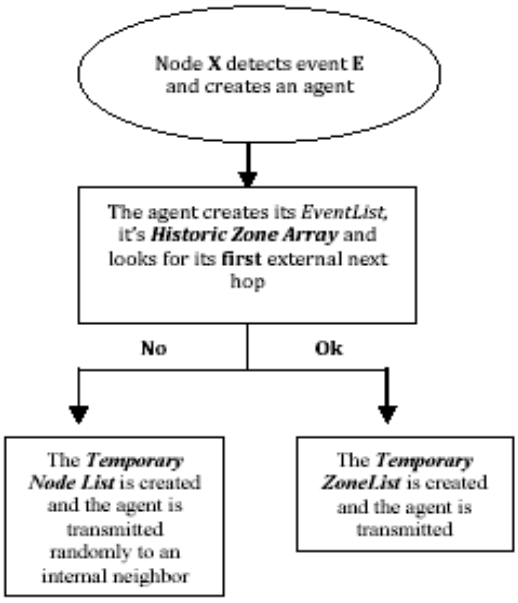
- Lorsqu'un nœud détecte un évènement, il génère de façon probabiliste un agent, qui prend en charge la propagation de cet évènement à travers le réseau.
- Le nœud détecteur de l'évènement (source) enregistre l'évènement dans sa liste d'évènements avec une distance égale à zéro.
- Cet agent, crée sa propre liste d'évènements afin de sauvegarder l'information sur l'évènement détecté. De plus, il crée le vecteur historique de zones et met le bit de l'entrée correspondante à la zone du nœud créateur à 1.
- Pour son premier saut, l'agent cherche un voisin externe (qui appartient à une zone différente que la sienne), si ce voisin n'existe pas, l'agent sera envoyé à un voisin interne différent de ceux qui existent dans la liste temporaire des nœuds.
- Si le destinataire de l'agent est un voisin externe, le nœud émetteur lui transfert la liste de zones temporaires. Dans le cas contraire, il lui transfert la liste de nœuds temporaires. Une fois que l'agent est reçu par un nœud, une synchronisation des listes d'évènements est effectuée.

Pour tous ses prochains sauts, l'agent essaye de trouver un voisin qui vérifie les deux conditions suivantes :

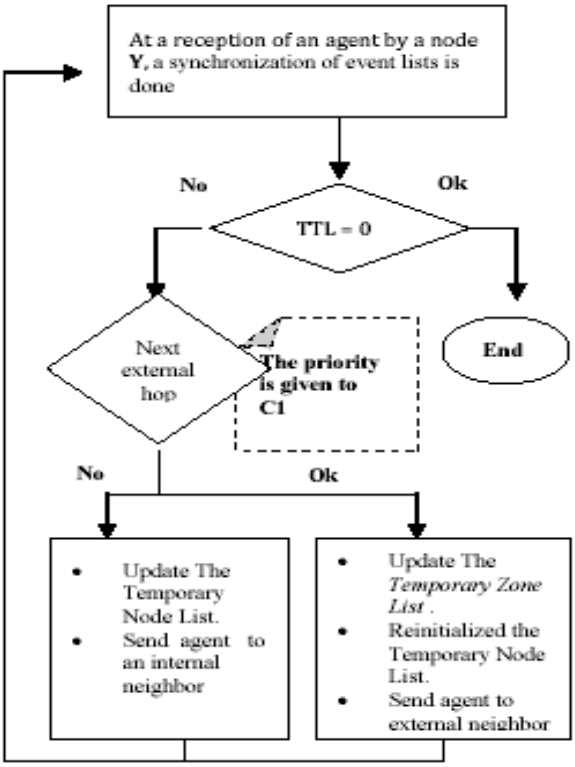
Condition1 (C1). Appartient à une zone non visitée.

Condition2 (C2). Appartient à une zone différente de celles contenues dans *TempZL* afin de maintenir une trajectoire aussi droite que possible.

- Notons, que la condition 1 est prioritaire. Au pire, le voisin peut appartenir à une zone qui existe dans *TempZL*. Si aucune des deux conditions n'est satisfaite, l'agent sera envoyé un à voisin interne. Durant tous les sauts internes à une zone, l'agent transporte les structures *TempZL* et *TempNL*. Mais dès que l'agent passe vers une nouvelle zone il garde uniquement le TempZL et réinitialise le TempNL. L'agent continu son parcours jusqu'à l'expiration de son TTL. La figure III.5 présente l'organigramme du choix sélectif des prochains sauts adopté par FRA (a) choix du premier saut (b) choix des prochains sauts.



- (a) premier saut.



- (b) prochains sauts.

FIG III.5. L'organigramme des choix sélectifs des sauts de l'agent dans FRA.

Le routage des requêtes est similaire à celui des agents excepté que la requête est à la recherche d'un évènement particulier. La dissémination de la requête s'arrête lorsqu'un chemin vers l'évènement recherché est trouvé ou bien lorsque son TTL expire. Dans ce dernier cas, la requête

peut soit être régénérée ou diffusée dans le réseau. Toutes les procédures du pseudo-algorithme du protocole FRA sont présentées au niveau de l'ANNEXE1.

Dans le but de mieux comprendre l'apport des structures de données utilisées dans FRA, la figure III.6 illustre le principe du choix des prochains sauts de l'agent dans les protocoles FRA et ZRR. Dans la figure III.6 (à gauche), un agent venu de la zone 1, se trouvant au niveau de la zone 6, procède au choix de son premier saut. Vu que les zones 2 et 5 appartiennent à sa liste TempZL, l'agent devra choisir un voisin externe des zones 9, 10 ou bien 7..... Pour tous ces prochains sauts l'agent évitera toujours d'aller vers les voisins de la dernière zone visitée afin de progresser aussi droit que possible dans le réseau. Par contre, dans la figure III.6 (à droite) même si l'agent évite les chemins en arrières, sa trajectoire divergent du chemin droit et devient plus longue (en nombre de sauts) ce qui risque d'augmenter le trafic et les chemins en boucle.

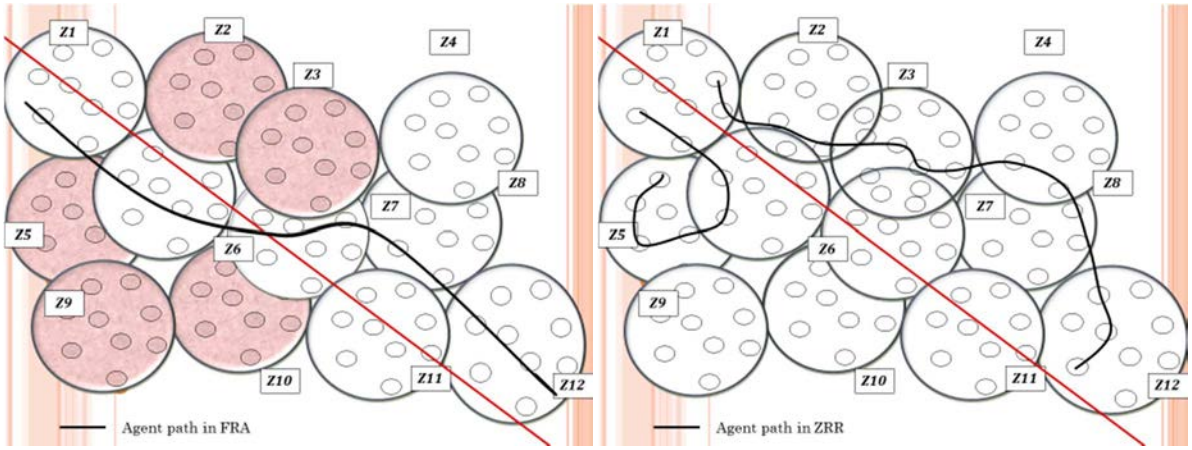


FIG III.6. Illustration du chemin des agents dans (à gauche) FRA et (à droite) ZRR.

III.3 PROTOCOLE EFFICIENT DATA ACESS BASED ON RUMOR DISSEMINATION (EDARD)

En considérant la contrainte de mise à l'échelle et afin d'uniformiser la dissémination de la donnée à travers le réseau. Nous avons amélioré le protocole FRA en proposant le protocole « Efficient Data Access based on Rumor Dissemination » (EDARD) [51]. En s'inspirant du protocole FA [42], mais en créant des chemins plus optimaux sans retour en arrière, nous proposons dans EDARD de créer pour chaque agent disséminé, un certain nombre d'agents fils. Ainsi, nous garantissons une optimisation dans le temps de réponse avec un plus grand nombre de nœuds informés. En plus des deux principales procédures du protocole FRA à savoir, la création de zones et la dissémination des agents et de requêtes, le protocole EDARD implémente la troisième procédure suivante :

III.3.1 Principe de Création d'agents fils (forking-proccess)

La création d'agents fils est faite selon le principe suivant :

- Suite à l'apparition d'un évènement dans le réseau, des agents maitres sont émis et pour chaque agent maitre reçu, un ACK (acknowledgment) est envoyé.
- Durant sa progression, chaque agent maitre donne naissance à des agents fils qui progressent de manière similaire et parallèle dans le réseau mais sans donner naissance à d'autres agents et sans générer des ACK lors de leurs transmissions.

En résumé, la transmission des agents maitres est assurée par l'envoi des ACK tandis que pour un agent fils l'évolution ne nécessite pas l'envoi d'ACK dans un souci de performances afin d'éviter un overhead dans le réseau. L'avantage de cette procédure est que les agents fils n'auront pas à être émis depuis la source ce qui va économiser le nombre de transmissions générées.

- Afin d'éviter une concentration des agents de rumeurs dans une seule région et de favoriser leur dispersion pour une plus grande couverture du réseau, les agents fils sont créés à un intervalle de *n* saut. Dans [42], une meilleure dissémination a été obtenue pour des instances de *n* supérieures à 1/4 des TTL des agents.
- Nous définissons le Time To Fork (TTF) comme étant le nombre de sauts qu'un agent maitre doit exécuter pour commencer la création des agents fils. Dans le but de déterminer la valeur du paramètre TTF, nous avons implémenté et testé dans la section III.4.2 plusieurs variantes du protocole EDARD dans lesquelles nous faisons varier le nombre d'agents fils ainsi que le TTF.

III.3.2 Le pseudo algorithme du protcole EDARD

Le tableau III.1 présente le pseudo-algorithme du routage des agents dans EDARD.

Tableau III.1. Pseudo-algorithme du routage des agents dans EDARD.

Algorithm III.1. EDARD agent routing scheme.
Input:
EvtList : {Evt, direction, distance};
TempZL:{ID zone1, … , ID zone i};
TempNL: {ID node1, … , ID node i};
At event detection:
1. Node n detects an event (Evt);
2. Node n creates an agent;

3. Node n and agent save Evt in their respective *EvtList* with (Evt.distance = 0);

4. IF (TTL ≠ TTF)

{agent proceeds to a selective next hop scheme according to TempZL and TempNL;

Agent is transferred and *EventList* synchronization is done;

agent updates/reinitialized *TempNL;}*

5. Forking process starts with child agent's creation;

Each child agent is routed according to path direction until the expiration of its TTL;

III.4 ANALYSE DES PERFORMANCES

Nous avons utilisé le simulateur de réseau NS2.34 [76] pour tester et analyser les protocoles FRA [48-50] et EDARD [51] proposés. Nous avons aussi implémenté le protocole ZRR [43] afin de le comparer à nos contributions. Dans cet objectif, nous avons exécuté plusieurs scénarios dont lesquels nous avons fait varier le nombre de requêtes ainsi que le nombre de nœuds :

- Afin de mieux mesurer les performances de nos contributions face à la contrainte de mise à l'échelle, nous avons exécuté des simulations en faisant varier le nombre de nœuds déployés dans le réseau (529, 625, 729, 841, 1024).
- Dans tous les scénarios, un nombre de requêtes (3, 10, 20, 30, 40, 50, 60, 70) est à la recherche de deux évènements dans le réseau.

Initialement, nous avons commencé par mesurer les performances du protocole FRA ainsi que sa comparaison avec le protocole ZRR [43] (section III.4.1). Par la suite, nous avons comparé notre première contribution (FRA) avec le protocole EDARD proposé et ZRR (section III.4.2).

III.4.1 Analyse des performances du protocole FRA

Pour mesure les performances du protocole FRA, nous avons considéré les paramètres suivants:

- La surface de déploiement des nœuds a été fixée à : 100x100 m^2
- La portée de captage d'un nœud capteur est de 5 mètres.
- Agent TTL: 100 sauts.
- Requête (Query) TTL : 125 sauts.
- La durée de simulation : 500 secondes.
- Le nombre de zones: 127 zones. Le nombre de zones du réseau est calculé selon la formule suivante :

$$Nz = N/Zo \text{ [43]} \qquad \text{(III.1)}$$

Où N est le nombre de nœuds du réseau et Zo est la taille optimale d'une zone, donnée par la formule (III.2):

$$Zo = \pi\, c2\, d \text{ [43]} \qquad \text{(III.2)}$$

Où « c » est le rang de communication du nœud et « d » est la densité du réseau.

Le tableau III.2, résume les principaux paramètres de simulation du protocole FRA.

Tableau III.2. Paramètres de simulation du protocole FRA.

Paramètre	**Valeur**
Taille du réseau	100X100 m^2
Nombre de nœuds	1024
Rang de transmission	5 m
Distribution des nœuds	Uniforme
Nombre de zones	127
Nbre de requête / évènement	[3 – 25]
Nbre d'évènements	2
Event-agent TTL	100 sauts
Query-agent TTL	125 sauts

III.4.1.1 Métriques

Dans ce qui suit, la liste des métriques utilisées:

• *Traffic de requête (Query traffic)*: le trafic généré par les requêtes est le nombre de toutes les transmissions ou réceptions de requêtes dans le réseau. Cette métrique est un bon indicateur sur la consommation énergétique du réseau.

• Taux de délivrance de requête (*Query delivery rate)*: une requête est considérée comme délivrée si elle atteint la source de l'évènement recherché. La mesure de cette métrique est très importante car elle nous renseigne sur l'efficacité de la stratégie de routage adoptée. Le taux de délivrance de requêtes est le rapport R_Q des requêtes correctement routées sur l'ensemble de celles générées dans le réseau pour un évènement particulier. Si Q_d est le nombre total des requêtes délivrées et Q_g est le nombre total des requêtes générées alors le taux $R_Q$est donné par la formule (III.3) suivante :

$$R_Q = Q_d/Q_g \qquad \text{(III.3)}$$

• Le temps d'établissement de chemin (la latence): c'est la différence entre le temps où la requête atteint le nœud notifié par l'évènement (croisement) et celui où elle fut générée.

III.4.1.2 Interprétation des résultats

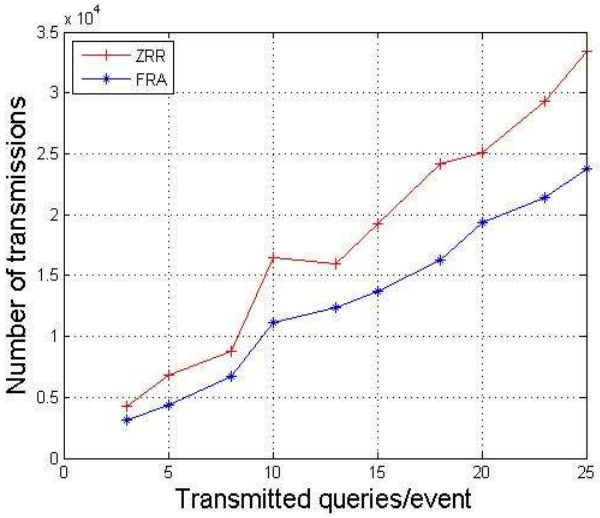


FIG III.7. Trafic de requêtes.

Dans la figure III.7, nous constatons que le trafic générées par les requêtes dans les deux protocoles FRA et ZRR converge jusqu'à 15 requêtes par évènement. Au-delà de cette valeur, les performances du protocole FRA améliore le trafic de 26%, ceci est dû à la stratégie de routage en ligne droite implémentée dans FRA qui permet aux agents ainsi qu'aux requêtes une propagation rapide et profonde dans le réseau avec des chemins sans boucles et sans retour arrière. Les résultats de la figure III.8, montrent que le nombre de requêtes transmises par évènement n'affecte pas de manière significative le taux de délivrance de requêtes, qui reste satisfaisant [90%-100%]. Par contre, nous remarquons que le taux du protocole FRA est légèrement inférieur à celui du protocole ZRR. Cette baisse peut être expliquée par le fait que la dispersion de la rumeur est plus optimale dans FRA, le nombre de nœuds informés tout au long du couloir de la progression de la rumeur est moins que celui dans ZRR. De ce fait, certaines requêtes risquent de ne pas croiser les évènements recherchés dans les délais requis.

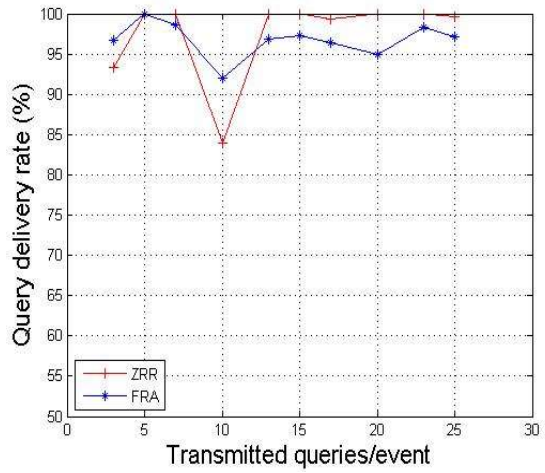


FIG III.8. Taux de délivrance de requêtes.

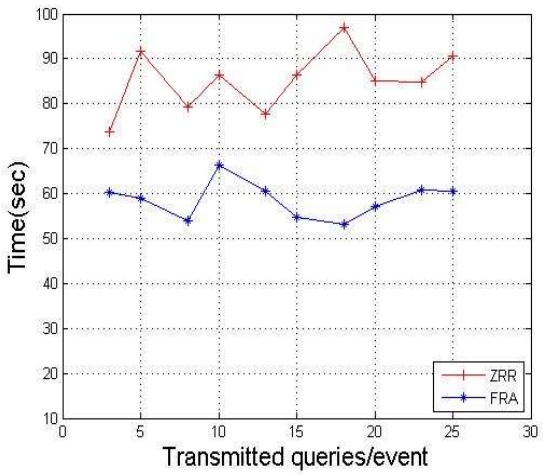


FIG III.9. Temps d'établissement de chemins.

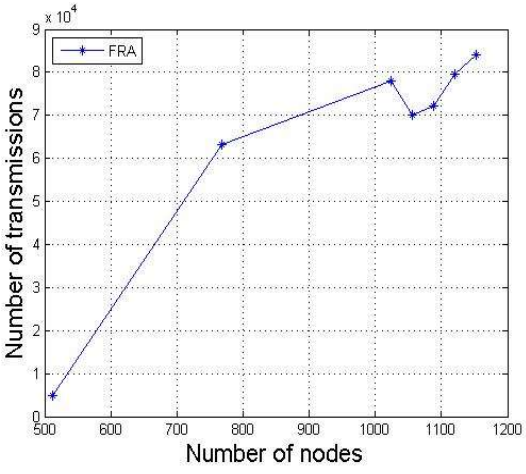
Les résultats de la figure III.9, montrent que le protocole FRA assure un meilleur temps d'établissement de chemins et donc une meilleur latence (jusqu'à moins de 30%). Ceci prouve l'efficacité et la rapidité de sa stratégie de propagation des agents et des requêtes qui permet d'établir des chemins dans de meilleurs délais.

III.4.1.3 Etude de la mise à l'echelle

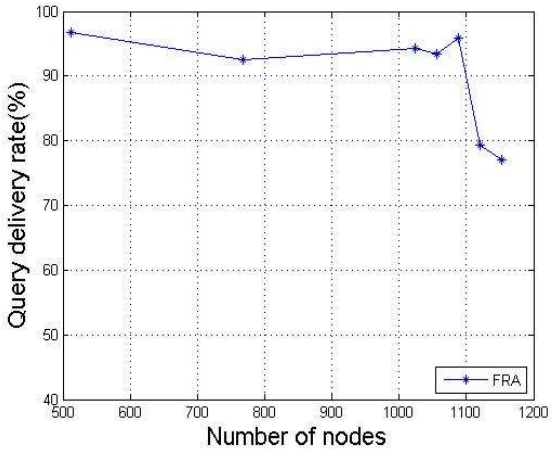
Pour l'étude de la mise à l'échelle du protocole FRA (scalability), nous avons fait varier le nombre de nœuds et la surface du réseau en maintenant une seule densité du réseau (0 .1024) tel qu'illustré par le tableau III.3.

Tableau III.3. Nombre de nœuds par surface.

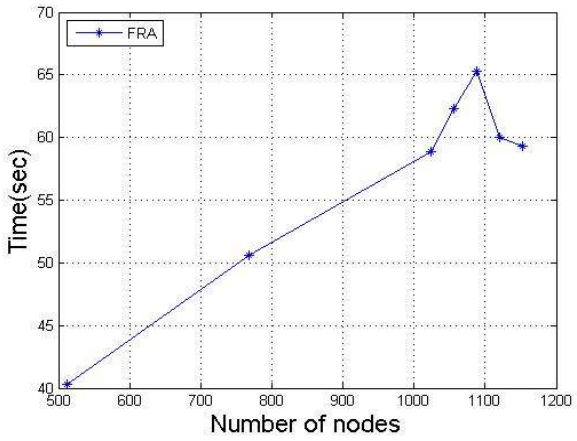
Nombre de nœuds	Surface du réseau
512	50*100
768	75*100
1024	100*100
1056	103.125*100
1088	106.25*100
1120	109*375
1152	112.5*100



(a) Trafic de requêtes



(b) Taux de délivrance de requêtes



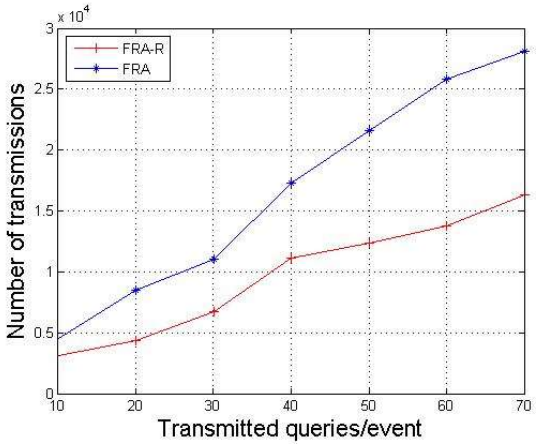
(c) Temps de réponse

FIG III.10. Etude de la mise à l'échelle dans FRA.

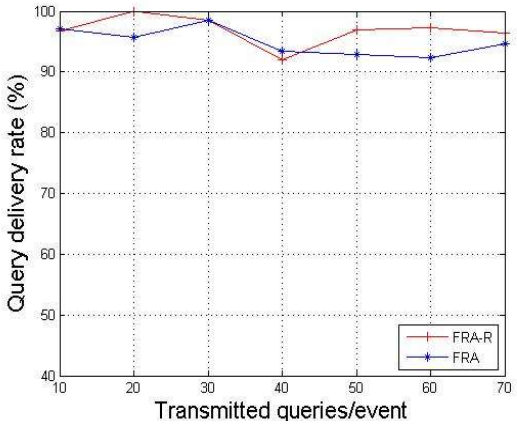
Les résultats de l'étude de la mise à l'échelle du protocole FRA illustré par la figure III.10 montrent qu'il maintient de bonnes performances jusqu'à1088 nœuds. Au-delà de ce nombre, le taux de délivrance de requêtes décroit jusqu'à 70% et le trafic atteint les 10000 messages. Ces résultats peuvent être expliqués par le fait que certains agents et requêtes se perdent dans le réseau surtout que leurs TTLs respectifs n'ont pas été adaptés à la nouvelle taille du réseau.

III.4.1.4 FRA avec réplicas

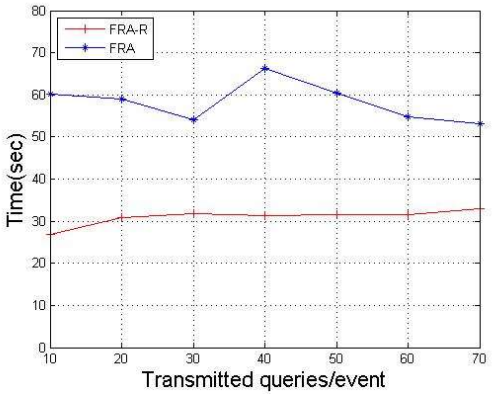
Pour les applications temps réels avec des données critiques, nous avons simulé une variante du protocole FRA nommée : FRA_R. Dans FRA_R, au lieu que l'agent de rumeur, transporte la métadonnée uniquement, il va transporter et propager toute la donnée le long de son chemin. Une fois à 50 % et 80 % de son parcours (TTL), l'agent va procéder à une réplication de données. De cette manière, nous permettons à la donnée de se rapprocher des sinks potentiels du réseau et présentons une alternative au problème de défaillance de chemins. Les résultats de la figure III.11 montrent que le protocole FRA_R améliore considérablement le taux de délivrance de requêtes et la latence (de 9.75 jusqu'à 32.85 secondes). Par contre, le processus de réplication implémenté va engendrer une surconsommation de la bande passante que nous comptons évaluer dans nos travaux futurs.



(a) Trafic de requêtes



(b) Taux de délivrance de requêtes



(c) Temps de réponse

FIG III.11. Les performances du protocole FRA_R.

III.4.2 Analyse des performances du protocole EDARD

Pour mesure les performances du protocole EDARD, nous avons considéré les même paramètres de simulations que celui de la version initiale FRA. Par contre, nous avons augmenté le nombre de requêtes générées dans le réseau [10- 70] et vu que le protocole EDARD implémente une procédure supplémentaire de création d'agents fils (forking process), nous avons fixé le TTL de l'agent fils à 50 sauts. Le tableau III.4 illustre les nouveaux paramètres de simulations du protocole EDARD.

Tableau III.4. Les paramètres de simulation du protocole EDARD.

Paramètre	**Valeur**
Nombre de requêtes	[10-70]
TTL agents fils	50 sauts
TTF	50, 80

III.4.2.1 Evaluation des variantes du protcoles EDARD

Afin de mieux estimer le meilleur temps pour l'exécution de la procédure de création d'agents fils, nous avons évalué les variantes suivantes du protocole EDARD où le Time To Fork (TTF) définie le nombre de sauts nécessaires traversés par l'agent maitre avant la création d'agents fils :

- EDARD1: deux agents fils à TTF = 50 (à 50 % du parcours de l'agent maitre);
- EDARD2: deux agents fils à TTF = 80 (à 80% du parcours de l'agent maitre) ;
- EDARD3: deux agents fils à TTF = 50 et deux autres à TTF = 80;
- EDARD4: un agent fils à TTF = 50 et un autre à TTF = 80.

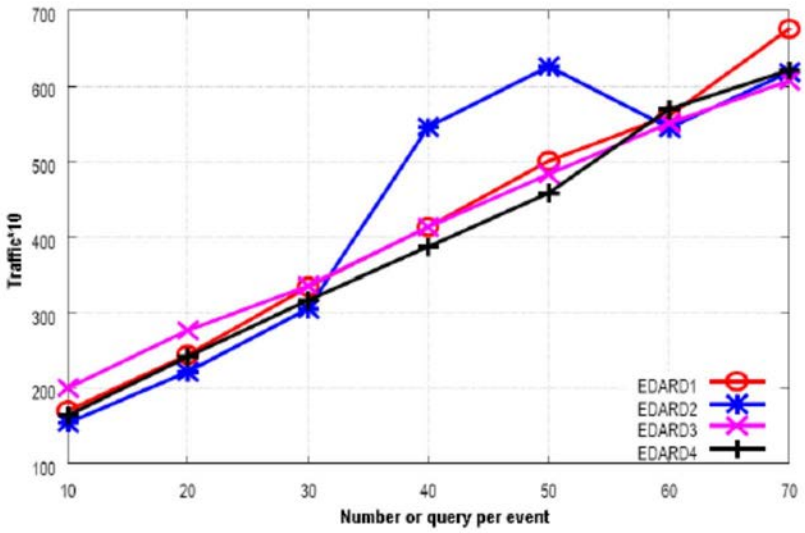


FIG III.12. Trafic de requêtes.

La figure III.12 illustre le trafic généré par les quatre variantes du protocole EDARD. Les résultats montrent que le trafic augmente avec l'augmentation du nombre de requêtes générées. Cependant, pour EDARD2, le trafic enregistre un pic entre 30 et 60 requêtes. Ceci peut être expliqué comme suit : vu que la création d'agents fils dans EDARD2 est retardée à 80 % du parcours de l'agent maitre, le nombre de requêtes non délivrées risque d'augmenter ce qui augmente à son tour le nombre de transmissions générées dans le réseau.

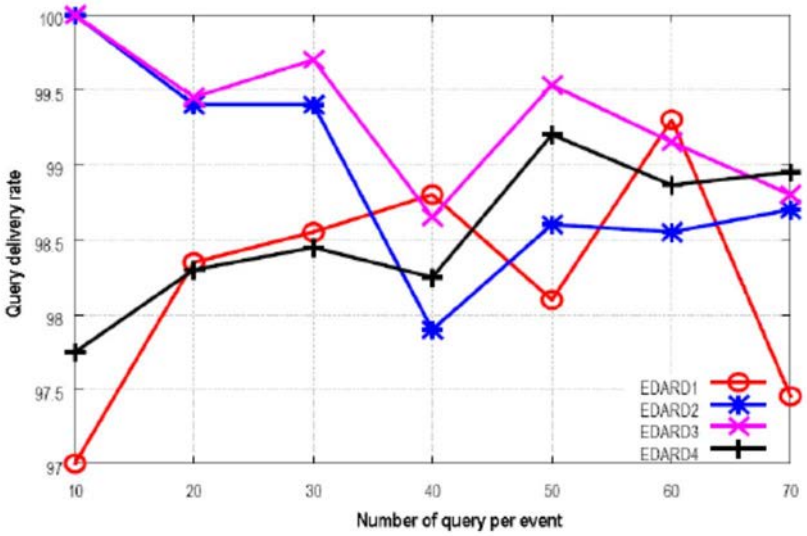


FIG III.13. Taux de délivrance de requêtes.

La figure III.13 illustre le taux de délivrance de requêtes. Deux évènements sont générés dans le réseau et 10 à 70 requêtes sont à leurs recherches. Les résultats montrent que toutes les variantes d'EDARD enregistrent un taux satisfaisant entre 97 et 100%. Notons que pour EDARD1 et EDARD4, le taux de délivrance -en dessous des 40 requêtes- est légèrement inférieur à celui d'EDARD3 et EDARD4. Pour EDARD2, le taux de délivrance chute de 99.5 à 98% après 30 requêtes ce qui rend les performances d'EDARD3 nettement meilleurs en comparaison aux autres variantes grâce à son double forking process avec plus d'agents fils.

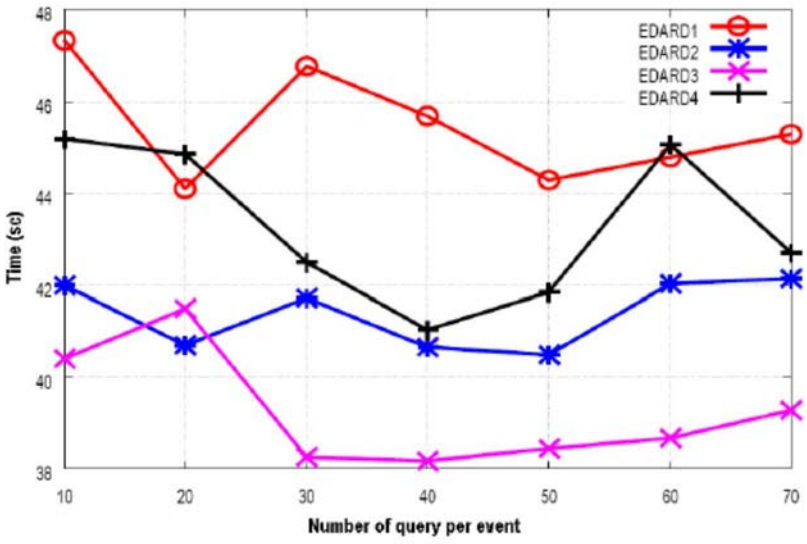


FIG III.14. Le temps d'établissement de chemin.

La figure III.14 illustre la latence des quatre variantes du protocole EDARD. A partir de ces résultats ainsi que ceux des figures III.12 et III.13, nous concluons qu'EDARD3 enregistre les

meilleures performances en terme de trafic, de taux de délivrance de requêtes et de temps d'établissement de chemins. Grace à son double forking process, il permet une meilleure distribution et un accès efficace à la donnée en minimisant le trafic généré.

III.4.2.2 Comparaison des résultats

Dans ce qui suit, nous allons interpréter les résultats de comparaison de la meilleur variante du protocole EDARD à savoir EDARD3 avec le protocole FRA et le protocole ZRR.

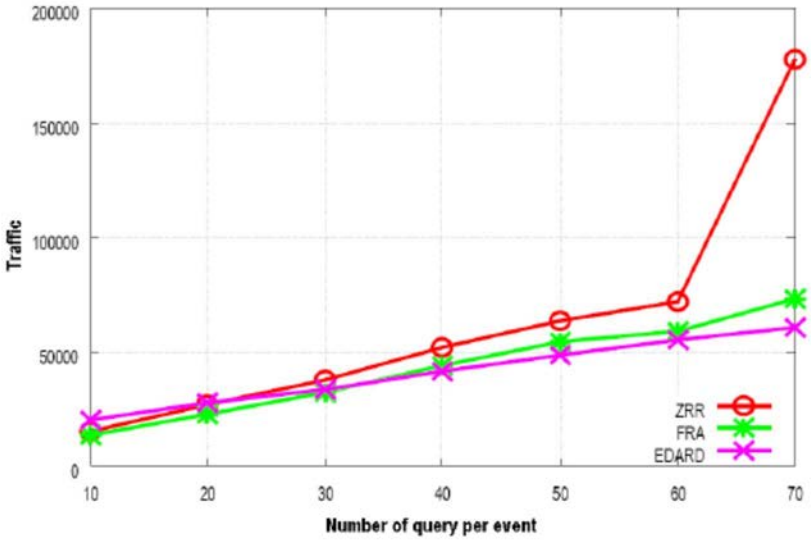


FIG III.15. Comparaison du trafic de requêtes.

Les résultats de la figure III.15 montrent que le nombre de requêtes n'affecte pas les performances des protocoles FRA et EDARD. Par contre, le trafic généré par le protocole ZRR augmente significativement après 60 requêtes ceci est probablement dû au trafic causé par les requêtes perdues dans le réseau. Les résultats montrent l'efficacité de la stratégie de routage du protocole EDARD car en dépit du nombre d'agents générés, le trafic reste satisfaisant, la transmissions de plus de 12000 messages est sauvegardée en comparaison avec le protocole FRA.

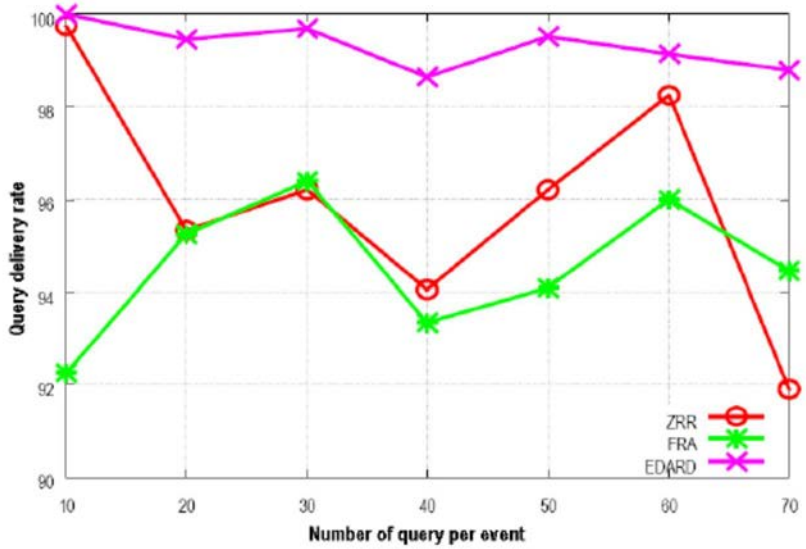


FIG III.16. Comparaison du taux de délivrance de requêtes.

Les résultats de la figure III.16 montrent que contrairement aux protocoles FRA et ZRR, le nombre de requêtes n'affecte pas les performances du protocole EDARD. Ce dernier améliore le taux de délivrance de requêtes de 7.75% comparé aux autres protocoles grâce à sa stratégie de routage et surtout à sa meilleure distribution de la rumeur qui atteint différentes régions du réseau.

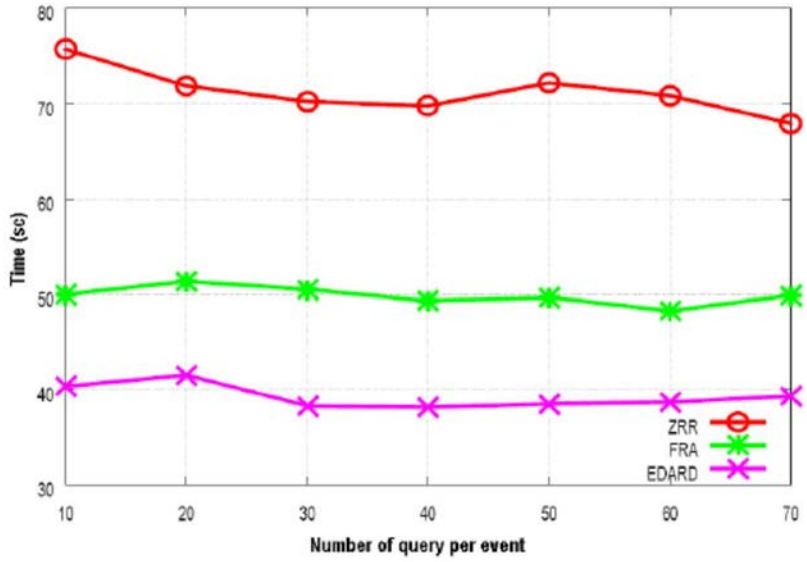


FIG III.17. Comparaison du temps d'établissement de chemin.

Les résultats de la figure III.17 montrent que le protocole EDARD améliore la latence de 35% en comparaison avec le protocole ZRR et de 12% en comparaison avec le protocole FRA.

III.5 CONCLUSION

Nous avons présenté dans ce chapitre deux protocoles de dissémination de données à base d'agents de rumeurs dans les RCSF. Dans la version initiale : *Fast Rumor Agent (FRA)*, nous avons amélioré la trajectoire des agents et des requêtes en implémentant un principe de routage en ligne droite grâce au maintien d'une information de voisinage dans des structures de données réinitialisables. Par la suite, dans un souci de mieux uniformiser la distribution de la données à travers le réseau, nous avons proposé le protocole *Efficient Data Access based on Rumor Dissemination (EDARD)*. EDARD implémente une procédure supplémentaire de création d'agent fils (forking process) afin de couvrir plus de régions dans le réseau.

Les résultats de simulations ont montrés que FRA et EDARD donnent de bonnes performances. De manière générale, le protocole EDARD est globalement meilleur car il offre une meilleure latence avec un trafic réduit. Pour les applications temps réelles avec des données critiques, nous avons testé une variante du protocole FRA avec réplication de données (*FRA_R*). Les résultats sont satisfaisants et montrent que la réplication permet de réduire la latence et présente une alternative au problème de défaillance de chemin. Dans cette optique, nous comptons l'implémenté dans le futur pour le protocole EDARD.

Dans le prochain chapitre, nous aborderons deux autres protocoles de dissémination de données conçus pour des environnements de RCSF plus proche de la réalité. Par exemple, nous considérons des RCSF avec un déploiement aléatoire des nœuds et des situations où l'information de localisation des nœuds peut être indisponible à cause du coût ou du manque de précision des systèmes de géolocalisation utilisés.

CHAPITRE IV. PROTOCOLES D'OPTIMISATION D'ACCÈS À LA DONNÉE.

Afin d'assurer un accès optimal à la donnée en termes d'énergie et de latence dans des déploiements réelles des RCSF, nous proposons dans ce chapitre deux nouveaux protocoles de dissémination de données à base d'agents de rumeurs. Le premier protocole Corridor Star Routing (CSR), se base sur l'information de localisation des nœuds pour assurer une propagation efficace et uniforme de la donnée à travers le réseau. Ainsi, des chemins pour le routage des requêtes seront construits avec un moindre trafic ce qui convient aux applications temps réelles (non-delay tolerant applications).
Le second protocole Backbone Web Routing (BWR), adopte une approche de routage à base de points de rendez-vous. Il est conçu pour les environnements dépourvus de systèmes de localisation géographique. Dans BWR, un backbone sous forme de toile (Web) est utilisé pour router les évènements et les requêtes vers une région centrale de rencontre. La toile proposée est simple avec un coût minimal de construction et de maintenance. Une étude de la complexité de l'algorithme de construction de la toile ainsi qu'une analyse de performances des protocoles proposés conclue ce chapitre.

IV.1 INTRODUCTION

Rumor Routing (RR) [41] est l'un des premiers protocoles de routage utilisant la dissémination d'agents de rumeurs dans les RCSF. Toutes les extensions de ce dernier proposées dans la littérature [42-61] [69-75] ont pour objectif de réduire le trafic, d'assurer un bon taux de délivrance de requêtes et un meilleur temps d'accès à la donnée. Parmi les protocoles proposés, certains [55] [57-58] [65] [69-75] (location-based protocols), améliorent significativement les métriques citées précédemment mais sont fortement dépendants d'un système de localisation géographique tel qu'un GPS (Global Positioning System). Parfois, ces protocoles nécessitent l'utilisation des algorithmes et techniques de localisation très gourmands en énergie. D'autres [43-44] [53] [60-61] (free-location based protocols) n'utilisent ni les dispositifs, ni les algorithmes de géolocalisation, mais consomment une importante quantité d'énergie lors de l'initialisation topologique du réseau.

Dans ce chapitre, nous proposons deux nouveaux protocoles de dissémination d'agents de rumeurs dans les réseaux de capteurs afin d'assurer un accès efficace à la donnée en optimisant les facteurs temps de réponse et énergie. Le premier protocole, Corridor Star Routing (CSR) [52] nécessite la connaissance des positions géographiques des nœuds. Une fois que la position des nœuds du réseau est acquise, ces derniers peuvent éteindre leurs dispositifs de localisation pour une utilisation optimale de l'énergie. Dans CSR, suite à l'apparition d'un évènement dans le réseau, quatre agents de rumeurs sont envoyés selon les principales directions de la boussole (Nord, est, ouest et sud). Au fur et à mesure de la progression des agents, des couloirs sont construits afin de choisir des sauts qui permettent à l'agent de maintenir une trajectoire droite vers sa direction. Chaque couloir construit, est caractérisé par sa longueur L et sa largeur W déterminés selon le rang de transmission du nœud émetteur et la densité de la région. Lorsqu'un nœud sink cherche à obtenir des données sur un évènement particulier, il lance quatre requêtes (queries) qui progresseront de manière similaire aux agents jusqu'à ce que leurs chemins se croisent.

Pour certains déploiements réels des RCSF où la localisation des nœuds ne peut être correctement obtenue (soit à cause du coût ou de manque de précisions), nous proposons le protocole Backbone Web Routing (BWR) [52]. BWR est une approche de routage à base de point de rendez-vous (RDV). Il consiste en la construction d'un backbone sous forme d'une toile (Web) qui a pour rôle de router les évènements (*events)* et les requêtes (*queries)* vers une région centrale de RDV. La construction du backbone proposée, se fait selon les principales étapes suivantes: 1.La détermination du centre virtuel du réseau (Virtual Center node); 2. L'identification de la région centrale de RDV ainsi que 3. Sa maintenance. Toute la difficulté de l'approche ainsi proposée réside dans le fait que l'agent doit construire des chemins aussi droits que possible avec le minimum de déviations et évitant les retours arrière en l'absence d'une information de localisation des nœuds. Pour ce faire, l'agent implémente pour son routage une technique sélective du prochain saut similaire à celle des protocoles FRA [48-50] et EDARD [51] proposés dans le chapitre précédent. Le reste de ce chapitre détaille la conception des deux protocoles CSR et BWR ainsi qu'une étude de leurs performances par le biais de simulation. Enfin, une discussion sur leurs éventuelles améliorations conclue ce chapitre.

IV.2 CORIDOR STAR ROUTING (CSR)

Bien que les protocoles de routages à base d'agents de rumeurs (Unicast agent based routing protocols) offrent un meilleur temps de réponse et réduisent la redondance des données, ils présentent dans certains cas des faiblesses. Un de leurs inconvénients, est qu'ils disséminent la donnée vers une seule région du réseau. En conséquence, le reste des nœuds du réseau n'ont pas connaissance de la donnée disséminée. Aussi, dans certains scénarios où la distribution des nœuds est aléatoire, la sélection du prochain saut de l'agent devient impossible à cause des vides crées dans le réseau. Dans la littérature, certains travaux ont tenté de mieux uniformiser la donnée. Par exemple, dans les travaux de *Shokrzadeh. H et al* (2009) [45], les auteurs proposent un routage en étoile avec des trajectoires droites pour les agents et les requêtes. Pour la sélection des prochains sauts, les auteurs proposent de choisir le nœud voisin possédant le plus petit angle de déviation de la ligne droite tracée depuis le nœud source vers une destination précise. Le challenge d'un tel protocole réside dans le fait de connaitre au préalable la vision topologique du réseau ainsi que les nœuds frontières. Souvent, dans les déploiements réels des réseaux de capteurs, la connaissance préalable de la topologie du réseau est irréelle à cause de l'information de voisinage limitée d'un nœud dans le réseau.
Récemment, dans les travaux d'*Ahvar. I et al* [54] (2012) [55] (2016), les auteurs proposent un protocole de routage dit : query-driven based dans lequel les chemins sont construits en utilisant la logique floue, les voisins sélectionnés comme prochain saut, sont ceux satisfaisant une certaine relation (énergie, distance, saut). Tandis que le protocole proposé assure une efficacité en énergie et un équilibrage de charge dans le réseau, il dépend fortement de l'utilisation des systèmes de localisation géographique afin de pouvoir estimer la localisation de la destination à un instant t donnée. Aussi, le protocole à de meilleures performances dans les réseaux avec un degré important de voisinage des nœuds. Dans le cas contraire, il est contraint d'avoir recours à la contraignante technique de diffusion à travers tout le réseau et les chances de trouver des chemins optimaux vont nettement diminuées.

En proposant le protocole CSR, nous essayons de résoudre les problèmes cités ci-dessus. Dans le but d'uniformiser la dissémination de la donnée sur le réseau, nous proposons de lancer quatre agents de rumeurs selon les quatre principales directions géographiques (Nord, Sud, Est et Ouest). Les requêtes sont lancées de manière similaire afin d'augmenter les chances de croisement. Tel qu'illustré par la Figure IV.1, le concept global de l'approche du routage proposée dans CSR est similaire à celle du protocole TTDD proposé par *Luo. H et al* [36] (2005). Par contre, dans CSR, la donnée est disséminée sur une topologie plate, de manière homogène et plus uniforme sans la contrainte d'une surconsommation énergétique due à la construction de gille virtuelle à chaque détection d'un stimulus dans le réseau.

.

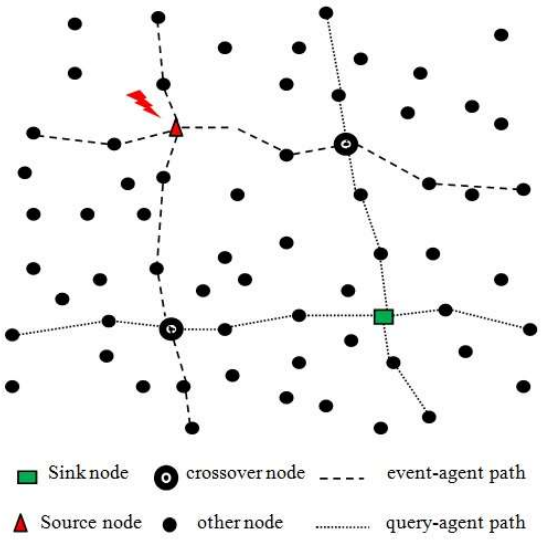


FIG IV.1. Concept global du protocole CSR.

IV.2.1 Modèle d'environnement

Dans ce qui suit, nous considérons un RCSF stationnaire avec une distribution aléatoire des nœuds. Nous supposons des transmissions fiables (sans perte de paquets) entre les nœuds. Nous supposons pour chaque nœud capteur, un rang « R » de transmission (captage, transmission et réception). Initialement, CSR doit passer par une phase de calibrage, où chaque nœud doit être en mesure de connaitre sa position géographique à partir d'un appareil spécifique tel qu'un GPS. Par contre, nous supposons que chaque nœud du réseau peut être un puits potentiel qui émet des requêtes pour certains évènements. Suite à l'apparition d'un évènement dans le réseau, quatre agents ont le rôle de disséminer la méta-donnée (une description de la donnée captée). Lorsque le nœud puits a besoin d'une information sur un évènement capté, il lancera aussi quatre requêtes (query). Dès que les chemins des agents et des requêtes se croisent, un chemin depuis la source vers le puits est établit, la donnée s'écoule donc tout au long de ce chemin. Chaque agent (resp. requête) est caractérisé par son TTL (Time To Live) qui détermine le nombre de saut que l'agent peut effectuer avant son expiration. Les valeurs des TTLs sont ajustées à la taille du réseau, ils doivent permettre à l'agent de progresser en profondeur dans le réseau [42-43].

Avant de procéder à une description détaillée du protocole CSR dans la prochaine section, nous présentons dans le tableau IV.1 la liste des principales notations mathématiques utilisées dans ce chapitre ainsi que leurs définitions.

Tableau IV.1. Liste et descriptions des principales notations mathématiques.

Notation mathématique	Description
A	Région du réseau
AB	Nœud Absorbant
A_R	Nombre de réceptions totales d'agents pour un nœud
d	Densité de la région *A*
$dist$	Distance
E	Energie
$\check{E}$	L'énergie nécessaire pour la transmission d'un agent
K	Paramètre da la région coins
L	Longueur du couloir
M	Nombre de nœuds dans la région *A*
M_{bk}	Nombre de voisins de chaque nœud dans le backbone path
n	Un nœud du réseau
N	Nombre total de nœuds dans le réseau
N_{bk}	Nombre de nœuds dans un backbone path
Nb (agent/query)	Nombre d'agents / de requêtes
R	Rang de transmission
R_Q	Taux de délivrance de requêtes
R_{mort}	Taux de mortalité
S	Nœud Star
T	Seuil énergétique de la région centrale virtuelle (VRC)
T_{bk}	Seuil énergétique du backbone path
$Traf$	Trafic généré
TTL (agent/query)	Nombre de sauts d'un agent / requête

Q_R	Nombre de réception totale de requêtes par un nœud
Qf	Nombre de requêtes à diffuser
W	Largueur du couloir
x	Coordonnée horizontale du nœud n
y	Coordonnée verticale du nœud n
α	Paramètre de communication
β	Nombre d'agents lancés à partir des régions coins
θq	Taux moyen pour un TTL garantissant le croisement de la requête avec les nœuds notifiés

IV.2.2 Principes de base du protocole CSR

Dès qu'un nœud source capte un évènement, le protocole CSR lance quatre agents. Pour son prochain saut, chaque agent choisi un nœud candidat à partir d'un couloir construit dans sa direction. Par exemple, un agent progressant vers l'Est qui est sur un nœud *n* dont la coordonnée horizontale est *x* et la coordonnée verticale est *y (n(x,y))*; construit un couloir d'une longueur *L* et hauteur ou largeur *W* et sélectionne un nœud voisin dans ce dernier. Le paramètre *L* doit être aussi grand que possible pour permettre des chemins courts (avec un grand nombre de sauts); il sera calculé selon la formule (IV.1) suivante:

$$L = R - \alpha \qquad \text{(IV.1)}$$

Où *R* est le rang de transmission du nœud n et α un paramètre qui dépend de la qualité de transmission. Par contre, le paramètre *W* doit être aussi petit que possible pour permettre un routage sur une ligne droite tout en évitant les situations de couloirs vides; il est calculé selon les formules (IV.2) et (IV.3) suivantes exprimées par rapport à la densité d'une région *A* [77].

$$d = (M\pi R^2)/A \qquad \text{(IV.2)}$$

$$W \geq \left(\frac{1}{L,d}\right) \qquad \text{(IV.3)}$$

Où *M* est le nombre de nœuds dans la région *A*.

Durant leurs progressions, chaque évènement et requête marque son prochain saut par un (S) pour l'identifier en tant que *star-node*. Afin de palier au problème de croisement des chemins entre les évènements et les requêtes -qui risquent, en effet, de passer l'un à côté de l'autre sont réellement se croiser-, chaque nœud star diffuse un message vers tous ses voisins les identifiant

comme des nœuds absorbants (AB). Ainsi, si une requête manque un nœud star, elle y est immédiatement routée via les nœuds (AB) (voir figure IV.2).

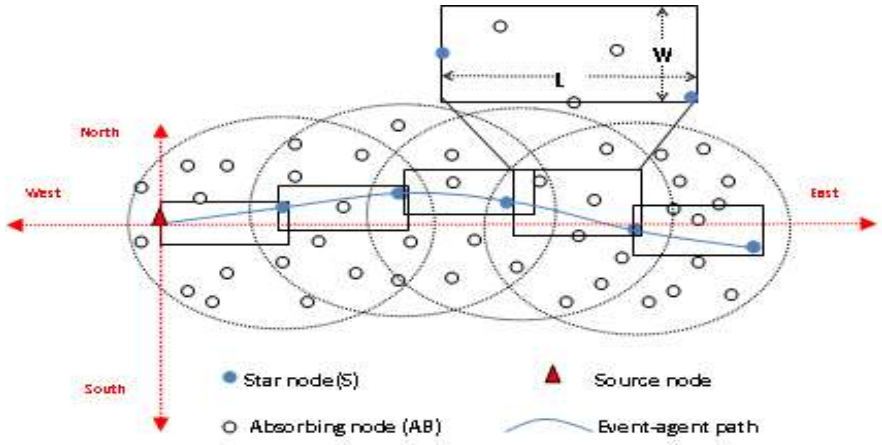


FIG IV.2. Principe de construction de couloirs dans CSR.

L'algorithme IV.1 suivant présente le principe de sélection des nœuds construisant le couloir dans une progression vers l'Est.

Algorithm IV.1. Eastern CSR routing scheme.

Input:

n(x, y)

Neighboring list *(n)*

Output:

eastern neighboring list *(n);*

Repeat until TTL (agent) = 0

For i = 1 to | neighboring list *(n)* |

{

if (($x_i > x$) && ($x_i \leq x+L$) && ($y_i \geq y_i - (W/2)$) && ($y_i \leq y_i + (W/2)$))

{ $n_i \in$ eastern_neighboring list *(n)*; }

}

IV.2.3 Améliorations du protocole CSR

Bien que le protocole CSR proposé soit simple, il présente deux inconvénients. Le premier, est le problème du couloir vide rencontré surtout dans les réseaux avec une faible densité où aucun nœud ne peut être sélectionné comme prochain saut. Le second, est relatif aux positions des nœuds sources et des nœuds puits dans le réseau. Si ces derniers sont à la proximité des frontières du réseau, le principe de la dissémination en étoile devient obsolète et un meilleur routage doit être définit. Dans cette section, nous proposons de résoudre ces problèmes en présentant quelques solutions alternatives.

IV.2.3.1 problème du couloir vide

Dans un but de contourner les vides, nous proposons la construction de couloirs alternatifs. Lorsqu'un agent progresse horizontalement en direction de l'Est ou l'Ouest, par exemple, une des solutions est de construire des couloirs verticalement jusqu'à ce qu'un voisin potentiel vers la direction initiale soit trouvé. Ainsi, l'agent peut continuer sa progression jusqu'à l'expiration de son TTL. Dans la figure IV.3, un agent qui progresse vers l'Est construit le couloir numéro 1 qui est vide (représenter par un rectangle). Le couloir alternatif numéro 2 vers le Nord est immédiatement construit et le plus proche voisin y est sélectionné. Puisque des voisins vers l'Est existent à l'intérieur du rang de transmission du nouveau voisin, l'agent continu sa progression en construisant les couloirs 3, 4, 5,...etc.

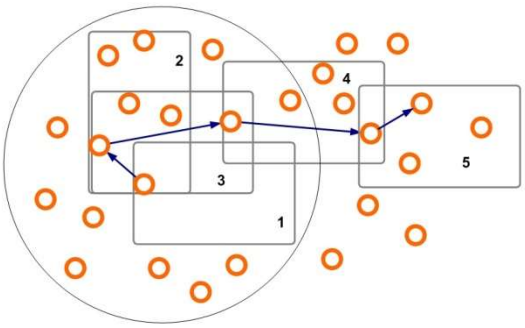


FIG IV.3. La construction de couloirs alternatifs dans CSR.

IV.2.3.2 problème des nœuds à proximité de la frontière du réseau

Ce problème concerne tous les nœuds source et puits qui se trouvent à une distance *dist* par rapport à la frontière du réseau tel qu'illustré par la figure IV.4. Suite à la détection d'un évènement ou à la demande d'une requête, nous proposons de construire un seul couloir au lieu de quatre en direction du centre du réseau. Par exemple, pour les nœuds positionnées sur les quatre coins du réseau: Nord-est, Nord-ouest, Sud-est et Sud-ouest, une progression diagonale de l'agent vers le centre du réseau est indiquée afin de maximiser les chances de croisement entre les chemins des agents et des requêtes.

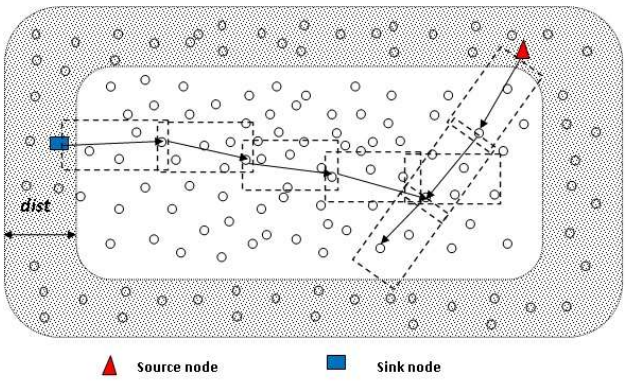


FIG IV.4. La dissémination diagonale et horizontale dans CSR.

IV.3 BACKBONE WEB ROUTING (BWR)

Dans beaucoup de déploiements réels des RCSF, les protocoles de dissémination et de routage qui utilisent l'information de localisation géographique des nœuds [45] [54-55] [57-58] ne peuvent fonctionner correctement. Souvent, les dispositifs de localisation géographique tels que le GPS (Global Positioning System) ou les algorithmes proactifs présentent un manque de précision et une surconsommation de l'énergie du réseau. De ce fait, la conception de techniques de dissémination de données indépendantes de l'information de localisation des nœuds dans un RCSF devient indispensable et très utile.

Dans cette section, nous présentons le protocole Backbone Web Routing (BWR) dont l'approche de routage se base sur la détermination d'une région centrale de rendez-vous où les agents et les requêtes se rencontrent pour accéder à l'information. Dans BWR, nous proposons de construire une structure simple et indépendante de tout system de localisation géographique.

IV.3.1 Les étapes d'execution du protcole BWR

BWR utilise le même modèle d'environnement que celui du protocole CSR proposé précédemment (section IV.2.1). Par contre, il nécessite une phase d'initialisation lors du déploiement des nœuds. Les principales étapes d'exécution du protocole BWR sont les suivantes :

IV.3.1.1 Etape 1: Initialisation

Dans BWR, les nœuds capteurs sont déployés aléatoirement mais avec quatre nœuds frontières ou balise (*corner nodes*). Chaque nœud frontière est positionné à l'extrémité des deux diagonales du réseau. Nous définissons la région frontière (*corner region*) comme étant le nœud frontière lui-même ainsi que ses voisins à *K*-saut; *K* est un paramètre déterminé selon la densité et l'échelle du réseau. Dans le reste de ce document, nous considérons K égale à 1. La figure IV.5 illustre un réseau avec quatre régions frontières.

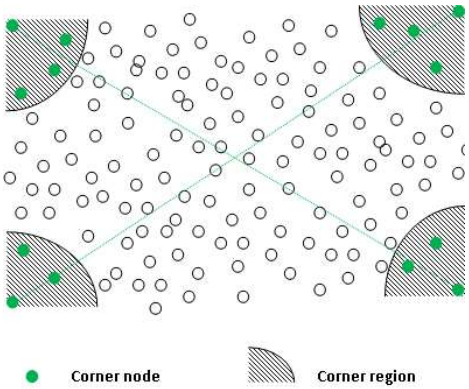


FIG IV.5. La phase de déploiement des nœuds dans BWR.

IV.3.1.2 Etape 2: Identification de la région centrale virtuelle

Avant de déterminer la région centrale virtuelle (*Virtuel Center Region (VRC)*), nous avons besoin de déterminer le centre virtuel du réseau (*Virtuel Centre node (VC)*). Ainsi, à partir des deux régions frontières -situées sur la partie droite du réseau- (voir figure IV.6), nous procédons au lancement d'un nombre β prédéfini d'agents dont le rôle est de construire des chemins pour atteindre les régions frontières opposées. Le principe de routage des agents est le même que celui du protocole FRA [48-50] défini dans le chapitre 2. De manière similaire au protocole Directed Diffusion (DD) [12], les chemins optimaux (en nombre de sauts) joignant chaque deux région frontière opposée sont considérés comme le backbone du réseau (*Backbone paths*). Le nœud (*Virtual Center node)* est donc situé à l'intersection des deux *backbones paths*; les nœuds voisins de ce dernier définissent la région VCR (voir figure IV.6).

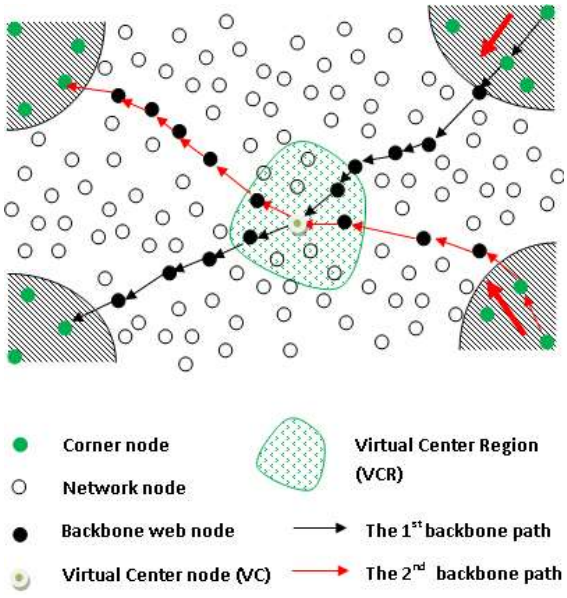


FIG IV.6. La construction du backbone dans BWR.

Le rôle de la VCR est de rassembler l'information sur tous les évènements détectés afin de permettre aux requêtes générées d'accéder de manière efficace à la donnée. Si nous considérons que tous les voisins du nœud VC à *p*-saut de ce dernier se trouvent à l'intérieur de la VCR, alors p doit être défini de telle sorte que la quantité d'énergie de la VCR (E_{VCR}) soit toujours supérieure à un certain seuil (T). En supposons $p = 0$, VCR est réduit au nœud VC lui-même.

$$E_{VCR} > T \qquad \text{(IV.4)}$$

$$E_{VCR} = E_{VCnode} + \sum_{i=1}^{p} E_i \qquad \text{(IV.5)}$$

Ou E_i est l'énergie de tous les voisins à i-saut du nœud VC.

IV.3.1.3 Etape 3: routage des agents et des requêtes

Dans BWR, un agent essaye toujours d'atteindre un *backbone path*. Une fois dessus, l'agent est immédiatement routé vers le nœud VC. A chaque saut, l'agent met à jour et synchronise son *EvtList*. Lorsque l'agent arrive au nœud VC, ce dernier diffuse l'information sur les évènements détectés ainsi que leurs chemins à tous les nœuds de la VCR. A cette fin, nous définissons une table de routage commune entre tous les nœuds de la VCR.

Le principe de routage d'une requête est toujours similaire à celui de l'agent. Mais, si la requête croise un nœud de la VCR avant d'atteindre le backbone ou le nœud VC, elle est directement routée vers la source telle qu'illustré sur la figure IV.7 (b).

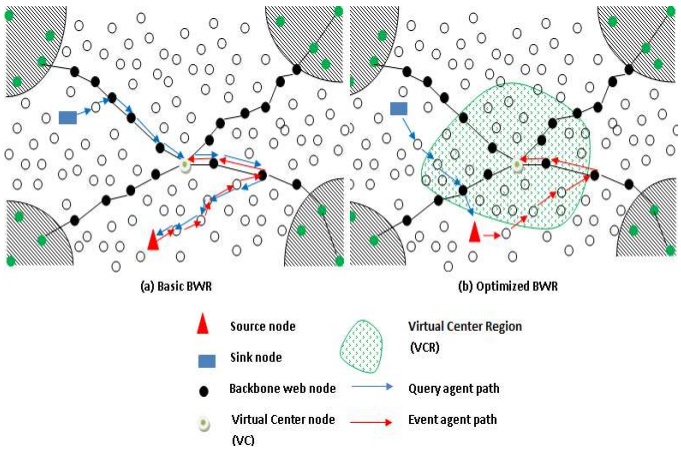


FIG IV.7. Routage des agents et des requêtes dans BWR (a) basique, (b) optimisé.

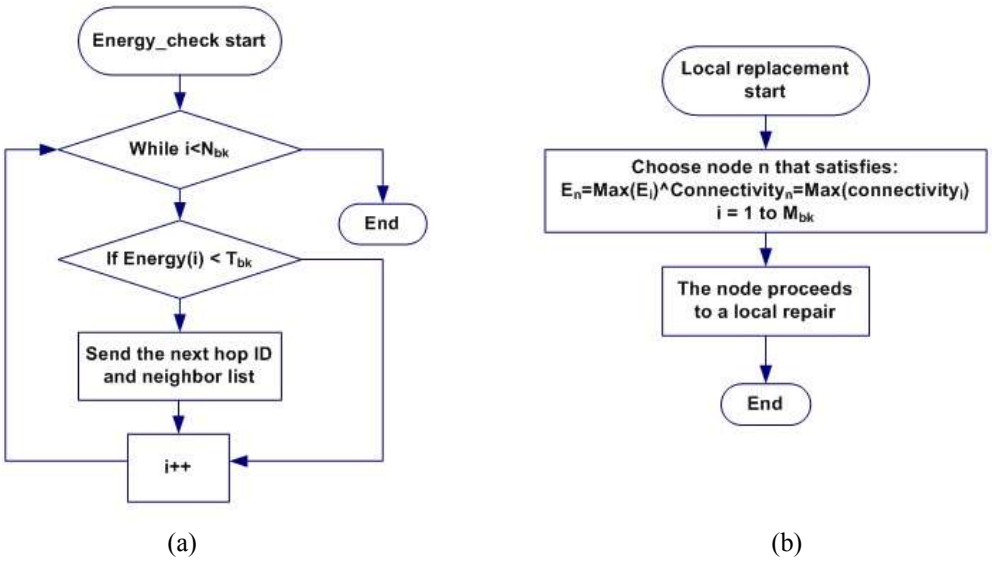
Le pseudo-algorithme IV.2 suivant présente la stratégie de routage des agents dans BWR.

Algorithme IV.2. Routage des agents dans BWR.

At an event occurrence :
Create & Send an agent;
The agent tries to reach a backbone-web node;
If not found: the agent chooses a next-hop according to its path direction.
At a backbone-web node :
The agent is routed toward the VC node;
The agent updates its *EvtList*;
At a VC-node :
The VC node broadcasts *EvtId* and *Evtpath* to all VCR nodes.

IV.3.1.4 Etape 4: Maintenance du backbone Web

Pour permettre au backbone de rester fonctionnel le plus longtemps que possible, les chemins vers la VCR doivent être toujours opérationnels. C'est pour cela que nous avons considéré la panne énergétique des nœuds sur les backbone paths. Nous proposons donc un schéma de remplacement local des nœuds sur les backbone paths sur la base de l'énergie résiduelle des nœuds voisins. Soit N_{bk} le nombre de nœuds de chaque backbone path et M_{bk} le nombre total de voisin de chaque nœud sur le backbone path. Périodiquement, la fonction *energy_check* contrôles l'énergie résiduelle des nœuds du backbone. Si cette dernière se trouve en dessous d'un seuil critique soit (T_{bk}), alors le nœud en question envoi l'identificateur de son prochain saut et sa liste de voisins à son prédécesseur sur le chemin. Le nœud prédécesseur va procéder au remplacement locale en choisissant parmi ses voisins le nœud adéquat telle qu'expliqué sur les organigrammes de la figure IV.8.



(a) (b)

FIG IV.8. Organigramme de (a) la fonction energy_check et (b) la procédure de remplacement locale (b).

IV.3.2 Compléxite du protcole BWR

Nous avons étudié la complexité du coût en énergie pour la construction du backbone dans le protocole BWR. La complexité d'un problème est définie par le coût du pire cas estimé par un algorithme optimal [78]. Nous définissons la complexité du coût en énergie comme étant la quantité d'énergie totale dissipée par les transmissions des agents jusqu'à l'identification de la VCR. Pour cela, nous considérons un réseau avec N capteurs sans fil stationnaires et homogènes $\{n_1, n_2, \dots, n_N\}$ déployés dans une région géographique.

Théorème: Le coût en énergie pour la construction du backbone-web est de l'ordre de

$\ominus (2(N-1))) \sim \ominus (N)$.

***Preuve**: Soit $\check{E}$ la quantité d'énergie dissipée lors de la transmission d'un agent par un nœud n. Dans l'algorithme du protocole BWR, (β) agents lancés depuis les régions frontières doivent construire la structure du backbone web en traversant le réseau d'une extrémité à une autre. Dans le pire des cas, ils devraient traverser les N nœuds du réseau et exécuter ainsi (N-1) transmissions. De ce fait, la quantité totale d'énergie dissipée est de $\beta.\check{E}(N)$*

IV.4 ANALYSE DES PERFORMANCES

Dans cette section, nous allons simuler les deux protocoles proposés CSR et BWR à l'aide du simulateur de réseau NS2.34 [76]. A cet effet, nous avons considéré un réseau de 1024 nœuds distribués aléatoirement sur une surface de 100*100 m^2. Chaque nœud capteur à un rang de transmission et de captage de 12m. Les TTL des agents et des requêtes sont déterminés selon la taille du réseau. Pour les besoins de simulation, ils ont été fixés respectivement à 50 et 100 sauts. Pour le protocole CSR, chaque nœud a besoin de connaitre sa position géographique. Tandis que dans BWR, les nœuds construisent leurs listes de voisinage en utilisant l'envoi local de message. Dans chaque scénario de simulation, deux évènements sont générés et 5 à 60 requêtes lancées à partir de différents nœuds puits (sink) du réseau sont à leur recherche. Afin de mieux évaluer les performances des protocoles proposés, nous les avons comparé à trois autres protocoles : le protocole ZRR [43] et AlAc [44] comme exemple de « free-location » protocoles avec différentes topologie réseau (zones pour ZRR et arbre pour AlAc) et le protocole ARR [57-58] [73] comme exemple de protocole à base de région de rendez-vous (centrale et autour de la frontière du réseau). L'ensemble de ces protocoles sont comparés en termes d'énergie consommée, de trafic généré, de taux de délivrance de requêtes, de latence, de taux de mortalité des nouds ainsi que de la durée de vie du réseau indiquée par la mesure de l'énergie totale consommée. Le tableau IV.2, résume les principaux paramètres de simulation des protocoles CSR et BWR.

Tableau IV.2. Paramètres de simulation des protocoles CSR et BWR.

Paramètre	**Valeur**
Taille du réseau	100X100 m^2
Nombre de nœuds	[256, 512, 1024]
Déploiement des nœuds	aléatoire
Rang de transmission	12 *mètres*
Energie initiale pour chaque nœud (joule)	5 joules
Energie consommée pour la transmission d'agents	0.6 *watt*
Energie consommée à la réception d'agents	0.3 *watt*
TTL agent	50 sauts
TTL requête	100 sauts

IV.4.1. Métriques

Dans ce qui suit, la liste des métriques utilisées:

• Energie consommée: la quantité d'énergie totale consommée pour router q requêtes dans le protocole Rumor Routing (RR) [42] est donnée par la formule IV.6 suivante:

$$\mathrm{Et(q)} = \mathrm{Esetup} + \mathrm{q} * \left(Epath + N * \frac{Qtotal - Qf}{Qtotal}\right) \qquad \text{(IV.6)}$$

Où:

q: est le nombre de requêtes ;

Et(q) : est la quantité d'énergie nécessaire pour router q requêtes ;

Esetup: est la quantité d'énergie nécessaire pour établir les chemins notifiés par les agents ;

Epath: est l'énergie moyenne dissipée lors d'un saut d'un agent ;

N: est le nombre de nœuds du réseau ;

Qtotal: est le nombre total de requêtes générées ;

Qf: est le nombre de requêtes qui ont besoin d'être diffusées.

Pour les besoins de simulation et afin de rendre notre analyse plus réelle, nous proposons d'utiliser la formule suivante donnée par l'équation IV.7 [60] et calculée pour chaque transmission d'agents et de requêtes dans le réseau.

$$E = \check{E} * [Nb(agent) * TTL(agent) + (Nb(query) - Qf) * (TTL(query) * \theta q) + (Qf * N)] \qquad \text{(IV.7)}$$

Où:

Ĕ: est l'énergie dissipée lors d'un saut ;

Nb(agent): est le nombre d'agents ;

TTL (agent): est le nombre total de sauts pour un agent ;

Nb(query): est le nombre de requêtes ;

TTL (query): est le nombre total de sauts pour une requête ;

Qf: est le nombre de requêtes qui ont besoin d'être diffusées.

θq: Taux moyen pour un TTL garantissant le croisement de la requête avec les nœuds notifiés ;

N: est le nombre de nœuds du réseau.

• *Traffic de requête (Query traffic)*: le trafic généré par les requêtes est le nombre de toutes les transmissions ou réceptions de requêtes dans le réseau.

• Taux de délivrance de requête (TDR) (*Query delivery rate)*: une requête est considérée comme délivrée si elle atteint la source de l'évènement recherché. Le taux de délivrance de requêtes est le rapport R_Q des requêtes correctement routées sur l'ensemble de celles générées dans le réseau pour un évènement particulier. Si Q_d est le nombre total des requêtes délivrées et Q_g est le nombre total des requêtes générées alors le taux $R_Q$est donné par la formule (IV.8) suivante :

$$R_Q = Q_d/Q_g \qquad \text{(IV.8)}$$

• Le temps d'établissement de chemin (la latence): c'est la différence entre le temps où la requête atteint le nœud notifié par l'évènement (croisement) et celui où elle fut générée.

• Le taux de mortalité des nœuds, R_{mort}: pour calculer ce taux, nous avons considéré l'énergie résiduelle de chaque nœud. Si cette dernière est en dessous d'un certain seuil, nous considérons le nœud comme mort. Le taux R_{mort} est donc calculé par la formule IV.9 suivante :

$$R_{mort} = \sum(dead\ node) \times 100/N \qquad \text{(IV.9)}$$

Où N est le nombre de nœuds dans le réseau.

IV.4.1. Comparaison et interprétation des résultats

Dans le but d'évaluer le coût de construction du backbone web dans le protocole BWR, nous l'avons comparé à deux autres structures topologiques à savoir les clusters dans le protocole ZRR et l'arbre à niveau dans le protocole AlAc. Dans ZRR, chaque cluster-head ou chef de groupe diffuse des messages d'invitation à ses voisins locaux. Ces derniers répètent le même procédé jusqu'à ce que chaque nœud devienne membre d'un seul cluster. Alors que dans le protocole AlAc, un nœud racine aléatoire déclenche la construction de l'arbre à niveau en diffusant à ses voisins directs les messages « Tree-build packets ». Chaque voisin met à jour

son niveau et diffuse à son tour ce message à travers le réseau. Les résultats de la figure IV.9 nous renseignent sur le trafic généré en nombre de messages transmis lors de la construction des structures topologiques des trois protocoles en fonction du nombre de nœuds. Notons que pour le protocole BWR, le nombre de nœuds n'affecte pas le trafic généré lors de la création du backbone. Par contre, dans ZRR et AlAc ce dernier augmente considérablement. Ceci est expliqué comme suit : lors de la création des topologies en clusters ou en arbre, les nœuds du réseau exécutent un processus de transmission répétitif qui est d'ailleurs plus important avec plusieurs cluster-head qu'un seul nœud racine. En revanche, dans BWR, le coût de communication est réduit à la transmission d'agents. Plus d'agents sont transmis, plus d'énergie est consommée mais cette dernière est nettement plus inférieure à celle des protocoles ZRR et AlAc.

Le trafic total généré dans les trois protocoles ZRR, AlAc et BWR, est respectivement [1.2 - 7.5] x 10^4, [0.5 - 2.2] x 10^4 and [0 - 0.5] x 10^4 transmissions. Ces résultats confirment que le backbone web proposé permet de préserver la transmission d'environ 7000 messages en comparaison à la construction de clustres et environ 1700 messages pour les structures en arbre.

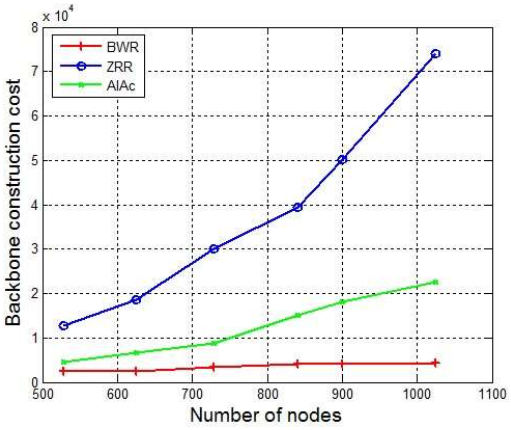


FIG IV.9. Comparaison du trafic.

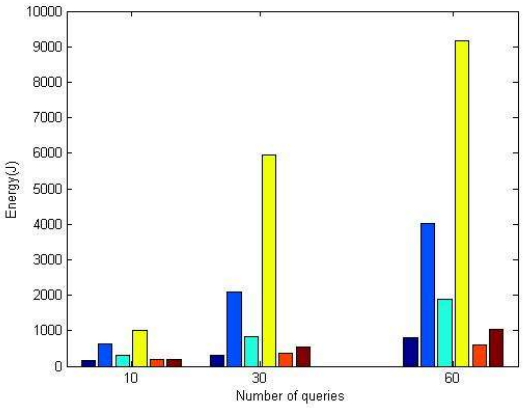


FIG IV.10. Comparaison de l'énergie.

La figure IV.10 montre l'énergie moyenne consommée par les protocoles comparés dans un réseau de 1024 nœuds. Nous avons exécuté 3 scénarios dans lesquels le nombre de requêtes varie de 10, 30 à 60. Tel qu'illustré par la figure IV.10, le nombre de requêtes affecte considérablement les performances des protocoles ZRR, ARR-EDGE et AlAc. Pour ZRR et ARR-EDGE, les requêtes générées se propagent sur des chemins non optimaux (en termes de nombre de sauts) ce qui va dissiper une quantité importante d'énergie en se déplaçant d'une zone à une autre ou autour de la frontière du réseau. Pour AlAc, toutes les requêtes générées circulent à travers le même niveau de l'arbre ce qui engendre une surconsommation de l'énergie comparé aux protocoles BWR, CSR et ARR-COG. La stratégie de routage du protocole CSR et les chemins optimaux construits dans BWR et ARR-COG minimisent le nombre de transmissions et sauvegarde ainsi l'énergie consommée.

Les résultats de la figure IV.11 montrent le trafic généré par l'exécution de plusieurs protocoles. Nous remarquons que les protocoles CSR, BWR et ARR-COG améliorent considérablement le

trafic généré en comparaison avec les protocoles ZRR et AlAc. Concernant CSR, cette amélioration est due à la stratégie de routage en ligne droite implémentée qui réduits le nombre de transmissions générées. Pour BWR et ARR-COG, les chemins optimaux générés aux alentours du centre du réseau sont potentiellement la raison du trafic réduits ce qui n'est pas le cas pour les protocoles AlAc et ARR-EDGE dont les chemins deviennent plus longs surtout sur la frontière du réseau. Enfin le protocole ZRR enregistre le pire trafic car en dépit de ses zones sa stratégie de routage au sein des zones reste aléatoire ce qui engendre un nombre important de transmissions.

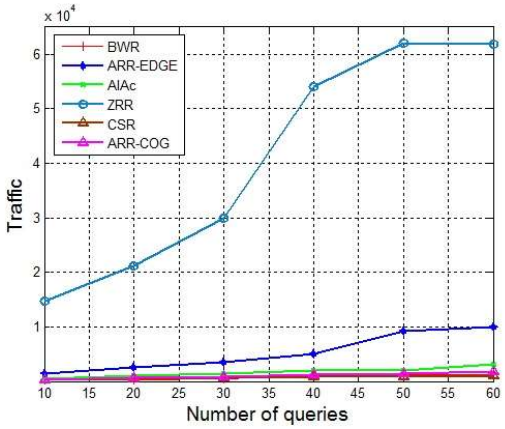


FIG IV.11. Comparaison du trafic.

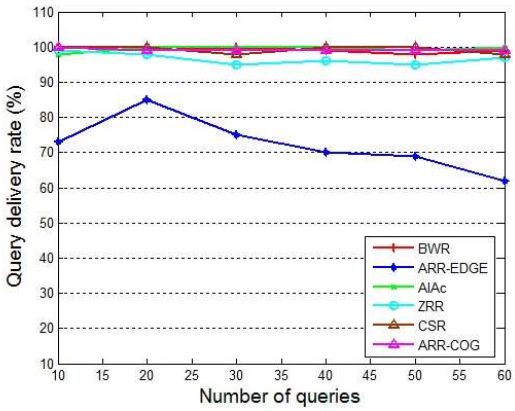


FIG IV.12. Comparaison du TDR.

La figure IV.12 montre le taux de délivrance de requêtes pour 10 à 60 requêtes générées. Les résultats montrent que ce taux est satisfaisant [90%-100%] mis à part pour le protocole ARR-EDGE. Ceci peut être expliqué comme suit : dans ARR-EDGE, un seul agent est responsable de la dispersion de la rumeur et une seule requête est à sa recherche sur la frontière du réseau. Lorsque le nombre de requêtes augmente, un trafic important est généré sur les nœuds frontières qui résultent en une congestion du réseau ce qui probablement fait chuter les performances de ce protocole.

La latence est une métrique qui permet de mesurer la qualité des chemins construits en termes de nombre de sauts. Les résultats de la figure IV.13 peuvent être discutés selon trois groupes de performances. Le premier, celui des protocoles : BWR, CSR et AlAc, réussi à créer des chemins entre 3 *ms* et 9 *ms*. La meilleure performance est celle de BWR grâce à son backbone qui route les évènements et les requêtes directement vers la région centrale de rendez-vous. BWR réduit la latence de près de 20% en comparaison au deuxième groupe. Le deuxième groupe, celui des protocoles ARR-COG et ARR-EDGE construit des chemins en un peu plus de temps entre 15 *ms* et 25 *ms*. Et enfin, le dernier groupe du protocole ZRR qui enregistre la plus grande latence. Les résultats des figures IV.11 et IV.13 confirment que pour les réseaux à grande échelle, les approches à base de points de rendez-vous et les stratégies de routage en ligne droite sont en générale meilleures que celles des protocoles à marches aléatoires.

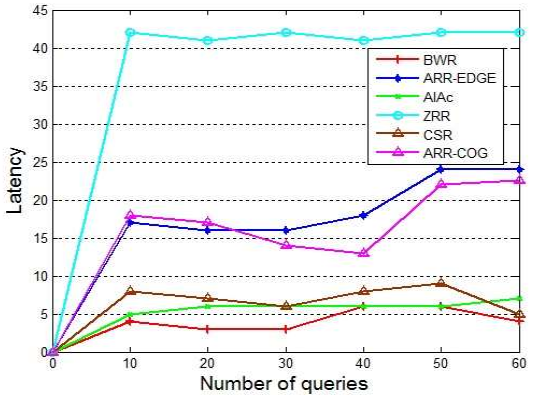


FIG IV.13. Comparaison de la latence.

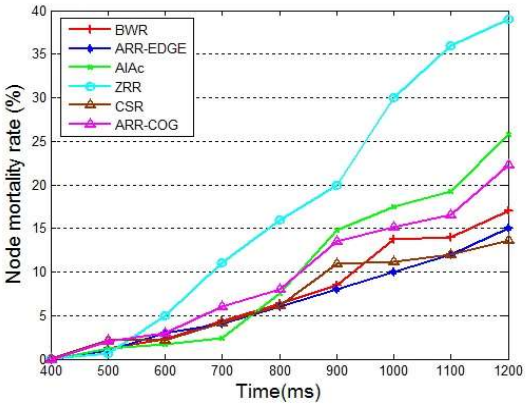


FIG IV.14. Taux de mortalité des nœuds.

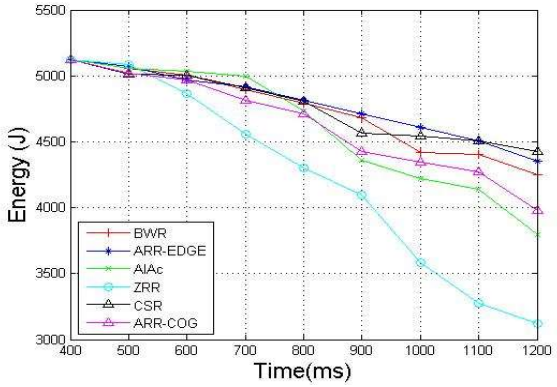


FIG IV.15. Comparaison de la durée de vie du réseau.

Les résultats des figures IV.14 et IV.15 montrent le taux de mortalité des nœuds ainsi que la durée de vie du réseau pour les protocoles comparés. Le protocole ZRR exploite un routage à marche aléatoire. Le nombre de nœuds qui participent au routage augmente continuellement et la durée de vie du réseau diminue de manière significative. Pour le protocole AlAc, le taux de mortalité des nœuds diminue par rapport à ZRR car la dissémination de données se fais par les nœuds d'un même niveau dans l'arbre ce qui sauvegarde l'énergie du reste des nœuds. Pour les protocoles BWR et ARR-COG, ce taux progresse de la même manière mais BWR l'améliore nettement grâce à sa stratégie de routage optimisée qui améliore la qualité des chemins. Enfin, les protocoles CSR et ARR-EDGE enregistrent les meilleures performances. Pour CSR, ceci est due grâce à sa stratégie de dissémination homogène et uniforme qui assure un équilibrage de charge. D'un autre côté, le nombre de nœuds frontière dans ARR-EDGE est beaucoup plus important que ceux dans le centre ce qui permet aux nœuds de perdre leurs énergies de manière plus uniforme.

IV.5 DISCUSSION

Les deux protocoles proposés dans ce chapitre CSR et BWR sont conçus pour assurer un accès efficace à la donnée pour les applications critiques qui ne tolèrent pas un délai de réponse. Ils sont ainsi efficients en énergie et en latence grâce à leurs stratégies de routage respectives.

Dans le protocole CSR, les nœuds du réseau participent de manière uniforme dans le processus de dissémination de données. Par contre, pour le protocole BWR, l'équilibrage de charge au sein du réseau reste un challenge. Dans certains travaux récents [54-55] [65], un nombre satisfaisant de challenges pour les RCSF a été assuré tel que l'efficacité énergétique, l'équilibrage de charge et la qualité des chemins construits. Cependant, les solutions proposées assument souvent des hypothèses fortes et parfois loin des déploiements réels des RCSF. Par exemple, dans [55], le protocole « *RER* » est proposé. *RER* est un protocole temps réel et efficient en énergie qui prend en considération la mobilité des puits. Il se base sur une architecture multi couches avec 4 niveaux d'applications. Grace à son module d'unité d'estimation de zones (ZEU) il peut estimer la zone où la requête à le plus de chance d'être générée afin de restreindre sa diffusion. Par contre, l'efficacité de ce module est très dépendante de la capacité du nœud *S* (initiateur de la requête) à correctement estimer sa zone. En d'autres termes, le puits *S* a besoin de connaitre la localisation exacte des destinations de ses requêtes. Dans le cas contraire, RER devra inévitablement avoir recours à la contraignante technique de diffusion sur l'ensemble du réseau. Actuellement, équipé chaque nœud capteur d'un dispositif de localisation tel que le GPS devient couteux et parfois en dessus du budget de beaucoup de projets de déploiements et de recherche [79].

BWR est un protocole qui est proposé pour les applications critiques. Les requêtes et les évènements sont routés vers une région centrale de rendez-vous afin d'accélérer le croisement de leurs chemins respectifs et assurer ainsi un accès efficace à la donnée. Pour les applications moins urgentes, il est plus intéressant de distribuer les régions de rendez-vous ou de proposer une région de rendez-vous dynamique. Par exemple, afin d'étendre la durée de vie du réseau et assurer un meilleur équilibrage de charge, nous pouvons proposer une phase d'initialisation du protocole BWR avec huit (8) nœuds balises au lieu de quatre (4). Ainsi, tel qu'illustré par la figure IV.16, six agents sont utilisés pour construire un backbone avec cinq régions de rendez-vous distribuées sur le réseau. La prochaine étape consiste à définir une stratégie de management pour assurer l'équilibrage de charge. A titre d'exemple, nous pouvons assigner la tâche de routage aux VCR formant une diagonale tandis que les autres sont dans un état « sleep ». D'autres mécanismes de scheduling du temps (*time scheduling*) peuvent être proposés pour supporter cette alternative.

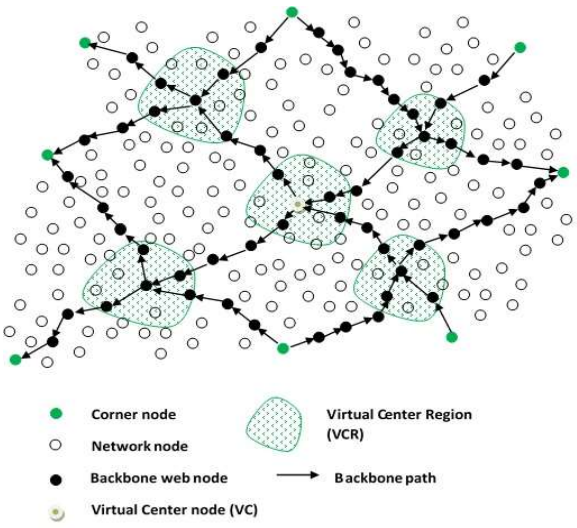


FIG IV.16. Le protocole BWR avec plusieurs VCR distribuées.

IV.6 CONCLUSION

Dans certaines applications de secours ou de gestion de catastrophe, nous avons besoin d'accéder à la donnée de manière efficace et rapide quel que soit le type d'environnements dans lequel nous nous trouvons. C'est dans ce contexte, que nous avons proposé dans ce chapitre, deux protocoles qui assurent un accès efficace à la donnée. Le premier protocole CSR exploite l'information de localisation des nœuds pour assurer un routage uniforme et rapide dans le réseau. Il ne nécessite pas la connaissance préalable de la topologie générale du réseau.

Pour les environnements dépourvus de tout système de localisation géographique, nous proposons le protocole BWR qui est une approche de routage basée sur une région centrale de rendez-vous. Une étude de la complexité du coût énergétique de la construction du backbone web dans BWR a été fournie afin de prouver l'efficacité de son déploiement.

Nous avons également évalué les performances des deux protocoles proposés à travers des simulations en mesurant différentes métriques. Les résultats obtenus sont satisfaisants et encourageants car les performances ne sont pas affectées par l'augmentation de la taille du réseau (passage à la l'échelle).

Ce chapitre termine ainsi la description des différents protocoles constituants nos contributions dans le cadre de cette thèse. Dans ce qui suit, nous présenterons la conclusion générale ainsi que d'éventuelles perspectives en vue.

CONCLUSION GENERALE

Dans les travaux de cette thèse, nous nous somme intéressé à la problématique de routage et de dissémination de données dans les RCSF. Nous avons commencé dans la première partie par présenter les principaux protocoles de routage et de dissémination qui existent dans la littérature. Par la suite, nous nous sommes intéressés à une classe particulière de ces protocoles dite : protocoles à marche aléatoire (*random walk routing*) ou (*unicast agents based routing prtocols*). Le concept clé qui a motivé l'étude des protocoles de cette classe ainsi que leur classification proposée au niveau du chapitre 2, est le fait d'utiliser la notion d'agents de rumeur pour propager la métadonnée relative un évènement capté dans le réseau. L'utilisation du concept d'agent, permet en effet deux gains non négligeables pour les RCSF. Le premier, est que la taille des agents ainsi que leur transmission en unicast permet de préserver la bande passante des RCSF. Le second est relatif au principe de coopération entre agents qui permet d'aboutir à des mises à jour pour optimiser leurs cheminements et améliorer ainsi la qualité des chemins construits.

La seconde partie de ce document, présente nos contributions pour améliorer les protocoles de routages existants dans cette classe. Dans le chapitre 3, nous avons proposé le protocole *Fast Rumor Agent (FRA)* qui implémente une technique de routage en ligne droite permettant une propagation rapide des agents au sein du réseau. Nous avons proposé également une étude de mise à l'échelle et une réplication de données pour pallier au problème de défaillance de chemin. Par la suite, nous avons amélioré ce protocole en assurant une distribution uniforme et profonde dans le réseau aboutissant au protocole *Efficient Data Access based on Rumour Dissemination (EDARD)*.

Dans le chapitre 4, nous avons considérés des environnements de RCSF plus proches de la réalité et nous avons proposé deux autres protocoles de dissémination. Le protocole *Corridor Star Routing (CSR)* propose un routage en ligne droite sur la base de l'information de localisation des nœuds. CSR établit des chemins efficacement sur la base de construction progressive de couloirs dans le réseau. En cas d'absence ou de non disponibilité de l'information de localisation des nœuds, nous avons proposé le protocole *Backbone Web Routing (BWR).* BWR propose un backbone efficace pour le routage et peu couteux en terme de construction et de maintenance comparé à d'autres structures topologiques telles que les clusters, arbre, grille, …etc.

Tous les protocoles développés ont été implémentés à travers le simulateur de réseau Network Simulateur (NS). L'analyse des performances a donnée des résultats satisfaisants. Selon les résultats obtenus, nous avons réussi à réduire le trafic et la latence engendrée grâce à la réduction du nombre de transmissions globales dans le réseau. Les résultats ont montré également que l'approche de routage à base de points de rendez-vous proposée dans BWR surpassée de manière générales les techniques à marches aléatoire.

Perspectives

Dans ce qui suit, nous présentons quelques perspectives et travaux futurs pour améliorer les protocoles proposés :

Notre première idée de contribution est de proposer une amélioration à la marche aléatoire des agents en proposant un mécanisme de routage en ligne droite. Ce mécanisme est implémenté dans le protocole *FRA* qui permet de garder une information de voisinage et un historique du parcours de l'agent. Par la suite, pour adapter *FRA* aux applications avec des données critiques, nous avons proposé dans *FRA_R* d'implémenter une réplication de données pour y améliorer l'accès. Cependant, une étude plus détaillée et plus formelle sur l'implémentation de la réplication doit être envisagée. De nouvelles données doivent être prises en considération telle que la nouvelle taille du paquet agent, le coût de transmission en bande passante et en énergie ainsi que la rupture des liens, souvent très envisageable dans les RCSF.

Pour une étude plus performante sur la mise à l'échelle des protocoles proposés, nous envisageons de les simuler sur des réseaux au-delà de 1024 nœuds avec différentes densités. A cet effet, une étude formelle sur la définition des Time To Live (TTL) des agents par rapport à la taille du réseau doit être fournie car ce paramètre est un élément clé dans la conception de ces protocoles.

Enfin, comme le protocole BWR proposé est l'ébauche d'une première idée, beaucoup de travail et d'améliorations sont à envisager. En premier lieu, nous comptons optimiser encore la construction du backbone web en minimisant le trafic induit par la transmission des agents. De plus, afin de permettre un meilleur équilibrage de charge, nous envisageons une structure dynamique distribuée avec de nouveaux mécanismes de maintenance et de scheduling entre les nœuds. Nous devons faire face aussi au problème des vides qui risquent de faire diminuer les performances du protocole. En effet, différents types de vides peuvent se former dans un RCSF créant des régions géographiquement corrélées.

REFERENCES BIBLIOGRAPHIQUES

[1] Chuan Zhu, ChunlinZheng, LeiShu, GuangjieHan (2012). "A survey on coverage and connectivity issues in wireless sensor networks". Journal of Network and Computer Applications (35) 619–632.

[2] Huang Lu (2009). "A Novel Routing Algorithm for Hierarchical Wireless Sensor Networks". Master thesis of Engineering at the University of Tsukuba.

[3] Zahoor Ali Khan (2012). "Advanced Zonal Rectangular LEACH (AZR-LEACH): An Energy Efficient Routing Protocol for Wireless Sensor Networks". Master Thesis of Computer Science at Dalhousie University Halifax, Nova Scotia.

[4] Mohamed Aissani (2011). "Optimisation du routage dans les réseaux de capteurs pour les applications temps-réel". Thèse de doctorat en cotutelle (USTHB- Université Paris-Est).

[5] Kamal Beydoun (2009). "Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs". Thèse de doctorat de l'Université de Franche Comté.

[6] Zheng XL, Wan M (2014). "A survey on data dissemination in wireless sensor networks". Journal of computer science and technology 29 (3): 470–486.

[7] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002) "Wireless sensor networks: a survey". Computer Networks, 38(4), 393–422.

[8] K. Akkaya and M. Younis, (May 2005). "A survey on routing protocols for wireless sensor networks," Elsevier Ad Hoc Networks, Vol. 3(3), pp. 325-349,

[9] Sajal K.D, Ammari M.H (2009) "Wireless Sensor Networks: A Networking Perspective". Edited by Jun Zheng and Abbas Jamalipour Copyright © 2009 Institute of Electrical and Electronics Engineers. University of Texas at Arlington, USA.

[10] W. Heinzelman, J. Kulik, and H. Balakrishnan, (1999) "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks". In proceeding of the ACM/IEEE Conference on Mobile Computing and Networking (MobiCom), pp. 174-185, Seattle, WA.

[11] J. Kulik, W.R. Heinzelman, and H. Balakrishnan, (2002) "Negotiation-Based Protocols for Disseminating Information in Wireless Sensor Networks," ACM Wireless Networks, Vol. 8(3), pp. 169–185.

[12] Intanagonwiwat, C., Govindan, R., & Estrin, D. (2000). "Directed diffusion: A scalable and robust communication paradigm for sensor networks". In Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, (pp.56-67).

[13] Belkadi, M., Aoudjit, R., Daoui, M., & Lalam, M. (2013) "Energy-efficient secure directed diffusion protocol for wireless sensor networks". International Journal of Information Technology and Computer Science, 6(1), 50.

[14] W.R. Heinzelman, A. Chetrakasan, and H. Balakrishnan, (2000) "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", in proceeding of the 33rd Annual Hawaii Int' Conference on System Sciences (HICSS), Vol. 2, pp. 10-19, Hawaii.4-7.

[15] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, (2002) "An Application-Specific Protocol Architecture for Wireless Micro sensor Networks," IEEE Transactions on Wireless Communications, Vol. 1(4), pp. 660-670.

[16] S. Lindsey and C. Raghavendra, (2002) "PEGASIS: Power-Efficient Gathering in Sensor Information Systems," in proceeding of the IEEE Aerospace Conference, Vol. 3(1), pp. 1125-1130, Los Angeles, CA, USA.

[17] A. Manjeshwar and D. P. Agrawal, (2001) "TEEN: A routing protocol for enhanced efficiency in wireless sensor networks", in proceeding of the IPDPS' 01, San Francisco, CA, Apr. 2001, pp. 2009 – 2015.

[18] V. Rodoplu and T. H. Meng, (1999) "Minimum energy mobile wireless networks," IEEE Journal on Selected Areas in Communications, vol. 17, no. 8, pp. 1333-1344.

[19] B. Nath and D. Niculescu, (2003) "Routing on a curve", ACM SIGCOMM Computer Communication Review, vol. 33, no. 1, pp. 155-160.

[20] Y. Xu, J. Heidemann and D. Estrin, (2001) "Geography-informed energy conservation for ad-hoc routing," in Proceedings of the ACM/IEEE MobiCom'01, Rome, Italy.

[21] Y. Yu, R. Govindan, and D. Estrin, (2001) "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks", Technical Report UCLA/CSD - TR - 01 - 0023, UCLA Computer Science Department.

[22] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, (2001) "Highly-resilient, energy-efficient multipath routing in wireless sensor networks", in proceedings of the ACM MobiHoc' 01, Long Beach, CA, Oct. pp. 251 – 254.

[23] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, (2001) "Highly-resilient, energy-efficient multipath routing in wireless sensor networks", Mobile Computing and Communications Review, vol. 5, no. 4, pp. 10 – 24.

[24] W. Lou, (2005) "An efficient N-to-1 multipath routing protocol in wireless sensor networks", in proceedings of the IEEE MASS' 05, Washington, DC, pp1 – 8.

[25] N. Sadagopan, B. Krishnamachari, and A. Helmy, (2003) "The ACQUIRE mechanism for efficient querying in sensor networks," in proceeding of the IEEE Int'l Workshop on Sensor Network Protocol and Applications, pp. 149-155, Anchorage, Alaska, May 11-12.

[26] K. Sohrabi and J. Pottie, (2000) "Protocols for Self-Organization of a Wireless Sensor Network," IEEE Pers. Commun., vol. 7, no. 5, pp. 16-27.

[27] T. He and al, (2003) "SPEED: A Stateless Protocol for Real-time Communication in Sensor Networks," in Proceeding of the International Conference on Distributed Computer Systems, Providence.

[28]T. He, J.A. Stankovic, C. Lu, and T. Abdelzaher, (2005) "A Spatiotemporal Communication Protocol for Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 16(10), pp. 995-1006.

[29] Shio Kumar Singh 1, M P Singh, and D K Singh, (2010) "Routing Protocols in Wireless Sensor Networks – A Survey", International Journal of Computer Science & Engineering Survey (IJCSES) Vol.1, No.2.

[30] Haas Z J, Halpern J Y, Li L. (2002) "Gossip-based ad hoc routing". In Proceeding of the 21st IEEE INFOCOM, pp.1707-1716.

[31] Kyasanur P, Choudhury R R, Gupta I. (2005) "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks". In proceeding of the 3rd IEEE MASS, pp.91-100.

[32] K. Akkaya, M. Younis, (2003) "An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks," Proc. of the 23rd IEEE Int'l Conference on Distributed Computing Systems (ICDCS), pp. 710–715, Washington, DC, USA, May 19-22.

[33]J. Luo; J. P. Hubaux, (2005) "Joint mobility and routing for lifetime elongation in wireless sensor networks", in proceedings of the IEEE INFOCOM' 05, vol. 3, Miami, pp. 1735-1746.

[34] R. C. Shah, S. Roy, S. Jain, and W. Brunette, (2003) "Data MULEs: Modeling a three tier architecture for sparse sensor networks", in proceedings of the SNPA' 03, Anchorage, AK, pp. 30-41.

[35] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two - tier data dissemination model for large-scale wireless sensor networks", (2002) in proceeding of the ACM/IEEE MobiCom' 02, Atlanta, GA, pp. 148-159.

[36] Luo, H., Ye, F., Cheng, J., Lu, S., & Zhang, L. (2005). TTDD: Two-Tier Data Dissemination in Large-Scale Wireless Sensor Networks. Wireless Networks (2005): 11, 161-175.

[37] H. S. Kim, T. Abdelzaher and W. H. Kwon, (2003) "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks", in proceeding of the ACM SenSys' 03, Los Angeles, CA, Nov. pp. 193-204.

[38] W. Zhang, G. Cao, and T. La Porta, (2004) "Dynamic proxy tree - based data dissemination schemes for wireless sensor networks", in proceeding of the IEEE MASS' 04, Fort Lauderdale, FL, pp. 21-30.

[39] M. Chu, H. Haussecker, and F. Zhao, (2002) "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks", International Journal of High Performance Computing Applications, vol. 16, no. 3, pp. 293-313.

[40] X. Du and F. Lin, (2005) "Improving routing in sensor networks with heterogeneous sensor nodes", in proceeding of the IEEE VTC' 05, Dallas, TX, pp. 2528-2532.

[41] Braginsky, D., & Estrin, D. (2002). "Rumor routing algorithm for sensor networks". In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (pp. 22-31).

[42] Haenselmann, T., & Effelsberg, W. (2005). "Forking Agents in Sensor Networks".GI Jahrestagung (2), 328-333.

[43] Banka, T., Tandon, G., & Jayasumana, A. P. (2005). "Zonal rumor routing for wireless sensor networks". In proceeding of the IEEE International Conference on Information Technology: Coding and Computing. ITCC. (2), 562-567.

[44] Chim, T. W. (2005) "Along & across algorithm for routing events and queries in wireless sensor networks". In proceeding of the IEEE International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS725-728.

[45] Shokrzadeh, H., Haghighat, A. T., &Nayebi, A. (2009) "New routing framework base on rumor routing in wireless sensor networks". Computer Communications, 32(1), 86-93.

[46] Chou, C. F., Su, J. J., & Chen, C.Y. (2005) "Straight line routing for wireless sensor networks". In proceeding of the10th IEEE Symposium on Computers and Communications, ISCC. pp 110-115.

[47] Liu, H. H.,Su, J. J.,& Chou, C. F. (2015) "On Energy-Efficient Straight-Line Routing Protocol for Wireless Sensor Networks", IEEE systems journal, 1-9.

[48] L. Kheroua, S. Moussaoui, F. Z. Djemmah & R. Cherif, (2010) " FRA: Fast Rumor Agents for Wireless Sensor Networks", In proceeding of the 4th International Conference On Sensing Technologies. (518-523) June 3-5 Lecce Italy. 2010. http://www-ist.massey.ac.nz/seat/conferences/icst2011/new/call_for_papers.asp

[49] Kheroua,L., Moussaoui, S., Mansour, L. (2011) "An Agent based Rumor Dissemination for Routing in Wireless Sensor Networks" . In proceeding of the 10th IEEE International Symposium on Programming and Systems. ISPS'2011 46-53.

[50] Kheroua, L., Moussaoui, S., & Mansour, L. (2011) "An Efficient Agent Based Rumor Propagation for Wireless Sensor Networks". International Journal of Measurement Technologies and Instrumentation Engineering, 1(2), 61-72.

[51] Kheroua, L., Moussaoui, S., & Mansour, L. (2013) "EDARD: Efficient Data Access based on Rumour Dissemination in wireless sensor networks". International Journal of Trust Management in Computing and Communications, 1(1), 73-84.

[52] Kheroua, L., Moussaoui, S., Guerroumi, M. et al. (2018) "Two energy and time-efficient data dissemination protocols for large-scale wireless sensor networks" Telecommun Syst (2018). https://doi.org/10.1007/s11235-018-0471-z (https://rdcu.be/OXRm)

[53] Wang, Z. H., Chen, K., Lin, M., & Yu, M. (2010). "Energy-efficient clustering rumor routing protocol for wireless sensor networks". In proceeding of the IEEE 7th International Conference on Ubiquitous Intelligence Computing and Autonomic & Trusted Computing (UIC/ATC), 200-205.

[54] Ahvar, E., Serral-Gracià, R.,Marín-Tordera, E.,Masip-Bruin, X.,& Yannuzzi, M. (2012). "EQR: A New Energy-Aware Query-Based Routing Protocol for Wireless Sensor Networks". Wired/Wireless Internet Communication. WWIC 2012. Lecture Notes in Computer Science, vol 7277.

[55] Ahvar, E., Lee, G.M., Crespi, N., & Ahvar, S. (2016). "RER: a real time energy efficient routing protocol for query-based applications in wireless sensor networks". Telecommunication Systems (2016) 61(1), 107-121.

[56]Mann, C.R., Baldwin, R.O., Kharoufeh, J.P., & Mullins, B.E. (2007). A trajectory-based selective broadcast query protocol for large-scale, high-density wireless sensor networks. Telecommunication Systems (2007) 35: 67-86.

[57] Shokrzadeh, H., & Fesharaki, M. N. (2009). "ARR: Appointment-base Rumor Routing in Wireless Sensor Networks". In proceeding of the 5th International Conference on Information & Communication Technology and Systems (ICTS).

[58] Shokrzadeh, H., Haghighat, A.T., Saadatmndi, P., &Goodarzi, M. H. (2011). "Rumor routing by appointment in center of gravity in wireless sensor networks". In proceeding of the IEEE International Conference on Information Networking (ICOIN), 177-181.

[59] Yu, C.W., Chen, R.H., Wu, T.K., & Jin, F.W. (2008). "A small-world routing protocol for wireless sensor networks". In proceeding of the IEEE 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08), 1-4.

[60] Kouassi, N.W., Djouani, K., & Kurien, A. (2013) "Performance study of an improved routing algorithm in wireless sensor networks". Procedia Computer Science, 19, 1094-1100.

[61] Gu, H. (2010). "Relative Coordinate Rumor Routing in Wireless Sensor Networks". Master tech dissertation, Tshwane University of Technology.

[62] Khelifi, M., Moussaoui, S., Silmi, S., & Benyahia, I. (2015) "Localization algorithms for wireless sensor networks: a review". International Journal of Sensor Networks, 19(2), 114-129.

[63] Han, G.,Xu, H., Duong, T.Q.,Jiang, J.,&Hara, T. (2013). “Localization algorithms of Wireless Sensor Networks: a survey”. Telecommunication Systems (2013) 52: 2419

[64] Al-Karaki, J. N.,U l-Mustafa, R.,& Kamal, A. E. (2009). “Data aggregation and routing in wireless sensor networks: Optimal and heuristic algorithms”. Computer networks, 53(7), 945-960.

[65] Chi, Y. P., & Chang, H. P. (2013) “An energy-aware grid-based routing scheme for wireless sensor networks”. Telecommunication Systems, 54(4), 405-415.

[66] Guerroumi, M., Badache, N.,&Moussaoui, S. (2015) “Mobile sink and power management for efficient data dissemination in wireless sensor networks”. Telecommunication Systems (2015) 58: 279.

[67] Min Chen & al (2006) “Mobile Agent Based Wireless Sensor Networks”, Journal of Computers, Vol. 1, No. 1, pp14-21.

[68] Min chen & al, "Mobile Agent-Based Directed Diffusion in Wireless Sensor Networks", EURASIP Journal an Advances in Signnal Processing.

[69] Arya, R., & Sharma, S. C. (2015). “WSN: Lifetime Maximization of Rumor Routing Protocol with Optimization Scheme and Bandwidth Evaluation”. British Journal of Mathematics & Computer Science, 7(4), 266-279.

[70] Khan, A. W., Bangash, J. I., Ahmed, A., & Abdullah, A. H. (2017). “QDVGDD: Query-Driven Virtual Grid based Data Dissemination for wireless sensor networks using single mobile sink”. Wireless Networks, 1-13. DOI: 10.1007/s11276-017-1552-8.

[71] Saleh, A. I., Abo-Al-Ez, K. M., & Abdullah, A. A. (2017). “A Multi-Aware Query Driven (MAQD) routing protocol for mobile wireless sensor networks based on neuro-fuzzy inference”. Journal of Network and Computer Applications, 88, 72–98.

[72] Mawloud, O., Soraya, Y.,Abdelmadjid,B. (2016) “Reliable and energy aware query-driven routing protocol for wireless sensor networks”. Annals of Telecommunications Springer, 71 (1-2), pp.73-85.

[73] Shokrzadeh. H & al "Distributed Appointment Points in Query-Driven Routing in Wireless Sensor Networks". In proceeding of the 6th IEEE International Conference on Telecommunication Systems, Services, and Applications (pp. 95-101) 2011.

[74] E. B. Hamida and G. Guelius, (2008) "A line based data dissemination protocol for wireless sensor networks with mobile sink", in proceeding of the IEEE International Conference on Communication (ICC 2008). Beijing, China.

[75] J.H.Shin, “RailRoad: Virtual Infrastructure for Data Dissemination in Wireless Sensor Networks” (2005) In Proceeding of the 2nd ACM Int. Wksp. Perf.Eval. Wireless Ad Hoc, Sensor and Ubiquitous Net.‟05, pp. 168-174.

[76] Fall, K., &Varadhan, K. (1999) “Network Simulator (NS-2) The NS Manual”. 2018-03-23. http://www. isi. edu/nsnam/ns/doc/ns.

[77] Bulusu, N., Heidemann, J.,Estrin, D.,&Tran, T. (2004) “Self-configuring localization systems: Design and experimental evaluation”. ACM Transactions on Embedded Computing Systems (TECS), 3(1), 24-60.

[78] Li, X. Y., & Wang, Y. (2011) “Complexity of data collection, aggregation, and selection for wireless sensor networks”. IEEE Transactions on Computers. 60(3), 386-399.

[79] Oliver, R., and Fohler, G (2010) “Timeliness in Wireless Sensor Networks: Common Misconceptions”. In Proceedings of the International Workshop on Real-Time Networks, July 2010.

ANNEXE1. FRA. LE PSEUDO-ALGORITHME.

```
Pour chaque noeud du réseau

{
    Node.liste Evts ← {} ;
    Node.liste des voisins ← {};
    Node.liste zone voisine ← {};
    Remplir la liste des voisins;
}
```

// Phase de création de zones

```
Si Msg_invitation reçu alors call msg_invitation _Receive (Msg_invitation);

Si Reponse_invitation reçu alors call reponse_invitation_Receive (Reponse_invitation); // À la fin de la phase de création de zones, chaque calcule l'ensemble de ces zones voisines.

Pour chaque nœud du réseau

{
    Remplir la liste des zone voisine;
}
```

// Phase de routage des agents et des requêtes

```
Repeat for ever
{
    Si le capteur détecte un évènement E      alors call event_Detect (E) ;
    Si AGENT reçu      alors call agent_Received (AGENT);
    Si le capteur s'intéresse à un évènement E alors call Search_event (E);
    Si REQUETE reçue alors call query_Received (REQUETE);
}
```

Fonction msg_invitation_Receive (Msg_invitation)

```
Début
Vérifier si j'appartiens déjà à une zone
Si (non)          alors  - Node.idzone ⇦ Msg_invitation.zone ;
        // J'invite tous mes voisins à rejoindre ma zone ;
       Pour chaque voisin
        Send (Msg_invitation) ; //l'envoi se fait via un broadcast
       Fsi
     Send (Reponse_invitation); //l'envoi se fait via un broadcast
 Fin.
```

Fonction reponse_invitation _Receive (Reponse_invitation)

```
Début
MAJ des listes de voisins
Fin.
```

Fonction Event_Detect (E)

Début

Inserer l'evt E dans la liste Evts du noeud;

Node.liste Evts (E).direction ← Node.id;

Node.liste Evts (E).distance ← 0; //Créer un agent.

AGENT.lasthop ←Node.id;

AGENT.TTL← durée de vie de l'agent;

Inserer l'evt E dans la liste Evts de l'agent;

AGENT.liste Evts (E).distance ← 1;

Créer le vecteur Historique ;

Historique (Node.idzone) ← 1 ; //zone visitée

AGENT.HistoZ ← Historique ;

Call Choose_ next_ hop1 (Node.id);

Si next_hop est un voisin interne

alors AGENT.TempLN ← (Node.liste des voisins – nexthop) + Node.id;

sinon AGENT.TempLZ ← (Node.liste des zones voisines – nexthop.idzone) ;

Fsi ;

Send (AGENT); //l'envoi se fait via un broadcast

Fin.

Fonction Agent_Received (AGENT)

```
Début
Call Synchroniser tables ();
Si (Node.id = AGENT.nexthop) alors
    TTL --;
    Si TTL =0
        alors détruire_agent ();
        sinon
        Call Choose_ next_ hop (Node.id,AGENT.HistoZ,AGENT.TempLZ,AGENT.TempLN);
        Si nexthop est un voisin interne
            alors  AGENT.TempLN ← (Node.liste des voisins – nexthop) + Node.id;
        // L'insertion des éléments dans la liste TempLN utilise le mécanisme FIFO
            sinon  AGENT.TempLZ ← (Node.liste des zones voisines – nexthop.idzone)
;
            AGENT.TempLN ← {};
        Fsi ;
        AGENT.HistoZ(Node.idzone) ← 1 ; //zone visitée
          Send (AGENT); //l'envoi se fait via un broadcaste
       Fsi;

  Fsi;

  Fin.
```

Fonction Search_event (E)

Début

//Créer une requête.

REQUETE.lasthop ⇦Node.id;

REQUETE.Evt cible ⇦ E ;

Créer le vecteur Historique ;

Historique (Node.idzone) ⇦ 1 ; //zone visitée

REQUETE.HistoZ ⇦ Historique ;

Call Choose_ next_ hop1 (Node.id);

Si next_hop est un voisin interne

alors REQUETE.TempLN ⇦ (Node.liste des voisins – nexthop) + Node.id;

sinon REQUETE.TempLZ ⇦ (Node.liste des zones voisines – nexthop.idzone) ;

Fsi ;

Send (REQUETE); //l'envoi se fait via un broadcast

Fin.

Fonction Query_Received (REQUETE)

```
Début
Si (Node.id = REQUETE.nexthop) alors
    TTL --;
      Si (REQUETE.Evt cible est trouvé)
        alors
         Si Node.liste Evts (Evt cible).distance= 0
         alors Chemin établi;
           sinon REQUETE.nexthop ⇦ Node.liste Evts(Evt cible).direction ;
         Send (REQUETE); //l'envoi se fait via un broadcast ; Fsi ;
                  Sinon

     Call Choose_ next_ hop      (Node.id,REQUETE.HistoZ,REQUETE.TempLZ,REQUETE.TempLN);
                Si nexthop est un voisin interne
                   alors REQUETE.TempLN ⇦ (Node.liste des voisins – nexthop) + Node.id;
// L'insertion des éléments dans la liste TempLN utilise le mécanisme FIFO
                    sinon
                      REQUETE.TempLZ ⇦ (Node.liste des zones voisines – nexthop.idzone)
;
                     REQUETE.TempLN ⇦{};
Fsi ;
       Fsi ;

    Fin.
```

Fonction Synchroniser tables ()

```
Début
// Synchroniser la table du nœud par rapport à celle de l'agent
Pour chaque éventement E de AGENT.liste Evts
Si (Node.liste Evts ne contient pas E)
        alors insérer (Node.liste Evts, E);
        sinon
                Si ( Node.liste Evts (E).distance > AGENT.liste Evts(E).distance)
                    alors Node.liste Evts (E).distance ↰ AGENT.liste Evts(E).distance ;
                   Node.liste Evts (E).direction ↰ AGENT.lasthop ;
Fsi ;
    Fsi ;
// Synchroniser la table de l'agent par a port celle du nœud
Pour chaque évènement E de Node.liste Evts
Si (AGENT.liste Evts ne contient pas E)
        alors insérer (AGENT.liste Evts, E);
         AGENT.liste Evts(E).distance ↰ Node.liste Evts (E).distance + 1 ;
            sinon
                AGENT.liste Evts(E).distance ↰ Node.liste Evts (E).distance + 1 ;
        Fsi ;

Fin.
```

Fonction Choose_ next_ hop1 (id)

Début

Chercher un voisin externe; //le choix du voisin se fait de manière aléatoire.

S'il existe alors return id voisin externe ; sinon return id voisin interne ;

Fsi ;

Fin.

Fonction Choose_ next_ hop (id, histo, tempz, tempn)

Début

Chercher un voisin externe appartenant à une zone non déjà visitée et sa zone ne figure pas dans tempz ;

S'il existe plusieurs voisins externes qui vérifient ces deux conditions alors choisir un de manière aléatoire ;

sinon

S'il n'existe aucun voisin externe qui vérifie les deux conditions alors

S'il existe plusieurs voisins externes appartenant à des zones figurants dans tempz alors choisir un de manière aléatoire ;

sinon choisir un voisin interne de manière aléatoire ;

Fsi ;

Fsi ;Fsi ;Fin.