UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
YOUCEF ZIANI

ÉTUDE COMPARATIVE DE MÉTHODES DE ROUTAGE DANS LES RÉSEAUX DE CAPTEURS SANS FIL POUR LE DOMAINE RÉSIDENTIEL

JUIN 2013

Résumé

Contrôler la température de sa maison et son taux d'humidité; permettre à un ingénieur en pétrochimie d'estimer le débit des gaz dans les pipelines, ou à une unité de pompiers de surveiller une zone forestière et ainsi de prévenir les feux de forêt; accorder à un médecin la possibilité de suivre l'évolution du rythme cardiaque de son patient; sont quelques exemples d'applications que les réseaux de capteurs sans fil (RCSF) permettent à l'homme du 21^{e} siècle d'accomplir à distance et de façon complètement automatique.

Le déploiement d'un RCSF est intimement lié à l'application. Dans ce projet, nous nous sommes intéressés à la problématique de maîtriser ce qui est nécessaire au développement d'un réseau de capteurs sans fil pour une application résidentielle avec des contraintes physiques liées au matériel et à l'environnement. Pour cela, nous nous sommes intéressés à l'aspect protocole de communication dans les RCSF, et spécialement aux couches MAC et Routage. Ainsi, une étude comparative a été effectuée entre plusieurs protocoles de routage dans des environnements de simulations les plus proches possible des conditions réelles. Nous avons élaboré, en partenariat avec une entreprise privée, les critères et scénarios de simulations qui pourront valider les performances des protocoles testés et formuler des recommandations pour leurs produits industriels.

Remerciements

Je tiens, en premier à remercier mon directeur le professeur Adel Omar Dahmane qui, tout au long de ma maîtrise, a su me donner le soutien moral et technique pour avancer. Je lui suis très reconnaissant aussi pour sa patience, son dévouement, et sa disponibilité. Tant sur le plan académique qu'humain, nos échanges m'ont grandement enrichi et encouragé.

Je remercie également les différents collaborateurs de l'entreprise Synapse Électronique pour leur disponibilité et leurs recommandations qui ont su donner plus de réalisme au projet.

Enfin, une pensée affective va à mes parents et à ma famille, qui m'ont toujours été d'un support essentiel.

Table des matières

Liste des tableaux

Liste des figures

Liste des symboles

σ_{dB} : Déviation du Shadowing

β : Exposant d'atténuation

ACK : Acknowledgement

AODV : Ad-Hoc On-Demand Distance Vector

AWK : Alfred Aho, Peter Weinberger, and Brian Kernighan

CBR : Constant Bit Rate

CCA : Channel Clear Assessment

CEDAR : Core Extraction Distributed Ad hoc Routing

CLR : Clear

A/N : Analogique/Numérique

CPU : Central Processing Unit

CSMA : Carrier Sense Multiple Access

CSMA/CA : Carrier Sense Multiple Access/Contention Avoidance

CTS : Clear To Send

DSDV : Destination Sequenced Distance Vector

DSR : Dynamic Source Routing

FDMA : Frequency Division Multiple Access

FFD : Full Function Device

FIFO : First In First Out

GTS : Guaranteed Time Slot

HSR : Hierarchical State Routing

Hz : Hertz

IARP : Intra-zone Routing Protocol

ID : Identité

IEEE : Institute of Electrical and Electronics Engineers

INI : Initialisation

IP : Internet Protocol

Kbps : Kilo bits par seconde

KHz : Kilo Hertz

KO : kilo-octet

LAN : Local Area Network

LL : Link Layer

LMST : Laboratoire des Microsystèmes et Télécommunications

LQI : Link Quality Indicator

m/s : Mètres/Seconde

MA : Multicast Announcement

MAC : Media Access Control

RAM : Random Access Memory

MHz : Méga Hz

ms : Milliseconde

NAM : Network Animator

NED : Network Definition

NICTA : National Information and Communications Technologies Australia

NS : Network Simulator

OSI : Open Systems Interconnection

P2P : Peer-to-Peer

PAN : Personal Area Network

PUMA : Protocol for Unified Multicasting through Announcements

QoS : Quality of Service

QRY : Query

RCSF : Réseau de Capteurs Sans Fil

RF : Radio Fréquence

RFD : Reduced Function Device

RREP : Route Response

RREQ : Route Request

RERR : Route Error

RTS : Request To Send

SINR : Signal to Interference plus Noise Ratio

SPIN : Sensor Protocol for Information via Negotiation

SYNC : Synchronisation

TCL : Tool Command Language

TCP : Transmission Control Protocol

TDMA : Time Division Multiple Access

TEEN : Threshold sensitive Energy Efficient sensor Network

TI : Texas Instruments

TORA : Temporally-Ordered Routing Algorithm

UDP : User Datagram Protocol

UPD : Update

WLAN : Wireless Local Area Network

WPAN : Wireless Personal Area Network

ZRP : Zone Routing Protocol

Chapitre 1 - Introduction

Les réseaux de capteurs sans fil occupent de plus en plus de champs d'applications dans la vie quotidienne allant du contrôle de la température et de l'humidité à l'estimation du niveau des batteries à hydrogène. Ce type de réseaux consiste en un ensemble de nœuds de capteurs sans fil agissants comme des générateurs et des relais à des données en contrôlant un phénomène physique [1]. Au-delà de leur rôle de base, les nœuds de capteurs disposant d'une unité de traitement peuvent être programmés pour effectuer des tâches supplémentaires. Partant de ces définitions, il est possible d'imaginer une infinité d'applications, jusqu'au moment où on se heurte aux différents défis que constitue le développement d'un réseau de capteurs sans fil.

Ce type de réseaux a connu sa plus grande avancée au début des années 1980 grâce au projet du Département de Défense Américain dans le programme DARPA (*Defense Advanced Projects Agency*) en utilisant les réseaux DSN (*Distributed Sensor Networks*) [2, 3]. Les DSN consistaient en un réseau de capteurs autonomes à faible coût utilisant le protocole Arpanet (prédécesseur d'Internet) comme protocole de communication. Depuis, les avancées en microélectronique et en télécommunication et l'intérêt des militaires pour ces réseaux ont contribué de façon à les rendre indispensables à une multitude d'applications.

Développer un réseau de capteurs sans fil (RCSF) pour une application donnée est une tâche qui prend en considération trois aspects interdépendants : L'application, le réseau, et la couche physique.

Un RCSF peut contenir plusieurs types de capteurs [1] dont thermique, sismique, magnétique, visuel, infrarouge, acoustique et radar. Il est possible de classer les applications des RCSF en 5 domaines : Militaire, Environnemental, Santé, Domotique, et Industriel. Dépendamment de l'application voulue, un ou plusieurs types de capteurs sont choisis pour le réseau. Le réseau se révèle être le plus important des aspects, car celui-ci définit le protocole de communication qui gèrera la génération des paquets de données, la recherche des routes, la collaboration entre les nœuds dans le routage, la méthode d'accès au canal, la gestion des conflits lorsque ce canal est occupé, et le choix des caractéristiques physiques du signal radio (modulation, fréquence,...). L'électronique du capteur, ou sa couche physique, consiste essentiellement en le module radio qui génère le signal, le processeur qui déterminera le traitement des paquets et l'application dans chaque capteur, et la partie de détection.

À supposer que la partie physique du capteur est prédéfinie, et que l'application du réseau l'est aussi, quelle serait la procédure de choix d'un protocole de communication adapté à ces contraintes physiques et d'application? Quand on parle de contraintes physiques, il est clair que cela concerne la fréquence, la consommation d'énergie, le comportement de chaque nœud dans le réseau en termes d'indépendance de prise de décision et d'auto-organisation du réseau selon la topologie, et des contraintes temporelles en terme de délai.

La procédure choisie a été d'étudier plusieurs protocoles de communication dans les RCSF, ensuite de s'intéresser de façon exhaustive à deux couches de ces protocoles, à savoir la couche MAC, et la couche de routage. Pour la couche MAC, le choix n'a pas été difficile, puisque la couche MAC du standard IEEE 802.15.4 est, d'un côté, la plus répandue dans les protocoles de communication (ZigBee, 6LoWPAN,...), et de l'autre côté, ce standard répond aux contraintes de réseau d'intérêt dans le cadre de ce projet. En effet, la couche MAC du standard IEEE 802.15.4 offre la possibilité de créer deux types de réseaux, le premier avec un coordinateur et donc une topologie étoile, et le second sans coordinateur et donc une topologie *peer-to-peer* (ou *Mesh*). Cette dernière permettra de répondre à la contrainte d'indépendance dans la décision de tous les nœuds de capteurs et ainsi d'offrir un réseau qui peut s'auto-organiser selon la topologie choisie.

Dans le cadre de ce projet, une étude comparative est effectuée entre plusieurs protocoles de routage fonctionnant sur un protocole MAC bien défini, à savoir, le standard IEEE 802.15.4. Cette étude comparative ne peut être complète qu'avec des conditions de simulation se rapprochant de la réalité. En effet, notre laboratoire est entré en partenariat avec l'entreprise Synapse Électronique pour le développement d'un projet de réseau de capteurs sans fil maillé fiable dans des environnements résidentiels. Notre réseau se compose de deux parties, la partie des capteurs et la partie des relais. Chaque groupe de capteurs forme un réseau en étoile avec son relai. Les relais communiquent entre eux ainsi qu'avec les capteurs sous forme d'un réseau ad hoc sans coordinateur afin d'acheminer l'information d'un point à un autre.

La partie du routage se distingue par une multitude de protocoles différents les uns des autres, mais qui peuvent prétendre répondre aux contraintes de notre réseau. Le choix d'un

protocole adéquat ne peut se faire qu'en simulant son comportement. En effet, plusieurs simulateurs ont été développés, plus ou moins spécialisés dans les RCSF, offrant par conséquent la possibilité de simuler les différentes couches d'un protocole de communication. Départager les différents protocoles simulés se fera à l'aide de plusieurs critères de qualité, à savoir le délai d'établissement du routage, le délai de transmission d'un paquet, le taux de réussite des paquets transmis, la complexité de l'algorithme, et la consommation énergétique.

Ce mémoire comporte six parties, la première partie consiste en l'introduction générale de ce mémoire qui va décrire la problématique, les objectifs et la méthodologie de travail dans ce projet. Par la suite, la transition se fera de façon naturelle, puisque le prochain chapitre sera une introduction aux réseaux de capteurs sans fil, où il sera question de décrire le fonctionnement d'un nœud de capteur, les domaines d'utilisation des RCSF, les différentes couches du modèle OSI dans ce genre de réseaux, et les différentes contraintes dans la modélisation mathématique de la propagation des signaux dans les environnements de déploiement des RCSF.

Le chapitre qui suivra cette introduction aux RCSF concernera les protocoles de routage. En effet, une classification de ces protocoles y sera effectuée, suivie d'une description détaillée de plusieurs protocoles qui pourraient être adéquats à ce projet. La description d'un protocole de routage donné concernera les différentes opérations qui décrivent son fonctionnement, à savoir, l'établissement des routes, la manipulation des paquets de routage (réception et émission), la coopération entre les nœuds lors du routage, la maintenance des routes,... etc.

La simulation des différents protocoles de routage sous la couche MAC du standard IEEE 802.15.4 nécessite la maitrise d'un ou plusieurs logiciels de simulation des RCSF ou des réseaux en général. Pour cela, le quatrième chapitre traitera de deux simulateurs, le premier, Castalia Simulator, est un simulateur spécialisé exclusivement dans les RCSF, et qui adopte une méthode très détaillée afin de simuler le comportement d'un nœud de capteur et de ses différents modules (radio, MAC, routage, application, batterie,...). Le second, Network Simulator 2 (NS2), est un simulateur de réseaux en général. NS2 se distingue par sa plus grande notoriété dans le milieu académique du fait de sa plus grande communauté d'utilisateurs et de contributeurs et de sa grande fidélité dans la modélisation du comportement des réseaux.

La cinquième partie de ce mémoire concernera les résultats de simulations. Les différents scénarios et les critères de qualité y seront décrits. Les résultats de simulations seront donnés par la suite. Ce chapitre se conclura par une présentation de l'émulation d'un réseau de capteurs sans fil dans le cadre d'un projet avec un partenaire industriel, et les comparaisons des simulations avec les résultats de cette émulation dans un scénario réel.

Enfin, ce document s'achèvera par une conclusion générale sur l'intégralité de ce travail, conclusion qui se positionnera comme une passerelle entre les accomplissements de ce travail et les différentes voies futures que peuvent prendre d'autres projets.

Chapitre 2 - Les réseaux de capteurs sans fil

2.1 Introduction

Le champ d'intérêt de ce projet étant les réseaux de capteurs sans fil, il est primordial d'effectuer un tour d'horizon sur les différents aspects de ce type de réseaux. Tout d'abord, ce chapitre présente une définition des différentes unités qui composent un nœud de capteur. Par la suite, une revue de littérature est effectuée montrant les différents domaines d'application des RCSF. Une description est apportée sur les différents aspects des couches du modèle OSI (*Open Systems Interconnection*) des RCSF. Ces couches permettent de développer un RCSF de l'électronique du capteur à l'application utilisateur en passant par le protocole de communication. Il est important aussi d'étudier l'environnement de déploiement d'un RCSF, pour cela, plusieurs modèles de propagation de signaux radio sont décrits dans ce chapitre.

Enfin, pour situer notre problématique dans le cadre de ce qui a été décrit, ce chapitre se conclura par une description des différentes contraintes de développement de notre réseau en termes d'environnement de déploiement, de communication, de temps et de consommation énergétique.

2.2 Qu'est-ce qu'un nœud de capteur sans fil ?

Un nœud de capteurs sans fil est composé de quatre unités essentielles : l'unité d'alimentation en énergie, l'unité de communication, l'unité de traitement, et l'unité de détection [4].

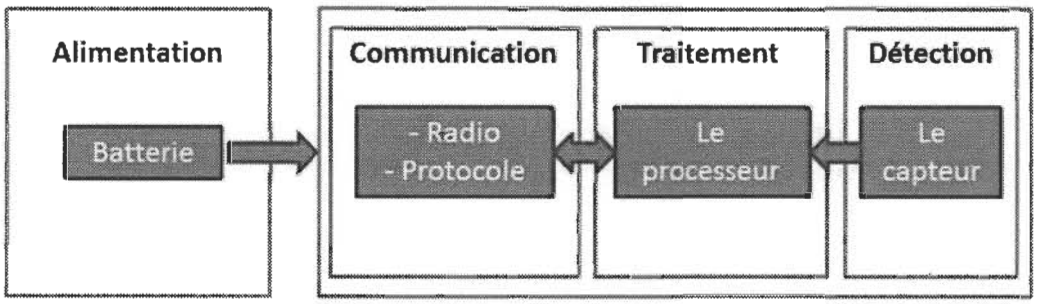

Figure 1- La composition d'un nœud de capteur

Plusieurs industriels et universités ont développé leurs propres nœuds de capteurs [5]. Le tableau suivant décrit quelques acteurs principaux de ce marché [5].

Tableau 1- Différents constructeurs de nœuds de capteurs

Nœud	**Unité de traitement**	**Module RF**	**Constructeur**
MICAz	ATmega1281	CC2420	UCB
Telos	MSP430	CC2420	Moteiv
iMote	ARM7TDMI	Bluetooth	Intel
Pluto	MSP430	CC2420	Harvard
Cicada1	MC9S08GT60	MC13193	Tsinghua
BSN Node	MSP 430	CC2420	Imperal
WINS	PXA255	802.11b	Sensoria

2.2.1 Unité de traitement

Cette unité est incluse dans le microprocesseur ou le microcontrôleur du nœud de capteur. Elle est responsable de tous les traitements des données captées par le détecteur ou reçues par la radio, que devrait effectuer un nœud de capteur. Ces microprocesseurs et microcontrôleurs pourraient inclure plusieurs fonctionnalités, à savoir, une unité de traitement, une mémoire, des interfaces de communication avec le monde extérieur (UART, USB, SPI, et I2C), et des convertisseurs A/N [4].

Il existe sur le marché plusieurs constructeurs d'unités de traitement pour les capteurs sans fil, comme le décrit le tableau suivant [5].

Tableau 2- Quelques constructeurs de processeurs pour capteurs sans fil

Constructeur	**Type**	**Processeur**	**RAM (KO)**	**Flash (KO)**
Silicon	80C51/C8051F	CIP-51	8	128
Microship	PIC18F4620	PIC	4	64
Freescale	MC9S08GT	HCS08	4	60
	MCF5222x	ColdFire	32	256
Atmel	ATMEGA128L	RISC	4	128
	AT91	ARM	256	1024
Intel	8051	MCS-51	1	16
	PXA27X	XScale	256	32
TI	MSP430F413	MSP430	10	48
Samsung	S3C44B0	ARM	8	S/O
OKI	4050/4060	ARM	16	128

2.2.2 Unité d'alimentation

Comme les nœuds de capteurs sont en technologie sans fil, ses ressources énergétiques sont très limitées. L'alimentation consiste généralement en des batteries, ou même des batteries avec une alimentation par des énergies renouvelables (photovoltaïque par exemple).

2.2.3 Unité de communication

La communication dans un RCSF se fait à l'aide de modules radio (modules RF) et par le moyen de protocoles de communication. Le module RF et le protocole de communication se trouvent dans cette unité. Les modules RF présentent certaines caractéristiques liées à la nature de l'antenne et au courant électrique généré [4].

Plusieurs constructeurs se partagent le marché des modules RF, comme le décrit le tableau suivant [5] :

Tableau 3- Quelques constructeurs de modules RF

Constructeur	**Module RF**	**Sensibilité (dBm)**
TI	CC2420	-95
	CC243x	-94
Freescale	MC1319x	-92
	MC132x	-92
Radio Pulse	MG2400	-99
Ember	EM250	-97.5
Jennic	JN513x	-97
OKI	ML7222	-90
Zensys	ZW0201	-101

2.2.4 Unité de détection

Le rôle principal d'un capteur, qui est la détection, est assuré par cette unité. La détection est liée soit à un changement d'un processus physique à contrôler, ou suite à une demande de mesure ou d'estimation. Dans les deux cas, le capteur est en mode veille jusqu'à une interruption externe. Le passage du mode veille au mode actif prend un certain temps qui est le temps de réveil. Le passage en mode veille prend aussi un certain temps. Ces deux caractéristiques temporelles différencient les capteurs les uns des autres [4].

La technologie de ces unités de détection est en fonction de l'application à laquelle est destiné ce capteur [4].

2.3 Domaines d'utilisation des réseaux de capteurs sans fil

L'utilisation d'un réseau de capteurs dans un processus industriel ou domotique facilite son contrôle et le suivi de ses paramètres. Dans le cas où ces capteurs sont en technologie sans fil, cela permettra une meilleure flexibilité du réseau et une réduction considérable de fil et de connexions inutiles. C'est pour ces deux raisons que plusieurs domaines utilisent ces réseaux pour leur fonctionnement.

- **Industries du pétrole et du gaz [6] :** dans ce domaine, les RCSF sont utilisés dans les unités de production, de raffinage, pétrochimiques, ou sous les mers pour des opérations de contrôle à distance, de maintenance, et de détection des gaz toxiques (comme le H2S) afin d'améliorer les conditions de sécurité et ainsi le rendement.
- **Militaire [7] :** Le domaine militaire utilise les RCSF pour son avancement dans les univers hostiles. Parmi les applications, on retrouve la détection et la localisation des tirs d'armes, de mortiers, et d'obus en utilisant des capteurs acoustiques,

détection des gaz chimiques et biologiques explosifs, détection des mines pour bateaux et sous-marins, ainsi que beaucoup d'autres applications.

- **Médecine [8] :** Un grand ensemble d'applications englobe les réseaux WBAN (*Wireless Body Area Network*) qui sont des capteurs placés à l'intérieur du corps humain (ECG, oxymétrie, température,...). On peut aussi les retrouver dans des applications d'aide à la chirurgie et de télémédecine.

- **Santé des structures :** Ce sont des réseaux de capteurs installés dans des structures physiques afin de prévenir les risques de détériorations. Ces structures englobent plusieurs domaines, par exemple : le génie civil (ponts [9, 10] et bâtiments [11]) et aviation [12].

- **Domotique :** Le confort et la fiabilité des environnements internes (maisons et bureaux) sont des défis majeurs dans le domaine des RCSF. Pour cela, plusieurs volets de la vie courante sont désormais automatisés. La sécurité dans ces environnements internes en prévenant et détectant les intrusions malveillantes [13] ou de fuites de gaz et d'incendies [14, 15]. La mesure de la température [16] ainsi que d'autres applications sont très ciblées par les RCSF dans l'automatisation des maisons et bureaux.

- **D'autres domaines font appel aux RCSF :** Par exemple, l'agriculture [17], la détection des feux de forêt [18],...

2.4 Le modèle OSI pour les réseaux de capteurs sans fil

Le modèle OSI pour les RCSF est décrit selon [19] par la figure ci-dessous.

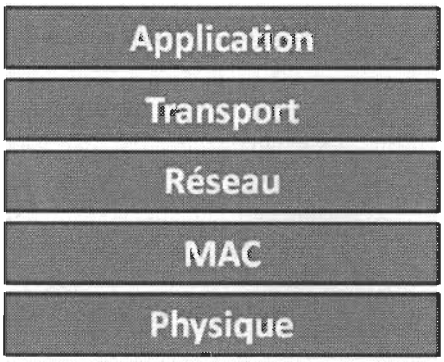

Figure 2- Modèle OSI pour les RCSF

Il existe cinq couches essentielles, la couche physique qui définit le dispositif matériel du nœud de capteur, la couche MAC pour l'accès au canal de transmission, la couche réseau pour la méthode de routage des paquets, la couche transport pour la congestion du réseau et du trafic ainsi que la détection d'erreurs, et la couche application qui définit le rôle de chaque nœud dans le RCSF.

2.4.1 Aspects de la couche Physique

La couche physique est responsable du choix de la modulation, la fréquence, du cryptage, et de la détection du signal afin de pouvoir convertir les flux binaires et de les transmettre en signaux radio sur le canal sans fil et vice versa [1].

Le standard le plus utilisé est le standard IEEE 802.15.4, qui a été développé pour les réseaux à faibles débits, faible puissance, et peu complexes [1].

Le tableau suivant présente la description de quelques modules de la couche physique existants dans le marché [1].

Tableau 4- Modules couche physique utilisés dans les RCSF sur le marché :

	RFM TR1000	Infineon TDA5250	TI CC1000	TI CC2420	Zeevo ZV4002
Plateforme	WeC, Rene, Dot, Mica	eyesIFX	Mica2Dot, Mica2, BTnode	MicaZ, TelosB, SunSPOT, Imote2	Imote, BTnode
Standard	S/O	S/O	S/O	IEEE 802.15.4	Bluetooth
Débit (kbps)	2.4-115.2	19.2	38.5	250	723.2
Modulation	OOK/ASK	ASK/FSK	FSK	O-QPSK	FHSS-GFSK
Fréquence radio (Hz)	916	868	315/433/868/915	2400	2400

2.4.1.1 La couche Physique du standard IEE802.15.4

La couche physique est responsable de [20] :

- L'activation et la désactivation du module radio.
- La détection d'énergie dans le canal.
- Estimation du LQI (*Link Quality Indicator*) pour la qualité de la liaison.
- Estimation du CCA (*Channel Clear Assessment*) pour voir si le canal est libre.
- Le choix de la fréquence.
- Transmission et réception des paquets.

Tableau 5- Quelques caractéristiques physiques du standard IEEE802.15.4

Fréquence	Modulation	Débit
868/915/2450 MHz	BPSK/ASK/O-QPSK	20-250 kbps

2.4.2 Aspects de la couche MAC

La couche MAC est responsable de l'établissement des liens entre les nœuds afin de garantir une certaine connectivité entre les nœuds [1]. L'accès au canal doit être tel que les collisions entre les transmissions sont minimisées, voire, éliminées [1]. En effet, deux nœuds voisins qui transmettent en même temps peuvent générer ce genre de phénomènes, et ainsi provoquer une collision des paquets et leur perte.

Il existe trois grandes catégories de protocoles MAC [21] basées sur la manière d'accéder au canal. Les protocoles à base de conflits d'accès, les protocoles libres de conflits, et les protocoles hybrides.

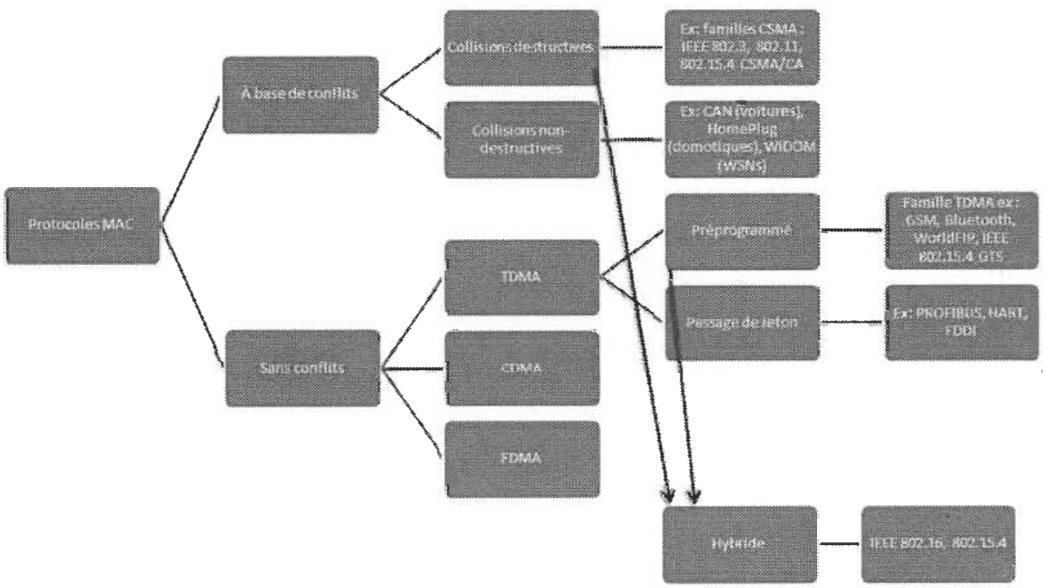

Figure 3- Classification des protocoles MAC

Tableau 6- Classification des protocoles MAC

<table>
<tr><td rowspan="2">À base de conflits</td><td>Collisions destructives</td><td colspan="2">- Les nœuds écoutent le canal, s'il est libre, ils transmettent, sinon, ils réessayent.
- Exemples : CSMA : IEEE802.3, 802.11 et 802.15.4 CSMA/CA
- Avantages : Simple et très flexible (le nombre de nœuds peut augmenter ou baisser)
- Défauts : consomme beaucoup d'énergie (une collision équivaut à une retransmission), pas de garantie de timing, et un débit limité.</td></tr>
<tr><td>Collisions non destructives</td><td colspan="2">- Chaque nœud dispose d'un identificateur.
- Algorithme pour éviter les collisions.
- Ex. : HomePlug, WiDOM,…
- Avantages : efficacité temporelle et énergétique.
- Défauts : Difficulté de synchronisation par rapport au nombre de nœuds.</td></tr>
<tr><td rowspan="2">Libre de conflits</td><td rowspan="2">TDMA</td><td>Préprogrammé</td><td>- Chaque nœud a un intervalle temporel pour la transmission
- Avantages : Efficacité énergétique et temporelle
- Défauts : non flexible</td></tr>
<tr><td>Passage de jeton</td><td>- Le nœud ayant le jeton transmet un certain temps, à la fin il transmet le jeton de façon prédéfinie à un autre nœud.
- Avantages : efficacité temporelle et énergétique.
- Défaut : peu flexible et risque de perte de jeton.</td></tr>
<tr><td>Hybride</td><td colspan="3">- CSMA+TDMA : ex. : IEEE802.16 et IEEE802.15.4
- Avantages : le meilleur des deux autres catégories.
- Défauts : Gestion complexe.</td></tr>
</table>

2.4.2.1 La couche MAC du standard IEEE802.15.4

Cette couche s'occupe de la gestion de l'accès au canal en effectuant les tâches suivantes [20] :

- Formation du réseau.
- Synchronisation du réseau par le coordinateur.
- Sécurisation des transmissions.
- L'accès au canal avec deux méthodes : à base de conflits (CSMA/CA) et préprogrammée (GTS : *Guaranteed Time Slot*).
- Gestion des communications P2P (*Peer-to-Peer*)

Le standard IEEE802.15.4 supporte l'existence de deux types de dispositifs :

- Le FFD (*Full Function Device*) : Dispositif pouvant supporter toutes les fonctionnalités du standard, et pouvant, entre autres, former et synchroniser un réseau local en étant son coordinateur (*PAN coordinator*).
- Le RFD (*Reduced Function Device*) : Dispositif possédant une partie des fonctionnalités du standard, et pouvant jouer le rôle de membre de réseau seulement.

Il existe 4 structures de paquets au niveau MAC d'un maximum de 127 octets :

- Trame de données : La trame complète contient 3 parties. Un entête pour le contrôle, le numéro de séquence, et l'adresse. Une cargaison de données provenant des couches supérieures. Une fin de séquence pour la détection d'erreurs.

- Trame « Beacon » (trame phare) : sert à la synchronisation du réseau, la définition du mode de transmission *Superframe*. Le coordinateur du PAN envoie régulièrement des Beacons et les dispositifs utilisent la réception régulière des Beacons pour tenir une base temporelle commune. De cette manière un dispositif qui n'a rien à transmettre peut se mettre en veille sachant exactement le moment de la prochaine communication. Les Beacons sont utilisés dans les transmissions indirectes, c'est-à-dire que le Beacon peut contenir des adresses de dispositifs ayant des données en attente chez le coordinateur du PAN et qu'il faut envoyer une requête pour les récupérer.

  La trame Beacon peut être utilisée pour spécifier le champ GTS. Ce champ définit des intervalles temporels de communication pour un certain nombre de dispositif. De cette manière, chaque dispositif ayant un champ GTS connait le début et la fin de sa communication.

- Trame d'acquittement (ACK) : sert à acquitter les communications, lorsque ceci est requis par un indicateur spécial.

- Trame de commande : Il existe 9 types de commandes. Voire Annexe A pour plus de détails.

Deux types de topologies sont supportés par ce standard, la topologie étoile (*Star*) et la topologie P2P [20].

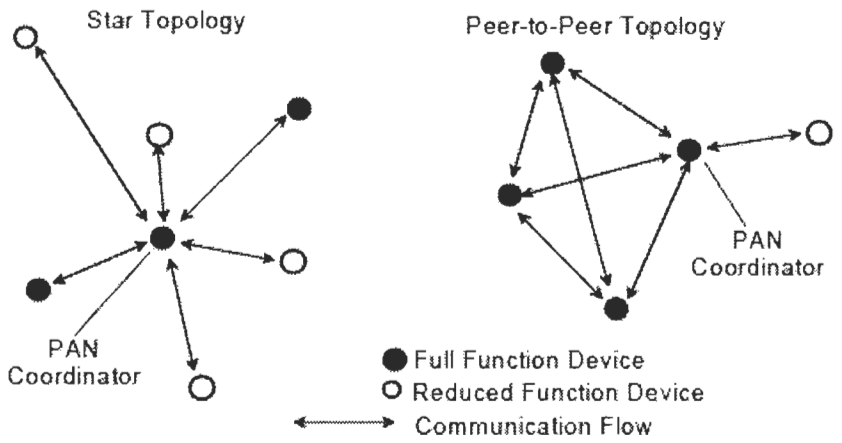

Figure 4- Exemples de topologies Étoile et P2P du standard IEEE802.15.4 [20]

2.4.2.2 Le protocole TMAC

Un autre exemple des protocoles MAC utilisés dans les simulateurs de RCSF, est le protocole TMAC (*Timestamp-ordered* MAC) [22].

Le protocole TMAC supporte quatre types de trames :

- SYNC : Trame de synchronisation.
- RTS (*Request To Send*) : Requête envoyée par l'émetteur au récepteur lui demandant l'autorisation d'envoyer son paquet DATA.
- CTS (*Clear To Send*) : Réponse positif à la requête.
- DATA : Paquet de données.

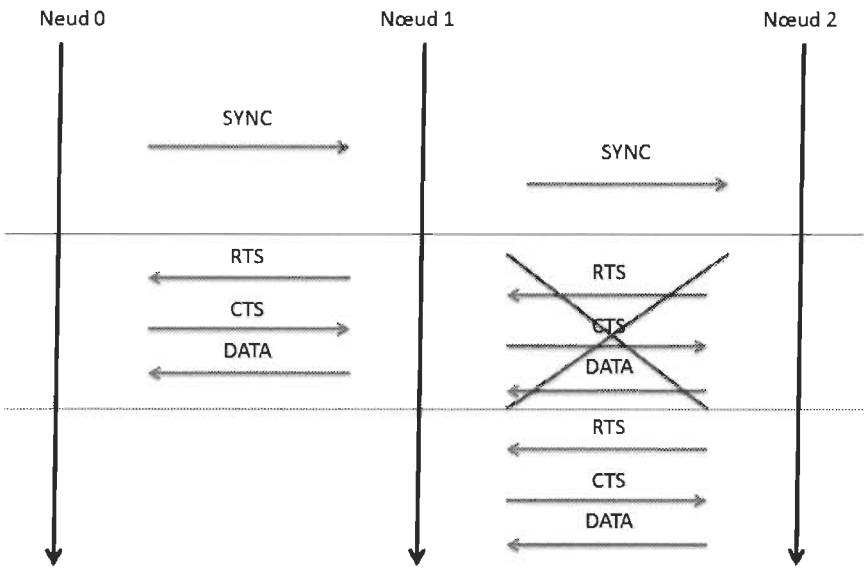

Figure 5- Déroulement des opérations dans le protocole TMAC

Selon la figure ci-dessus, un nœud commence par envoyer des synchronisations afin de définir le niveau de chaque nœud par rapport au nœud qui synchronise. Si le nœud 1 a des données à envoyer au nœud 0 et qu'en même temps le nœud 2 a des données pour le nœud 1, une sorte de compétition se met en place entre les deux nœuds (1 et 2) sur celui qui accèdera en premier au canal. Une fois qu'un des deux nœuds accède au canal (dans cet exemple, c'est le 1), suivant un choix d'implémentation par l'utilisateur [22], l'autre nœud peut, soit, réessayer tout de suite après la première tentative en espérant que l'autre communication n'a pas eu lieu, attendre un certain intervalle temporel fixé par l'utilisateur, ou bien attendre une autre trame de communication (610 ms par défaut).

Plus de détails sur l'accès au canal et la synchronisation seront donnés dans l'annexe A.

2.4.3 Aspects de la couche réseau

La couche réseau est responsable de la découverte et la gestion des routes dans un réseau quelconque. Le routage dans les RCSF rencontre plusieurs défis dus à la flexibilité de ce genre de réseau, la mobilité des nœuds et l'aspect sans fil des capteurs [1].

2.4.3.1 Les défis du routage dans les RCSF

Selon [1], cinq volets définissent les challenges du routage dans les RCSF :

- **La consommation d'énergie :** du fait des ressources énergétiques limitées dans les capteurs sans fil, la gestion de la consommation de l'énergie est le défi principal des protocoles de routage. La consommation d'énergie dans le cadre du routage peut être l'effet d'une recherche de voisinage ou d'un traitement de paquet ou de sa transmission.
- **Extensibilité :** dans les réseaux à grande densité de nœuds, l'information sur la position de chaque nœud peut être difficile à accéder. Par conséquent, il est nécessaire de développer des protocoles de routage où l'information sur la topologie n'est plus nécessaire. Les nœuds doivent donc supporter des informations de la part d'un grand nombre de nœuds sans entrave à la consommation énergétique.
- **Adressage :** les mécanismes d'adressage facilitent la communication entre les voisins. Cependant, lorsque le nombre de nœuds est important, l'information nécessite une communication multihop (plusieurs intermédiaires). Ceci provoque un phénomène de saturation dû à la juxtaposition des adresses des intermédiaire. Les protocoles de routage doivent donc supporter des mécanismes de prévention contre la saturation (*Overhead*) où l'adresse de chaque nœud n'est pas requise.

- **Robustesse :** Au cours du routage, un nœud peut subir une panne et empêcher le déroulement de ce processus. Les protocoles de routage doivent supporter des mécanismes de gestion de ce genre d'imprévus qui sont dus au canal ou aux composants du capteur, et ainsi de ne pas affecter l'efficacité du routage lors d'une perte de paquet.
- **Topologie :** le déploiement des capteurs peut se faire de façon prédéfinie ou, plus souvent de façon complétement aléatoire. Généralement, les capteurs ne connaissent pas la topologie et donc la position de leurs voisins, ce qui affecte directement les performances du routage. Les protocoles de routage doivent donc supporter des méthodes de découverte de voisinage et de son entretien afin de fournir à chaque nœud une connaissance de la topologie qui l'entoure; surtout s'il existe des nœuds mobiles, ce qui est le cas pour certains réseaux sans fil.
- **Application :** le type de la couche d'application influence directement le choix du protocole de routage. Dans les applications de contrôle d'un processus quelconque, les capteurs transmettent des informations à la destination de façon périodique, ce qui nécessite généralement des routes statiques. Dans les applications basées sur les évènements, le capteur est souvent en mode veille. À l'arrivée du premier évènement, une route doit être établie pour délivrer l'information.

2.4.3.2 Classification des protocoles de routage

Il existe quatre catégories de protocoles de routage : Les protocoles centrés sur les données d'architecture plate, les protocoles hiérarchiques, les protocoles de routage géographique, et les protocoles basés sur le QoS [1].

a- Les protocoles d'architecture plate

La donnée est transmise selon son contenu et non pas selon l'adresse de la destination. Si la requête consiste à effectuer une certaine tâche, les nœuds ayant effectués ce genre de requête (par exemple, lire une température) sont adressés par cette requête [15].

Quand une source transmet un paquet vers une destination, les nœuds intermédiaires peuvent effectuer un certain regroupage de plusieurs paquets afin de minimiser la consommation d'énergie [18].

Plusieurs types de protocoles dans cette catégorie existent [15]: Inondation, Rumeur, SPIN, et la diffusion directe.

b- Les protocoles hiérarchiques

Les nœuds y sont groupés dans des ensembles contrôlés par des têtes d'ensemble [15, 18]. La formation des ensembles se fait sur la base de la réserve d'énergie et de la distance de chaque nœud par rapport à la tête d'ensemble [18].

c- Les protocoles de routage géographique

Ces protocoles utilisent l'information géographique de chaque nœud pour un routage efficace [15, 18].

d- Les protocoles basés sur le QoS

Ces protocoles prennent en compte, non seulement, la consommation de l'énergie comme critère de choix de routes, mais aussi le délai de transmission comme autre aspect [15, 18].

2.4.4 Aspects de la couche de transport

La couche de transport est responsable du contrôle de la congestion, de délivrer les informations reçues par le nœud à l'application de façon efficace, et de gérer l'existence de plusieurs applications dans un même nœud [1]. D'une autre part, le contrôle du flux de données est aussi une des responsabilités de cette couche [23]. En effet, l'émetteur ne peut pas surcharger le récepteur de paquets plus que ce qu'il peut en recevoir. Le contrôle du flux va de pair avec le contrôle de la congestion, car le premier empêche la surcharge du récepteur, et le second s'occupe de la surcharge du réseau [1, 23].

La détection d'erreurs doit être supportée par n'importe quel protocole de communication qui se veut performant. La détection de paquets manquants ou perdus se fait généralement grâce au numéro de séquence [23], et cela se passe au niveau MAC. Quant à la détection de paquets corrompus, la couche transport doit supporter des mécanismes de détection de ce genre d'erreurs, généralement, par le moyen de retransmissions [23].

Deux des protocoles les plus connus de la couche de transport d'un réseau quelconque sont [23, 24] :

- Le protocole UDP (*User Datagram Protocol*).
- Le protocole TCP (*Transmission Control Protocol*).

L'utilisation de ces deux protocoles dans les RCSF est assujettie à plusieurs conditions, par exemple, la fiabilité, l'efficacité énergétique, et la simplicité [23], qui sont des caractéristiques propres aux RCSF.

2.4.5 Aspects de la couche d'application

La couche d'application offre à l'utilisateur une interface d'interaction avec le nœud et, par conséquent, le processus physique à contrôler avec le RCSF [1]. Il existe trois catégories de protocoles de la couche d'application [1].

- La compression de données : l'information à transmettre par l'émetteur est compressée afin de réduire sa taille.
- Le traitement de requêtes : cela revient à traiter les requêtes envoyées par un autre nœud afin d'effectuer une tâche spécifique.
- La gestion du réseau : offrir à l'utilisateur une interface simple pour interagir avec les données collectées, gérer les changements de topologie, adapter les routes et le protocole, gestion du trafic,…

2.5 Environnements de déploiement des RCSF

Le déploiement des RCSF se fait dans des environnements multiples et différents, dépendamment de l'application à réaliser. Les communications radio sont directement influencées par les objets et obstacles contenus dans cet environnement, ainsi que par l'architecture des lieux si l'on parle d'environnements internes (maison, bureaux, usine,…).

Plusieurs modèles de propagation des signaux radio ont été proposés dans la littérature, essayant de prédire l'influence de l'environnement sur le signal radio, en particulier, l'estimation de la puissance à la réception à une certaine distance.

Dans ce projet, trois modèles de propagation sont considérés, qui sont par ailleurs, proposés dans le logiciel de simulation NS2 [25].

2.5.1 Le modèle espace libre

Ce modèle suppose les conditions de propagation comme étant idéales, et que le rayon de propagation du signal radio est sous forme de disque, à l'intérieur duquel la réception est parfaite, et qu'au-delà plus aucune communication n'est possible.

La puissance à la réception à une distance d est donnée par l'équation suivante :

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{1}$$

Où, P_t est la puissance de transmission, G_t et G_r sont les gains respectifs de l'émetteur et du récepteur, λ est la longueur d'onde, et L est un coefficient de perte du système (généralement, égale à 1).

2.5.2 Le modèle Two-Ray Ground

Ce modèle considère le chemin direct de la propagation en plus de la réflexion provoquée par le sol, comme le montre la figure ci-dessous :

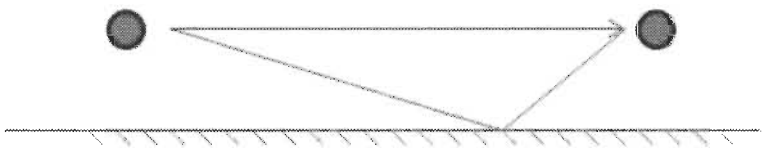

Figure 6- Le modèle de propagation Two-Ray Ground

La puissance à la réception y est décrite par l'équation suivante :

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \tag{2}$$

L'équation ne dépend plus de la longueur d'onde (et donc de la fréquence), mais dépend maintenant des hauteurs respectives de l'émetteur et du récepteur (h_t et h_r).

Le rayon de communication est toujours considéré comme un disque parfait, ce qui est loin de la réalité.

2.5.3 Le modèle Shadowing

Ce modèle est le plus proche de la réalité, puisque plusieurs phénomènes de propagation y sont pris en considération, à savoir, la réflexion, la diffusion et l'absorption. En plus, le rayon de communication n'est plus considéré comme un disque parfait.

La puissance à la réception à une distance *d*, est donnée par l'équation suivante :

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta \log(\frac{d}{d_0}) + X_{dB} \qquad (3)$$

Où, X_{dB} est une variable aléatoire de moyenne nulle et d'écart-type σ_{dB}, qui est appelée déviation du *Shadowing*, et obtenue par mesures. β est appelé exposant de l'atténuation.

Voici quelques valeurs typiques du σ_{dB} et du β :

Tableau 7- Valeurs typiques de l'exposant d'atténuation

Environnement		β
Externe	Espace libre	2
	Espace urbain	2.7-5
Interne	Visibilité directe	1.6-1.8
	Visibilité obstruée	4-6

Tableau 8- Valeurs typiques de la déviation du *Shadowing*

Environnement	σ_{dB}
Externe	4-12
Bureau, séparations fixes	7
Bureau, séparations flexibles	9.6
Usine, visibilité directe	3-6
Usine, visibilité obstruée	6.8

2.6 Description du système à l'étude

Dans le cadre de ce projet, le laboratoire LMST est entré en partenariat avec l'entreprise Synapse Électronique [26] pour le développement d'un projet de réseau de capteurs sans fil maillé fiable dans des environnements résidentiels.

L'environnement de simulation du projet peut être décrit par la figure ci-dessous. Le réseau se compose de deux parties, la partie des capteurs et la partie des relais. Chaque groupe de capteurs forme un réseau en étoile avec son relai. Les relais communiquent entre eux comme un réseau ad hoc afin d'acheminer l'information d'un point à un autre, et bien évidemment, en plus de leur rôle de relai, ceux-ci effectuent leurs tâches respectives. En effet, un relai peut être aussi un :

- Thermostat de ligne
- Thermostat 24 volts
- Déshumidistat
- Télécommande
- Contrôle de puissance intégré dans une plinthe électrique
- ...

Il existe 5 types de capteurs dans ce réseau : capteurs d'humidité, de température, de pression, de débit d'air, et de vibration.

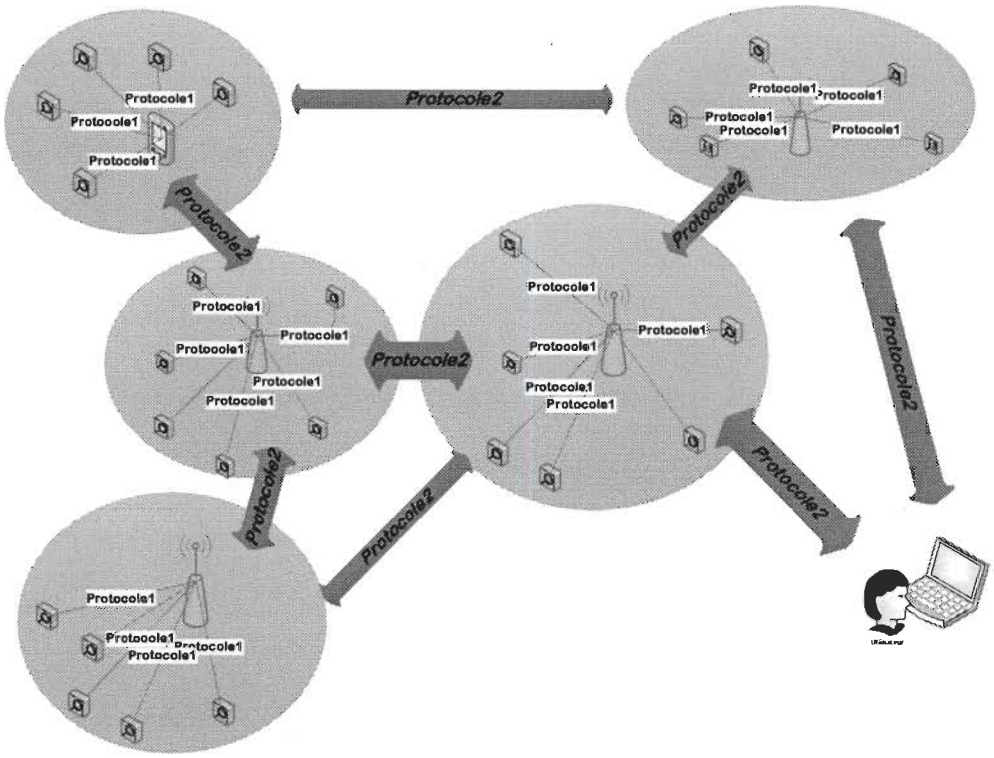

Figure 7- Description globale du réseau de Synapse

Le projet doit tenir compte de trois contraintes principales

a- Contraintes de communication

- Les organes de contrôle forment un réseau mesh sans coordinateur PAN. Donc le réseau est de type pear-to-pear avec stratégie de routage intelligent.
- Un capteur ne reçoit pas de commande. Il émet une information selon un changement d'état ou en fonction d'une pré-configuration ou une périodicité.
    - Il est tout le temps en mode *idle* sauf pour émettre une information.
    - Après avoir transmis une information, il peut demander s'il y a autre chose à faire (principe de l'acquittement).
- Faible débit de transmission.
- Les capteurs échangent avec une tour (une dizaine de capteurs/tour).

- Types de tours (relai)
    - Fixe
    - Mobile
- Stabilité du routage.

b- Contraintes de temps

Minimiser le temps nécessaire pour effectuer un routage, à savoir la recherche de la route et la transmission des paquets. À ce propos, il faut garantir un délai maximum de 500 ms pour les paquets non prioritaires et 10 ms pour les paquets prioritaires.

c- Contraintes énergétiques

- Minimiser le nombre d'évènements (un capteur doit rester le plus longtemps en mode veille)
- Le réseau doit supporter l'éventualité qu'un nœud quitte le réseau pour cause de perte d'alimentation.

2.7 Conclusion

Dans ce chapitre, une initiation aux RCSF a été faite afin de bien décrire les différents aspects d'étude de ce type de réseaux dans le cadre du développement d'une application donnée. Désormais, il est clair que le développement de ce genre de réseaux prend en considération ce qu'on peut classer en trois domaines différents et interdépendants, à savoir, l'électronique des nœuds de capteurs (couche physique), les protocoles de communication (MAC, Routage, éventuellement Transport, et Application), et l'environnement de déploiement.

Les contraintes de ce projet imposent une couche physique, une application, et un environnement de déploiement connus. Le protocole MAC étant spéciofié, plusieurs protocoles de routage ont été développés. Le chapitre 3 traite ainsi des protocoles de routage qui pourraient éventuellement satisfaire aux contraintes du réseau.

Chapitre 3 - Les protocoles de routage

3.1 Introduction

Les réseaux de capteurs sans fil ont été développés pour interagir avec l'environnement externe et d'acheminer cette interaction à un utilisateur ou une station de base. Cette spécificité permet de classer ce genre de réseaux parmi les réseaux centralisés qui sont gérés par un coordinateur du réseau [27]. Certaines applications modernes nécessitent du réseau d'être décentralisé, d'être capable de se déployer dans des environnements inconnus et de s'auto-organiser pour les opérations de routage sans recourir à une station de base. Ces réseaux sont appelés les réseaux ad hoc. Plusieurs protocoles de routage des réseaux ad hoc ont été adaptés aux RCSF qui se caractérisent par des ressources limitées en termes de mémoire et d'énergie.

3.2 Les protocoles de routage utilisés dans les RCSF

Il existe deux grandes catégories de protocoles de routage utilisés dans les RCSF. Les protocoles spécifiques aux RCSF et les protocoles des réseaux ad hoc adaptés aux RCSF.

3.2.1 Les protocoles propres aux RCSF

Les RCSF se caractérisent par leurs ressources limitées en mémoire et en énergie [28, 29]. La consommation d'énergie peut augmenter de façon drastique lorsque le nombre d'évènements du protocole de routage est important. La table de routage, et donc la mémoire utilisée, peut augmenter dans le cas de réseaux denses.

Plusieurs protocoles de routage ont été développés pour garantir l'efficacité des RCSF en termes de consommation énergétique et de mémoire [30].

Le routage par inondation [1] est un des algorithmes de routage les plus simples. L'information est envoyée en broadcast à travers tout le réseau jusqu'à ce que celle-ci atteigne sa destination. Hormis le fait de sa simplicité, ce protocole peut générer plusieurs problèmes de redondance, de chevauchement, et de consommation aveugle des ressources.

Le routage par rumeur [1] essaie d'éviter les problèmes de redondance du routage par inondation, en sélectionnant un nœud du voisinage comme relai de façon aléatoire. Les nœuds gardent aussi une copie de chaque paquet envoyé afin d'éviter la redondance, ce qui augmente les délais de transmission et surtout la consommation des ressources.

Le protocole SPIN (*Sensor Protocol for Information via Negotiation*) [1, 28-30] utilise des techniques de négociation afin d'éliminer les problèmes de redondance de données dans le routage. Chaque nœud effectue un suivi de la consommation de ses ressources ce qui influence ses décisions lors des négociations. Les négociations se font à travers des paquets d'avertissement, de requête et de données [1].

D'autres protocoles ont été développés spécialement pour les RCSF, comme la diffusion directe, PEGASIS, TEEN,… [30].

3.2.2 Les protocoles de routage ad hoc adaptés aux RCSF

Les réseaux ad hoc sont des réseaux auto-organisés et s'adaptent aux changements de la topologie et à la mobilité des nœuds [28]. Les protocoles de routage ad hoc peuvent être classés en trois grandes catégories [31]; les protocoles réactifs, proactifs et hybrides.

3.2.2.1 Les protocoles réactifs

Ce sont des protocoles qui calculent la route sur demande avant d'effectuer le routage, et n'ont pas besoin de connaitre la topologie du réseau ou d'échanger périodiquement des informations sur le routage. L'aspect « sur demande » du routage élimine la nécessité de mettre à jour la route, mais augmente le délai de démarrage du routage à cause du temps de découverte de la route [32].

Plusieurs protocoles ont été développés dans cette catégorie, on peut citer l'AODV, le DSR, TORA,... [31, 32].

3.2.2.2 Les protocoles proactifs

Les protocoles proactifs effectuent un calcul de toutes les routes possibles avant d'effectuer le routage. Les nœuds maintiennent une information sur la topologie du réseau sous forme de tables de routage, et ce, de façon périodique ou suite à un évènement [32]. Ce genre de protocoles consomme beaucoup de ressources du réseau, du fait de la connaissance préalable de la topologie.

Il existe plusieurs protocoles proactifs, tels que, le TBRPF (Annexe D), DSDV et le HSR [31, 32].

3.2.2.3 Les protocoles hybrides

À chaque fois qu'un nœud a besoin de router une information, d'abords, calcule toutes les routes possibles avec la méthode proactive ensuite s'adapte pendant le routage avec la méthode réactive [31].

ZRP, CEDAR, et IARP sont quelques exemple des protocoles hybrides dans les réseaux ad hoc [31, 32].

3.3 Le protocole de routage AODV

3.3.1 Introduction au protocole AODV

Le protocole de routage AODV (*Ad-Hoc On-Demand Distance Vector*) [33] est destiné aux réseaux ad hoc à nœuds mobiles. C'est un protocole multihop, adaptatif et dynamique par rapport aux conditions de la topologie, et permet d'éviter les situations de boucle fermée dans le routage.

Ce protocole utilise trois types de paquets, RREQ (*Route Request*), RREP (*Route Response*), et RERR (*Route Error*). Ces messages sont reçus à travers le protocole UDP en allouant une adresse IP à chaque nœud.

Quand une source cherche une route vers une destination, une requête RREQ est envoyée en broadcast à tout le voisinage. Chaque nœud recevant cette requête envoie de nouveau cette requête jusqu'à ce que celle-ci atteigne sa destination. La route est validée en renvoyant une réponse RREP via le chemin inverse de la requête jusqu'à sa source. Chaque nœud recevant la requête sauvegarde dans sa table l'adresse du hop précédent. Le hop suivant est reconnu dès la réception de la réponse.

Quand un lien est reconnu « rompu » avec un nœud, un message d'erreur est envoyé aux autres nœuds pour les prévenir que la route vers les destinations qui ne sont plus disponibles. Une réparation de la route est alors effectuée, soit par la source de la requête ou bien par le nœud se trouvant au niveau du lien brisé.

Le protocole AODV utilise une table de routage qui a les entrées suivantes :

- L'adresse IP de destination
- Le numéro de séquence de la destination
- Indicateur de validité du numéro de la séquence
- Autres indicateurs de la route (valide, invalide, réparable, en réparation)
- Interface réseau
- Compteur de hops
- Prochain hop
- Liste des précurseurs
- La durée de vie de la route (avant son expiration et son annulation)

3.3.2 Mécanismes de création des routes

Cette section décrit les scénarios selon lesquels les nœuds génèrent les différents paquets : RREQ, RREP et RRER.

3.3.2.1 Le maintien du numéro de séquence

Chaque table de routage doit inclure la dernière information sur le numéro de séquence pour l'adresse IP de la destination pour laquelle la table de routage est maintenue. Cette information est remise-à-jour si le nœud reçoit une nouvelle indication de changement du numéro de la séquence de la destination. L'indication se fait à travers les paquets reçus (RREQ, RREP et RRER). Une destination incrémente son propre numéro de séquence dans les deux cas suivants :

- Juste avant qu'un nœud ait entrepris une découverte de route.

- Juste avant qu'une destination ait généré une RREP en réponse à une RREQ. Le nœud doit remettre-à-jour son propre numéro de séquence au maximum entre le numéro de séquence actuel et celui contenu dans le paquet RREQ.

Un nœud peut changer le numéro de séquence dans la table d'entrée de la destination dans les cas suivants :

- Si ce nœud est la destination elle-même, ou
- si ce nœud reçoit un message AODV avec de nouvelles informations sur le numéro de séquence de la destination, ou
- si le chemin menant à cette destination arrive à expiration ou est rompu.

3.3.2.2 Les entrées de la table de routage et la liste des précurseurs

Dans le cas où un nœud reçoit un paquet AODV de son voisin, ou crée ou remet à jour la route vers une destination particulière, celui-là recherche dans la table de routage l'adresse de la destination. Si aucune entrée ne correspond à cette destination, une entrée est alors créée dans la table de routage contenant le numéro de séquence du paquet AODV. La route est modifiée si le numéro de séquence est :

- supérieur à celui de la destination, ou
- égale à celui de la destination, mais le nouveau compteur de hop plus un est inférieur au compteur de hop actuel dans la table de routage, ou
- inconnu.

Le champ de durée de vie de la table de routage est aussi une entrée de la table de routage. Ce paramètre est le temps d'expiration de la route valide et le temps de suppression de la route invalide.

Pour chaque route valide maintenue par le nœud sous la forme d'entrée à la table de routage, ce nœud maintient aussi une liste de précurseurs qui peuvent router les paquets sur cette route. Les précurseurs sont ceux qui ont déjà acheminé une réponse RREP vers ce nœud.

3.3.2.3 Génération et acheminement des requêtes RREQ

La requête est générée par un nœud si celui-là ne connait pas la destination ou qu'une route préalablement valide a été invalidée ou a expiré. Le numéro de séquence de destination dans la RREQ est le dernier numéro de séquence de destination connu, sinon un indicateur de numéro de séquence inconnu est activé.

Avant la transmission (en Broadcast) du paquet, la source sauvegarde le RREQ ID et l'adresse IP de la source (sa propre adresse) pour l'instant actuel. De cette manière, si la source reçoit le même paquet, celle-ci ne le retransmet pas. La source ne devrait pas générer plus qu'un maximum de paquets RREQ générés (RREQ_RATELIMIT) par seconde. Si une route n'est pas reçue en un temps inférieur à NET_TRAVERSAL_TIME millisecondes, la source peut générer un nouveau RREQ jusqu'à un maximum de retransmissions RREQ_RETRIES.

Les paquets de DATA en attente d'une route (en attente d'une RREP après la génération d'une requête RREQ) doivent être bufférisés en mode FIFO. Si le RREQ_RATELIMIT pour une certaine destination est atteint, tous les paquets de DATA qui lui sont destinés doivent être rejetés.

Quand un nœud reçoit une requête, celui-ci crée ou remet à jour le hop précédent si celui-là n'a pas de numéro de séquence valide dans sa table de routage. Par la suit, le nœud

vérifie s'il a déjà reçu le paquet RREQ avec la même adresse IP source et le même RREQ ID dans le dernier intervalle PATH_DESCOVERY_TIME.

Si une requête est acceptée, le compteur de hops est incrémenté d'une unité. Ensuite, le nœud établit la route inverse vers la source de la requête. Cette route sera nécessaire si ce nœud doit acheminer une RREP vers la source de la requête. Quand une route est créée, les actions suivantes sont entreprises :

- Le numéro de séquence de la source de la RREQ est comparé au numéro de séquence de la destination contenu dans la table de routage et y est copié si le premier est supérieur au deuxième. Le nœud qui fait le routage de la RREQ ne change pas son propre numéro de séquence de la même destination, même si celui-là vérifie la condition précédente.
- L'indicateur numéro de séquence valide est activé (vrai).
- Le prochain hop dans la table de routage devient le nœud à partir duquel le RREQ a été reçu (pas nécessairement la source originale de cette requête, mais probablement un nœud intermédiaire).
- Le compteur de hops est remplacé par celui contenu dans le message RREQ.
- Le champ de durée de vie de la route est remis à jour.

Si le nœud recevant la requête n'est pas supposé générer une réponse, celui-ci incrémente le compteur de hops par une unité et broadcast le requête. Si le nœud génère une réponse RREP, la RREQ est supprimée.

3.3.2.4 Génération et acheminement des réponses RREP

Un nœud génère une RREP si :

- Ce nœud est la destination de la requête, ou
- Ce nœud dispose d'une route active vers la destination de la requête ; le numéro de séquence de la destination existant (de la route que le nœud connait déjà) est valide et supérieur ou égale au numéro de séquence de la destination contenu dans le RREQ ; et que la requête n'est pas transmise exclusivement à la destination (à travers l'indicateur « *destination only* »).

La réponse RREP est générée en copiant les adresses IP de la destination et de la source dans les champs équivalents. L'envoi de cette réponse se fait en mode unicast vers la destination qui était la source précédente (hop précédent) de la requête selon la table de routage. Le compteur de hops continu d'être incrémenté suivant le chemin inverse suivi par la réponse jusqu'à la source originale de la requête.

Chaque nœud intermédiaire recevant la réponse RREP place dans sa liste de précurseurs le nœud à partir duquel il vient de recevoir ce paquet. La route directe est donc établie à travers l'acheminement de la réponse ; sachant que la requête avait servi à établir le chemin inverse. Sachant aussi que si la réponse n'atteint pas la destination (qui est la source de la requête), celle-ci est déjà en train d'attendre NET_TRAVERSAL_TIME millisecondes avant d'entreprendre une nouvelle découverte de route et de retransmettre la requête ou d'en générer une nouvelle.

3.3.2.5 Les messages « Hello »

Les messages HELLO permettent d'offrir des informations sur la connectivité entre les nœuds. A chaque HELLO_INTERVAL millisecondes, chaque nœud vérifie s'il a envoyé au moins une requête RREQ ou un message MAC.

Les nœuds peuvent aussi écouter le canal pour vérifier si leurs voisins reçoivent des messages (Hello ou autres). Si dans un certain intervalle, un voisin ne présente aucune activité de ce genre, le lien avec celui-ci est considéré comme rompu.

3.3.3 Mécanismes de maintenance des routes

3.3.3.1 Maintien de la connectivité locale

Chaque nœud devrait s'assurer de sa connectivité avec son voisinage (ses prochains hops), que ce soit avec ceux qui participent à l'acheminement des paquets durant le dernier ACTIVE_ROUTE_TIMEOUT, ou bien avec ceux qui transmettent des messages Hello durant le dernier intervalle de réception de ce genre de messages donné par ALLOWED_HELLO_LOSS * HELLO_INTERVAL.

3.3.3.2 Génération et acheminement des erreurs RERR

Les messages d'erreurs sont générés selon le processus suivant :

- Si le nœud détecte qu'un lien est rompu avec le prochain hop dans une route active ou dans sa table de routage lors du transfert de données ou qu'une opération de réparation a échoué, ou
- si ce nœud reçoit un paquet de données qui lui est parvenu d'une destination inconnue ou que ce nœud n'est pas en train d'effectuer une réparation de route, ou
- si ce nœud reçoit un message RERR d'un voisin pour une ou plusieurs routes.

Pour le premier cas, le nœud établit une liste de destinations injoignables, qui sont les voisins injoignables et toute autre destination contenue dans la table de routage utilisant ces voisins injoignables comme prochain hop.

Pour le deuxième cas, il existe une seule destination injoignable. Pour le troisième cas, le nœud établit la liste des destinations injoignables à partir du paquet RERR.

Avant de transmettre un RERR, le nœud doit apporter des mises-à-jour nécessaires à la table de routage concernant les numéros de séquences des destinations injoignables :

- Le numéro de séquence de la destination injoignable est incrémenté dans les deux premiers cas de création du paquet RERR. Sinon, dans le troisième cas, le numéro est copié à partir de RERR reçu.
- L'entrée est invalidée en marquant la route comme invalide.
- Le champ de durée de vie de la route est remis-à-jour (dans ce cas, ce champ joue le rôle du temps de suppression de la route, comme indiqué auparavant).

3.3.4 Caractéristiques supplémentaires

Voici la liste de quelques paramètres importants dans les opérations du protocole AODV et leur valeur par défaut dans le tableau suivant.

Tableau 9- Quelques valeurs par défaut du protocole AODV

Paramètres	Valeur par défaut
ACTIVE_ROUTE_TIMEOUT	3,000 Millisecondes
ALLOWED_HELLO_LOSS	2
HELLO_INTERVAL	1,000 Millisecondes
NET_TRAVERSAL_TIME	2 * NODE_TRAVERSAL_TIME * NET_DIAMETER
NODE_TRAVERSAL_TIME	40 Millisecondes
PATH_DISCOVERY_TIME	2 * NET_TRAVERSAL_TIME
RERR_RATELIMIT	10
RREQ_RETRIES	2
RREQ_RATELIMIT	10

3.4 Le protocole de routage DSR

3.4.1 Introduction au protocole DSR

Le protocole de routage DSR [34] a été développé pour les réseaux sans fil multi hops. Ce protocole permet au réseau de s'organiser et de se configurer par lui-même lors des changements de topologie (mobilité, défaillance des nœuds,…).

DSR est un protocole réactif, ce qui veut dire que la route n'est pas connue au préalable et que l'émetteur doit entreprendre une découverte de route avant d'envoyer son paquet. En plus du mécanisme de découverte de route, DSR offre la possibilité de réparer les routes

défectueuses qui sont dues aux changements de topologie à l'aide d'un mécanisme de maintenance de route.

3.4.2 Mécanisme de création des routes

Si une source désire envoyer un paquet à une destination, celle-ci commence par inspecter sa table de routage pour vérifier l'existence d'une route vers cette destination. Si la route n'existe pas, le mécanisme de découverte de route est entrepris. Dans le protocole DSR, l'initiateur est le nom de la source, la cible est celui de la destination.

La figure ci-dessous représente un exemple de découverte de route d'un initiateur A à une cible E.

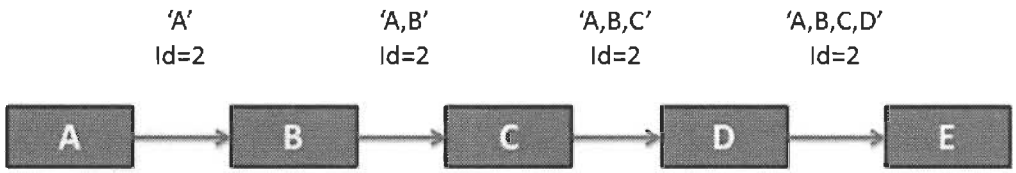

Figure 8- Exemple de découverte de route du protocole DSR

L'initiateur 'A' commence par envoyer une requête (*Request*) à son voisinage en broadcast. Cette requête inclut les adresses de la source et de la destination, ainsi que l'identificateur de cette requête (id=2). L'entête de la requête est chargé par les adresses des nœuds intermédiaires qui font le relai. Dans cet exemple, lorsque le nœud B reçoit la requête, il vérifie s'il n'est pas la destination de cette requête, ce qui n'est pas le cas, alors, il charge l'entête de la requête avec son adresse (B) et l'envoi en broadcast à son voisinage, et ainsi de suite. Lorsque le nœud E reçoit le requête et vérifie qu'il est bien la cible, une réponse (*Reply*) est alors transmise à l'initiateur A suivant la route inverse.

Un nœud qui génère un paquet doit vérifier d'abords l'existence d'une route vers la destination de ce paquet, sinon, une découverte de route est entreprise.

La découverte de route utilise deux types de paquets, les requêtes et les réponses. La maintenance de la route utilise, quant à elle, les erreurs et les acquittements.

3.4.2.1 Les requêtes

L'initiateur de la requête doit établir l'entête d'option DSR. Dans ce cas, tous les champs de l'option de requête de route doivent être initialisés, dont, le type d'option (dans ce cas, avec la valeur 2), l'adresse de la cible de cette requête, et l'adresse de la prochaine destination de cette requête (dans ce cas, c'est un broadcast).

Un nœud qui reçoit un paquet contenant une option de requête de route doit vérifier si l'adresse de cible correspond à son adresse. Si c'est le cas, le nœud doit générer une réponse de route à l'initiateur de la requête. Sinon, le nœud examine la liste des adresses de la route contenue dans cette requête afin de vérifier si son adresse apparait, dans ce cas, la requête est ignorée. Si son adresse n'apparait pas dans cette liste, le nœud doit vérifier certaines conditions de sécurité comme décrit dans [34]. Par la suite, le nœud vérifie s'il a déjà reçu ce paquet, si c'est le cas, la requête est ignorée. Sinon, le nœud effectue certaines tâches sur ce paquet avant de le retransmettre :

- Ajoute une entrée dans sa table de routage pour ce paquet (identificateur, cible) afin de pouvoir le reconnaitre dans le futur, s'il le reçoit encore une fois.
- Ajoute son adresse à la liste des nœuds ayant relayés cette requête.

- Le nœud vérifie s'il a déjà une route pour la cible, si c'est le cas, il génère une réponse de route à l'initiateur, sinon, ce nœud doit transmettre la requête en broadcast.

3.4.2.2 Les réponses

La réponse de route peut être générée soit par la cible de la requête, ou bien par un nœud intermédiaire qui dispose déjà d'une route vers la cible, la requête n'est alors pas retransmise.

Un nœud qui génère une réponse de route doit d'abord initialiser l'entête d'option de réponse de route en mettant, par exemple, le champ du type d'option à la valeur 3 (relatif à l'option de réponse) et le champ des adresses des relais de la requête.

Le protocole DSR offre une solution à un problème de congestion du canal qui peut survenir dans le cas où plusieurs nœuds recevant une requête disposent dans leurs tables de routage respectives un chemin vers la cible de la requête. En effet, ces nœuds pourraient envoyer simultanément une réponse de route à l'initiateur de la requête, ce qui risque de causer des collisions entre les paquets. Pour cela, DSR introduit un délai aléatoire à chaque nœud avant d'envoyer ce type de réponse.

3.4.2.3 Les acquittements

Le protocole DSR offre trois types d'acquittements : les acquittements de la couche de liaison (MAC), les acquittements passifs, et les acquittements de la couche réseau. Si le protocole MAC utilisé supporte les acquittements des données, alors les autres acquittements ne sont pas nécessaires pour la maintenance de route. Si les deux premiers

acquittements ne sont pas disponibles, le nœud qui envoie un paquet à un autre nœud doit insérer dans l'entête d'option une requête d'acquittement de la couche réseau.

3.4.3 Mécanisme de maintenance des routes

Un nœud assimile qu'un lien est rompu avec un de ses voisins, lorsque celui-ci n'acquitte pas un certain nombre de communications. L'acquittement peut se faire avec un paquet dédié, ou bien à l'aide d'un moyen passif. Comme dans la figure précédente, l'acquittement passif entre C et D, par exemple, se fait quand C écoute le canal pour vérifier si D a retransmis le paquet de données à E; sinon cela veut dire que D ne l'a pas reçu. Le nœud C devrait alors envoyer un message d'erreur (*Route Error*) à A lui signifiant la rupture du lien en question. Le nœud A doit alors supprimer ce lien de sa table de routage, et doit entreprendre une nouvelle découverte de route.

3.4.3.1 Les erreurs

Quand un nœud ne peut pas atteindre un prochain hop, le paquet est retransmis jusqu'à un maximum de retransmission. Un nœud génère un paquet d'erreur lorsque le nombre maximum de retransmissions est atteint.

Le paquet d'erreurs doit contenir, entre autres, l'adresse de sa source, une entête d'option d'erreur de route avec comme type d'erreur le nœud injoignable, et l'adresse de destination de cette erreur.

Le nœud de destination qui reçoit l'erreur doit supprimer cette route de sa mémoire et initier une nouvelle découverte de route.

3.4.4 Caractéristiques supplémentaires :

Le protocole DSR utilise une entête spéciale appelée entête des options. Il existe 8 options qui déterminent la nature du paquet DSR.

3.4.4.1 Option requête de route

Dans ce cas, l'entête doit contenir l'adresse de l'initiateur de la requête, la destination (en broadcast), et le nombre limite de hops. La requête elle-même doit contenir, entre autres, l'adresse de la cible de la requête, et un champ que chaque nœud faisant le relai de cette requête doit remplir avec son adresse.

3.4.4.2 Option réponse de route

L'entête doit contenir l'adresse du nœud qui envoie la réponse et l'adresse de destination (l'initiateur de la requête). Le reste du paquet de la réponse contient, entre autres, la liste inverse des adresses des nœuds du chemin parcouru par la requête, ainsi que d'autres options liées à la réponse.

3.4.4.3 Option erreur de route

Il existe trois types de message d'erreur pour signaler l'impossibilité d'atteindre un nœud, qu'une information contenue dans un paquet n'est pas supportée, ou qu'une option ne l'est pas aussi. L'entête doit contenir les adresses de la source et de la destination de ce message d'erreur, ainsi qu'un champ spécifiant le type de cette erreur.

3.4.4.4 Options supplémentaires

D'autres entêtes d'options existent dans le protocole DSR, par exemple, l'option de requête d'un acquittement, et l'option d'acquittement (qui doit désigner que ce paquet est

un acquittement pour une certaine adresse venant d'un certain nœud), ainsi que d'autres options [34].

3.4.4.5 Quelques paramètres par défaut du protocole DSR

Tableau 10- Certaines valeurs par défaut du protocole DSR

Paramètre	Valeur	Description
DiscoveryHopLimit	255 hops	Le nombre de hop limite
BroadcastJitter	10 ms	Délai maximum que peut attendre un nœud avant d'envoyer un paquet
RouteCacheTimeout	300 s	Délai donné à chaque entrée de la mémoire du nœud avant d'être supprimée, si elle n'a pas été utilisée durant ce délai
SendBufferTimeout	30s	Un paquet en attente dans le buffer de transmission est supprimé après ce délai
RequestTableSize	64 nœuds	
RequestTableIds	16 id	Table FIFO des identificateurs et cibles des requêtes les plus récentes
MaxRequestRexmt	16 retransmissions	Nombre maximal de retransmissions de requêtes
MaxRequestPeriod	10 s	Période maximale possible entre deux requêtes pour la même cible
MainHoldoffTime	250 ms	Si le nœud a vérifié si le prochain hop est joignable durant ce délai, il n'a pas besoin de le refaire.

MaxMaintRexmt	2 retransmissions	Nombre d'essais de confirmation qu'un nœud est joignable ou pas
TryPassiveAcks	1 essai	Nombre d'essai d'écoute d'acquittement passif
PassiveAckTimeout	100 ms	Délai maximum d'attente d'un acquittement passif

3.5 Le protocole de routage TORA

3.5.1 Introduction au protocole TORA

Le protocole TORA (*Temporally-Ordered Routing Algorithm*) [35] est un protocole de routage adapté aux réseaux à topologie changeante de façon rapide et aléatoire. Contrairement au protocole AODV, TORA s'adapte à ces changements en propageant périodiquement des messages de contrôle à un débit indépendant de la dynamique de la topologie, ce qui limite leur nombre. En effet, dans le protocole AODV, un changement de topologie résulte en la propagation d'un message d'erreur par le nœud qui l'a constaté.

Comme protocole de routage, TORA dispose de trois opérations principales; la création des routes, la maintenance des routes, et la suppression des routes. Trois paquets sont utilisés à cet effet :

- Requête QRY (*Query*) : pour la création des routes.
- Mise-à-jour UPD (*Update*) : pour la création et la maintenance des routes.
- Suppression CLR (*Clear*) : pour la suppression des routes.

3.5.1.1 Les notations du protocole TORA

Chaque nœud est identifié par un identificateur unique *i*. Ce nœud *i* dispose d'une série de voisins *k*, formant ainsi des liens (*i*, *k*). Un lien peut être un lien non dirigé, un lien descendant (de *i* vers *k*), ou un lien montant (de *k* vers *i*).

Le protocole TORA assigne à chaque nœud un quintuplé appelé « hauteur » défini par : (tau[i], oid[i], r[i], delta[i], i). Ce quintuplé se compose de deux parties, les trois premières valeurs représentent le niveau de référence, et les deux dernières, le décalage (*Offset*) du nœud.

Pour la première partie, la valeur tau[i] représente l'instant de la rupture de lien. La valeur oid[i] représente l'identificateur du nœud qui a défini le niveau de référence. La valeur r[i] représente un bit qui indique si le niveau de référence a été changé par rapport à la valeur originale. La première valeur de la deuxième partie du quintuplé est la valeur delta[i], qui est un entier utilisé dans le but d'ordonner les nœuds en fonction du niveau de référence. La valeur *i* représente l'unique identificateur de chaque nœud.

Le protocole TORA est déroulé pour chaque destination. Initialement, chaque nœud (autre que la destination) dispose d'une hauteur indéfinie (NULL), hauteur = (-, -, -, -, i). La destination dispose d'une hauteur zéro, hauteur = (0, 0, 0, 0, j). Chaque nœud maintient, en plus de sa hauteur, toutes les hauteurs de ses voisin *k*, qui sont initialisées à NULL, à moins que la destination soit un de ces voisins, dans ce cas, elle est initialisée à zéro.

Chaque nœud *i* maintient une table de statuts des liens avec ses voisins *k*. Le statut d'un lien est déterminé par la hauteur. Si la hauteur d'un voisin *k* est supérieure à celle du nœud *i*, le lien est marqué montant (UP). Si la hauteur y est inférieure, le lien est marqué

descendant (DN). Si la hauteur du voisin *k* est NULL, le lien est marqué non dirigé. Enfin, si la hauteur du nœud *i* est NULL, le lien est marqué descendant (DN) avec tous les voisins qui ont une hauteur non NULL.

Quand un nœud *i* établit un nouveau lien avec un voisin *k*, une nouvelle entrée est ajoutée à la table de hauteurs de ce nœud *i*, et est mise à la valeur NULL (-, -, -, -, k).

3.5.2 Mécanismes de création des routes

La création des routes se fait à l'aide des deux paquets QRY et UPD. Pour le premier, le paquet contient uniquement l'adresse de la destination *j*. Le second contient, en plus de l'adresse de la destination, la hauteur du nœud envoyant ce paquet.

Dans le protocole TORA, les nœuds disposent d'un indicateur de requête de route, initialement inactif, et qui est activé à chaque fois qu'un nœud désire créer une route en envoyant un paquet QRY. Les nœuds maintiennent aussi l'instant de la création du dernier UPD et l'instant où chaque lien avec ses voisins est devenu actif.

Dès qu'un nœud reçoit le paquet QRY, le protocole agit comme suit [35] :

- Le nœud retransmet en broadcast le QRY, si celui-ci n'a aucun lien descendant (DN) et que son indicateur de requête est inactif.
- Le paquet est ignoré si ce nœud n'a aucun lien descendant vers la destination et que son indicateur de requête est actif cette fois-ci.
- Si ce nœud dispose d'au moins un lien DN vers la destination et que sa hauteur est non-NULL, alors, il vérifie si un paquet UPD a déjà été envoyé après la création du lien. Si c'est le cas, la requête QRY est ignorée, sinon, un UPD est envoyé en broadcast.

Lorsqu'un nœud *i* reçoit un paquet de mise-à-jour UPD, il procède comme suit [35] :

- Si la hauteur du nœud est NULL, le nœud *i* met sa hauteur égale au minimum des hauteurs de ses voisins non-NULL, avec un delta+1. Ensuite, il adapte sa table de hauteurs en fonction de cette nouvelle valeur et retransmet le paquet UPD avec la nouvelle hauteur.
- Si le nœud n'a aucun lien descendant, alors le nœud entame la procédure de maintenance des routes.

3.5.3 Mécanismes de maintenance des routes

La procédure de maintenance des routes a lieu seulement pour les nœuds ayant une hauteur non-NULL. La maintenance est effectuée par un nœud *i*, lorsque celui-ci n'a plus de liens descendant vers ses voisins. Cette maintenance est faite avec les paquets UDP afin de mettre à jour les hauteurs des nœuds, selon plusieurs cas décrits dans [35].

3.5.3.1 La suppression des routes

La suppression des routes est initiée par un nœud *i* lorsqu'une mise à jour a été lancée par un autre nœud et que la hauteur du nœud *i* a été utilisée comme nouvelle référence pour l'autre nœud. Dans ce cas, le nœud *i* met sa hauteur à NULL ainsi que celles de ses voisins.

3.6 Le protocole de routage DSDV

3.6.1 Introduction au protocoles DSDV

Le protocole DSDV (*Destination-Sequenced Distance-Vector*) [36] est un protocole proactif qui a été développé pour les réseaux ad hoc mobiles. Les nœuds maintiennent une

certaine connaissance de la topologie du réseau à travers une table de routage indiquant les routes possibles vers chaque destination.

3.6.2 Mécanismes de création des routes

La table de routage de chaque nœud dans le protocole DSDV doit contenir la liste de toutes les destinations possibles ainsi que le nombre de hops pour les atteindre. Chaque route vers une destination est décrite par un numéro de séquence.

Comme le réseau est mobile, les changements de topologie doivent être pris en considération dans les tables de routage. Pour cela, chaque nœud doit transmettre de façon périodique à tous ses voisins l'information sur sa table de routage et, surtout, chaque nouvelle mise-à-jour de celle-ci.

Le processus de transmission des mises-à-jour est géré par le protocole DSDV de façon à apporter au nœud une connaissance quasiment continue de la topologie qui l'entoure. À chaque fois qu'un nœud détecte une nouvelle information sur la topologie (du fait d'un déplacement d'un nœud, nouvelle route vers une destination, nouveau numéro de séquence,...), cette information est propagée après un certain délai. Ce délai est choisi comme décrit dans [36] de telle façon à ne pas se précipiter pour propager cette information au cas où une meilleure mise-à-jour arrive juste après. De cette manière, le débit des mises-à-jour est limité.

Les paquets de mises-à-jour contiennent des informations sur l'adresse de la destination, le nombre de hops nécessaires pour l'atteindre, et le numéro de séquence de cette destination. Chaque nœud partage sa table de routage avec le réseau, où il inclut son numéro de séquence.

Comme les tables de routage pourraient dépasser la taille limite des paquets du protocole, le protocole DSDV prévoit d'envoyer les tables de routage en plusieurs séquences. Les modifications à ces tables de routage sont envoyées sous forme de mises-à-jour.

3.6.3 Mécanismes de maintenance des routes

Un numéro de séquence d'une destination servira à sélectionner la meilleure route vers celle-ci. En effet, le numéro de séquence le plus récent détermine généralement la meilleure route. En effet, il faut que le nœud qui reçoit ce numéro de séquence tienne compte d'un certain délai d'attente avant d'adopter cette route pour cette destination. Le nombre de hops est utilisé pour sélectionner la meilleure route entre les routes ayant le même numéro de séquence pour la même destination.

Le numéro de séquence d'une destination peut servir à déterminer si la route est encore praticable ou pas. En effet, tous les numéros de séquences sont des chiffres pairs. Un nœud qui ne peut plus atteindre une destination, propage un nouveau numéro de séquence de celle-ci et qui est un chiffre impair. Ainsi, chaque nœud qui reçoit ce nouveau numéro de séquence pour cette destination conclut que cette route n'est plus valide et propage encore cette nouvelle mise-à-jour à ses voisins.

3.6.4 Caractéristiques supplémentaires

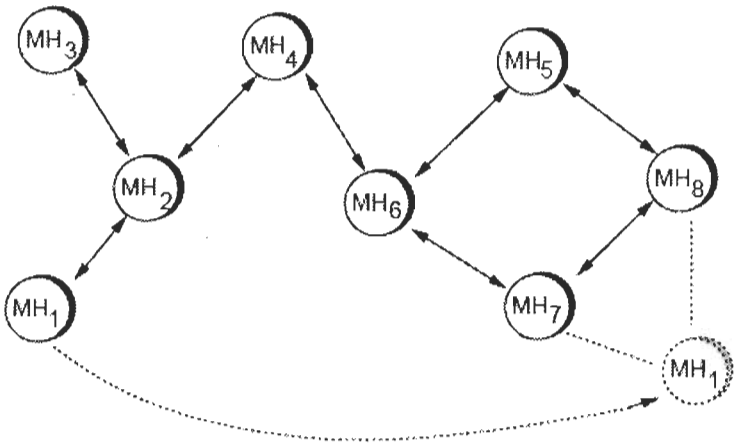

Figure 9- Exemple d'un scénario de routage DSDV

La figure ci-dessus décrit le réseau vu par le nœud MH4. Pour cela, ce nœud doit garder une table de routage, comme décrite par le tableau ci-dessous.

Selon ce tableau, le nœud MH4 sauvegarde pour chaque destination (MHi) le prochain hop, le nombre de hops nécessaires, le numéro de séquence de la destination, et le temps de la dernière mise-à-jour des informations sur la route vers ce numéro de séquence.

Le nœud MH4 doit propager sa table de routage à son voisinage en incluant seulement la destination, le nombre de hops et le numéro de séquence.

Si le nœud MH1 bouge de sa position actuelle jusqu'à une autre position (selon la figure précédente, dans la région de MH7 et MH8) et comme les transmissions de mises-à-jour doivent se faire de façon périodique, le nœud MH4 pourra détecter que la route vers MH1 n'est plus valide. En recevant un nouveau numéro de séquence, MH4 remet à jour sa table de routage et la transmet à son voisinage.

Tableau 11- Table de routage DSDV sauvegardée dans un nœud

Destination	Prochain hop	Nb. hops	Num. séq.	Temps d'installation
MH1	MH2	2	S406_MH1	T001_MH4
MH2	MH2	1	S128_MH2	T001_MH4
MH3	MH2	2	S564_MH3	T001_MH4
MH4	MH4	0	S710_MH4	T001_MH4
MH5	MH6	2	S392_MH5	T002_MH4
MH6	MH6	1	S076_MH6	T001_MH4
MH7	MH6	2	S128_MH7	T002_MH4
MH8	MH6	3	S050_MH8	T002_MH4

3.7 Le protocole de routage PUMA

3.7.1 Introduction au protocole PUMA

Le protocole PUMA (*Protocol for Unified Multicasting through Announcements*) est un protocole de routage Mesh qui entreprend une découverte de route seulement si celle-ci est requise [37]. Les nœuds se rassemblent autour de groupes de multicast. Chaque groupe est géré par un nœud élu appelé noyau, ce qui va servir à simplifier le routage des paquets. Le regroupement des nœuds de cette manière n'enlève rien à leur indépendance, ni au fait que ce protocole soit auto-organisable et de type Mesh.

3.7.2 Mécanismes de création des routes

Le protocole de routage PUMA utilise un seul type de paquet de contrôle, appelés MA (*Multicast Announcements*). Ces paquets servent à [37-39] :

- Élire un noyau.
- Détecter des sources de paquets de données n'appartenant pas au même groupe.
- Rejoindre et quitter un groupe.
- Maintenir des groupes.

PUMA s'occupe de créer les groupes. Si un nœud rejoint un groupe avant les autres nœuds, celui-ci devient automatiquement son noyau. Si plusieurs nœuds rejoignent un même groupe en même temps, le nœud disposant du plus haut ID devient le noyau [37, 39, 40].

Le protocole PUMA doit faire élire un noyau par groupe. Une fois crée, un groupe émet des annonces à travers les paquets MA [37, 39, 40]. Si un nœud en reçoit un, celui-ci rejoint ce groupe si un noyau a déjà été élu; sinon, le nœud se considère comme noyau de ce groupe [39, 40].

Le noyau d'un groupe transmet périodiquement des MA à travers le réseau afin d'établir une liste de connectivité qui sert à construire une structure Mesh et d'acheminer les paquets des nœuds vers le noyau du groupe. Chaque entrée dans la liste de connectivité sauvegarde l'adresse du voisin et l'instant où celui-ci a envoyé une annonce [37]. Toutes ces opérations de création d'un réseau à l'aide des annonces MA sont décrites dans l'exemple décrit par la figure et le tableau ci-dessous.

Un paquet MA contient les champs suivants [37, 38] :

- ID du noyau
- ID du groupe
- Numéro de séquence du groupe
- Distance par rapport au noyau en hops + 1
- Un champ indiquant si le nœud est membre ou pas d'un réseau Mesh

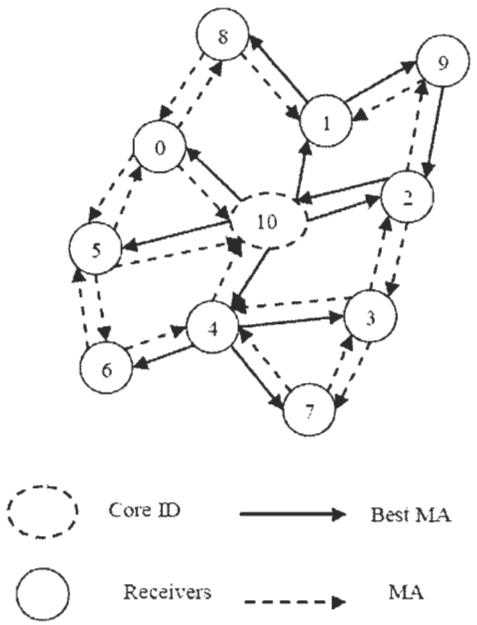

Figure 10- Exemple de formation d'un réseau avec les annonces multicast MA

Tableau 12- Exemple de l'organisation des nœuds dans le réseau

Nœud	**Distance du noyau**	**Parent**
10 (noyau)	1	10
5	2	10
6	3	4

3.7.2.1 Liste de connectivité et propagation des annonces Multicast

Un nœud qui se croit noyau d'un groupe émet des annonces Multicast (MA) de façon périodique à son groupe [37, 39]. Les annonces se propagent donc à travers le réseau en établissant une liste de connectivité de chaque nœud dans le réseau. Si plusieurs groupes existent, les nœuds gardent une liste de connectivité pour chaque groupe, à l'aide desquelles, les nœuds transmettent les paquets d'un point à un autre. Chaque nœud met-à-jour sa liste de connectivité en fonction des nouvelles informations sur le numéro de séquence reçu de son voisinage. Ainsi, un nœud qui reçoit un paquet MA sauvegarde, en plus du paquet, l'instant où celui-ci l'a reçu et le voisin qui l'a transmis. Après quoi, une annonce est générée par ce nœud en se basant sur sa propre liste de connectivité [39].

Un paquet MA généré par un nœud a les mêmes informations que son parent sur l'ID du noyau, ID du groupe, et le numéro de séquence de ses parents, mais une distance du noyau (hops) plus grande d'un pas par rapport à celle de son parent.

Pour les réseaux ayant plusieurs groupes, la propagation des annonces peut prendre beaucoup de temps. En effet, si une seule annonce d'un groupe est propagée, les nœuds la propagent rapidement. Mais si de multiples groupes génèrent des annonces, les nœuds doivent attendre un certain intervalle entre chaque annonce pour pouvoir les propager [39].

3.7.2.2 Routage des paquets de données

Le routage des paquets de données est effectué par les membres Mesh du réseau. Ainsi, un nœud qui veut transmettre un paquet de données choisit dans sa liste de connectivité le parent comme destination, et ainsi de suite, jusqu'à ce que ce paquet atteigne les membres

Mesh. Après quoi, ces derniers s'occupent de le propager dans le réseau en utilisant le ID du paquet pour éviter la duplication des paquets [39].

3.7.3 Mécanismes de maintenance des routes

Il existe trois types de dispositifs dans le routage PUMA : les récepteurs, les membres Mesh et les non-récepteurs. Initialement, pour rejoindre un réseau Mesh, les récepteurs doivent mettre l'indicateur 'membre mesh' à vrai dans les paquets MA. Un non-récepteur se considère comme membre Mesh si celui-ci dispose d'au moins un enfant dans sa liste de connectivité. Un nœud dans la liste de connectivité est considéré comme un enfant si :

- son indicateur membre Mesh est vrai,
- sa distance par rapport au noyau est plus grande que celle du nœud, et
- son annonce a été émise avant un certain délai correspondant au double de l'intervalle des annonces multicast.

Pour remettre à jour la liste de connectivité, on utilise les paquets de données. En effet, un nœud qui transmet un paquet à son parent, écoute le canal un certain intervalle de temps en attendant la retransmission du paquet par le parent. Ceci servira indirectement comme un acquittement de la réception du paquet par le parent. Si au bout de l'intervalle d'attente du nœud, aucune transmission n'a lieu, le nœud conclut que le parent n'y est plus, et est donc retiré de la liste de connectivité [39].

3.8 Conclusion

Les différences entre les protocoles de routage se démarquent après cette étude fonctionnelle de ces méthodes en termes d'établissement de routes, de leur entretien et des

paquets utilisés à cet effet. Statuer sur leurs performances pour choisir le meilleur est encore prématuré. Pour cela, une étude plus approfondie avec une simulation de leur comportement est indispensable.

Chapitre 4 - Les outils de simulation et environnement de test

4.1 Introduction

Lors du développement d'un réseau de capteurs sans fil pour une application réelle, la simulation par logiciel de son comportement et de ses performances est essentielle. Plusieurs facteurs influencent le comportement d'un RCSF dont la partie physique du capteur (antenne et radio), la topologie, le canal de propagation, la consommation d'énergie, et le protocole de communication. Un simulateur qui réussit à modéliser le mieux possible ces facteurs d'influence, permettra forcément une meilleure représentation du comportement réel du réseau.

Plusieurs simulateurs de RCSF ou de simulateurs de réseaux adaptés aux RCSF ont été développés. Par exemple :

- Castalia Simulator : simulateur spécialisé uniquement dans les RCSF; développé par le laboratoire NICTA [41] et basé sur la plateforme du simulateur OMNeT++. Ce dernier est un simulateur plus général de réseaux.
- MiXiM : simulateur basé également sur OMNeT++ et spécialisé dans la simulation des RCSF [42].
- TOSSIM : simulateur de RCSF développé pour simuler les plateformes utilisant le système d'exploitation TinyOS. Ce simulateur est performant pour modéliser le

comportement des couches d'application, mais présente de faibles performances pour simuler le comportement des protocoles MAC [43].

- WSNet : simulateur de RCSF qui peut être utilisé avec un autre outil de simulation WSim pour générer des codes pour les microcontrôleurs de capteurs [42].
- NS2 : simulateur de réseaux qui peut simuler les RCSF. Il supporte plusieurs protocoles de communication au niveau de toutes les couches OSI. Ce simulateur dispose de la plus grande communauté de développeurs, et donc d'une meilleure notoriété.

Entre les quatre premiers simulateurs (Castalia, MIXIM, TOSSIM, WSNet), Castalia présente les meilleurs atouts pour simuler un RCSF. Certes, il existe plusieurs similitudes entre les simulateurs. À l'exception de TOSSIM, ils supportent tous une topologie à 3D qui peut être générée de plusieurs façons (automatiquement ou manuellement) [42]. Concernant le canal de propagation, les quatre simulateurs adoptent des modèles de propagation assez réalistes (*Shadowing* et autres). Cependant, Castalia se démarque par son modèle de canal variant dans le temps. De plus, Castalia utilise un meilleur modèle de détection d'erreurs de paquets à la réception [42]. Castalia est aussi le seul à calculer la consommation d'énergie lors des transitions entre les trois états de la radio (RX, TX, et veille) [42]. Les autres simulateurs se contentent de calculer la consommation à la transmission et à la réception ou bien seulement entre deux transitions.

De son côté, NS2 se démarque par sa notoriété et par sa grande communauté d'utilisateurs. Ainsi, plusieurs contributions d'implémentation de protocoles de différentes couches ont été proposées par la communauté des utilisateurs. Castalia de son côté est un simulateur très flexible et permet de mieux comprendre le comportement d'un nœud de

capteur dans un réseau. Ainsi, les deux simulateurs ont été utilisés dans ce projet. Au début Castalia a permis la compréhension de plusieurs fonctionnalités des réseaux de capteurs (Une description des fonctionnalités de Castalia sont résumées dans l'annexe C). Cependant, vu la notoriété de NS2 dans le domaine de la simulation des réseaux, celui-ci s'est révélé être le choix ultime pour développer le reste du projet de recherche.

4.2 Le simulateur NS2

4.2.1 Introduction à NS2

Le simulateur NS2 est décrit par le simulateur TCL class. Le script général de la simulation y est décrit en appelant les différentes méthodes pour créer les nœuds, les topologies, et les autres aspects de la simulation.

4.2.2 Initialisation du simulateur

La procédure d'initialisation se fait dans un fichier tcl :

- Initialisation du format des paquets : établir les champs de paquets utilisés par la simulation
- Créer un planificateur d'évènements : il en existe quatre :
    - Liste des liens simples : les évènements sont classés du plus ancien au plus récent, l'exécution se fait sous forme FIFO.
    - Pile : un planificateur sous forme de pile.
    - Calendrier : c'est une structure qui ressemble au calendrier classique, où les évènements du même mois/jour mais d'années différentes peuvent être enregistrés dans le même jour

- A temps réel : encore en développement.

- Créer un « null agent » (relatif au nœud sink).

Tableau 13- Liste des commandes récurrentes dans les scripts des simulations

Commande	Description
set ns_ [new Simulator]	Création d'une proposition d'objet
set now [$ns_ now]	Le planificateur conserve une trace du temps dans la simulation
$ns_ halt	Arrêter complètement ou temporairement le planificateur
$ns_ run	Démarrer le planificateur ou la simulation
$ns_ at <time> <event>	Planifie l'exécution d'un évènement <event> à un temps <time>
$ns_ cancel <event>	Annuler l'évènement <event> de la liste du planificateur
$ns_ create-trace <type> <file> <src> <dst> <optional arg: op>	Créer un objet de traçage de type <type> entre la source <src> et la destination <dst> et attacher l'objet de traçage au fichier <file> afin d'écrire les sorties du traçage
$ns_ flush-trace	Nettoyer les buffers des objets de traçage
$ns_ gen-map	Enlever certaines informations concernant les liens, et les nœuds et leurs composantes. Ceci peut être utilisé pour les scénarios de réseaux sans fil
$ns_ use-scheduler <type>	Spécifier le type du planificateur
$ns_ after <delay> <event>	Planifier un évènement à exécuter après un certain intervalle de temps <delay>

$ns_ clearMemTrace	Mémoire de débogage
$ns_ is-started	La simulation a commencé ? Vrai ou faux ?
$ns_ dumpq	Enlever certains évènements lors de l'arrêt du planificateur
$ns_ create_packetformat	Etablir le format des paquets du simulateur

4.2.3 Les Agents

Les agents représentent le point où les paquets de la couche réseau sont construits ou traités, et sont utilisés pour l'implémentation des protocoles des différentes couches.

Les agents assignent plusieurs champs aux paquets avant leur transmission, comme le montre le tableau ci-dessous.

Tableau 14- Informations internes des agents

Information	**Descriptions**
addr_	Adresse source
dst_	Adresse de destination
size_	Taille des paquets en octets (information placée dans l'entête)
type_	Type du paquet
fid_	Le flux IP
prio_	Le champ de priorité IP
flags_	Indicateur de paquet
defttl_	Valeur IP par défaut

Certaines commandes reviennent souvent dans la manipulation des agents. Le tableau ci-dessous en liste quelques-unes.

Tableau 15- Liste de certaines commandes relatives aux agents

Commandes	Descriptions
ns_ attach-agent <node> <agent>	Cette commande attache <agent> à <node>. L'agent est créé à l'aide de la commande set agent [new Agent/AgentType], où, Agent/AgentType définit la classe de l'agent.
$agent port	Retourne le numéro du port auquel l'agent est attaché.
$agent dst-port	Retourne le numéro du port de la destination. Quand une connexion entre deux nœuds est établie, chaque nœud sauvegarde le port de la destination dans la variable dst_port.
$agent attach-app <s_type>	Cette commande attache une application de type <s_type> à l'agent.
$ns_ connect <src> <dst>	Établir la connexion entre les agents src et dst.
$ns_ create-connection <srctype> <src> <dsttype> <dst> <pktclass>	Établir une connexion complète entre deux agents. D'abords créer une source de type <srctype> et la lier à <src>. Ensuite, créer une destination de type <dsttype> et la lier à<dst>. Enfin, connecter les agents src et dst.
$agent attach-trace <file>	Attacher le fichier <file> l'agent afin de permettre le traçage des évènements avec NAM.

4.2.4 Les réseaux WLAN

Un réseau WLAN est simulé dans NS2 comme décrit dans la figure ci-dessous.

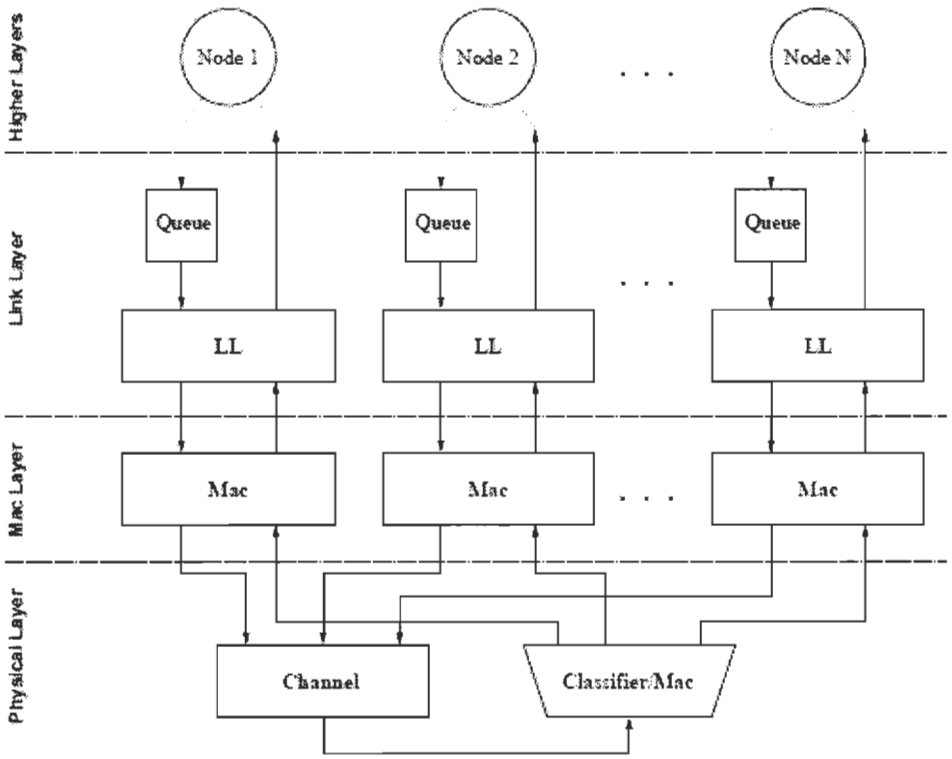

Figure 11- La connectivité dans un réseau WLAN [25]

4.2.4.1 La classe canal

Le canal simule l'état actuel de la transmission des paquets par la couche physique. Le canal intègre un support partagé pour les mécanismes de conflits de transmission. Ceci permet au MAC d'effectuer une détection de porteuse, de conflits de transmission et de détection de collisions. Si plus d'une transmission se chevauchent dans le temps, le canal génère un indicateur de collision que le MAC peut vérifier afin de gérer la détection des collisions.

4.2.4.2 Le classificateur MAC

Il est utilisé pour la duplication des paquets en broadcast. À la réception d'un paquet en broadcast, ou bien, dans le cas où l'adresse de destination du paquet est difficilement

identifiable, le message est dupliqué et envoyé en broadcast à tout le voisinage excepté l'émetteur de ce message.

4.2.4.3 La classe MAC

Elle implémente et simule la partie MAC du protocole. Lors de l'émission, la classe MAC est responsable de l'ajout de l'entête MAC au paquet à transmettre. À la réception, le MAC reçoit les paquets du classificateur MAC, effectue le traitement du protocole MAC et passe le paquet à la couche de liaison (couche réseau).

La classe MAC intègre les paramètres suivants :

bandwidth_	taux de modulation du MAC
hlen_	taille de l'entête MAC
label_	Adresse MAC

4.2.4.4 La classe couche de liaison

La couche de liaison (ou la couche réseau) LL (*Link-Layer*) implémente les protocoles de liaison des données (Routage). Cette classe sert aussi à déterminer l'adresse MAC de destination dans l'entête des paquets.

Cette classe gère deux différentes tâches : trouver l'adresse IP du prochain hop (routage) et la convertir en adresse MAC.

4.2.4.5 Quelques commandes relatives aux réseaux WLAN

Tableau 16- Liste de certaines commandes relatives aux réseaux WLAN

Commandes	Descriptions
$ns_ make-lan <nodelist> <bw> <delay> <LL> <ifq> <MAC> <channel> <phy>	Créer un LAN à partir d'un ensemble de nœuds donné par <nodelist>. Plusieurs caractéristiques peuvent être définies : débit, délai, le buffer (queue), MAC, canal, LL, couche physique,… Les valeurs par défaut sont : <LL> .. LL <ifq>.. Queue/DropTail <MAC>.. Mac <channel>.. Channel and <phy>.. Phy/WiredPhy
$ns_ newLan <nodelist> <BW> <delay> <args>	Créer un réseau similaire au précédent. Sauf que cette commande peut le décrire plus dans le détail. Les arguments qui peuvent être passés sont : LL, ifq, MAC, canal, phy et adresse.
$lannode addNode <nodes> <bw> <delay> <LL> <ifq> <MAC> <phy>	Ajouter une liste de nœuds à un réseau déjà créé.

4.2.5 Le modèle d'énergie

On peut configurer les paramètres suivants :

$ns_ node-config -energyModel $energymodel \

```
-rxPower $p_rx \
-txPower $p_tx \
-initialEnergy $initialenergy
```

Tableau 17- Les valeurs par défauts du modèle d'énergie

Paramètre	**Valeurs optionnelle**	**Valeurs par défaut**
`-energyModel`	« Modèle d'énergie »	Aucune
`-rxPower`	Puissance de réception	281.8mW
`-txPower`	Puissance de transmission	281.8mW
`-initialEnergy`	Énergie initiale en joules	0 joules

4.2.6 Démarrer NS2

NS2 démarre avec la commande « ns <tclscript> ». Le tclscript est un fichier en langage TCL qui définit le scénario de la simulation (topologie et les évènements). Tout le reste de la simulation dépend de ce fichier Tcl. Le script Tcl écrit les sorties, trace les évènements, et peut démarrer la fenêtre « nam » (*Network Animator*) afin de visualiser la simulation.

4.2.7 Démarrer NAM

La fenêtre d'animation NAM (*Network Animator*) démarre avec la commande « nam <nam-file> », donnant lieu à la fenêtre décrite par la figure ci-dessous.

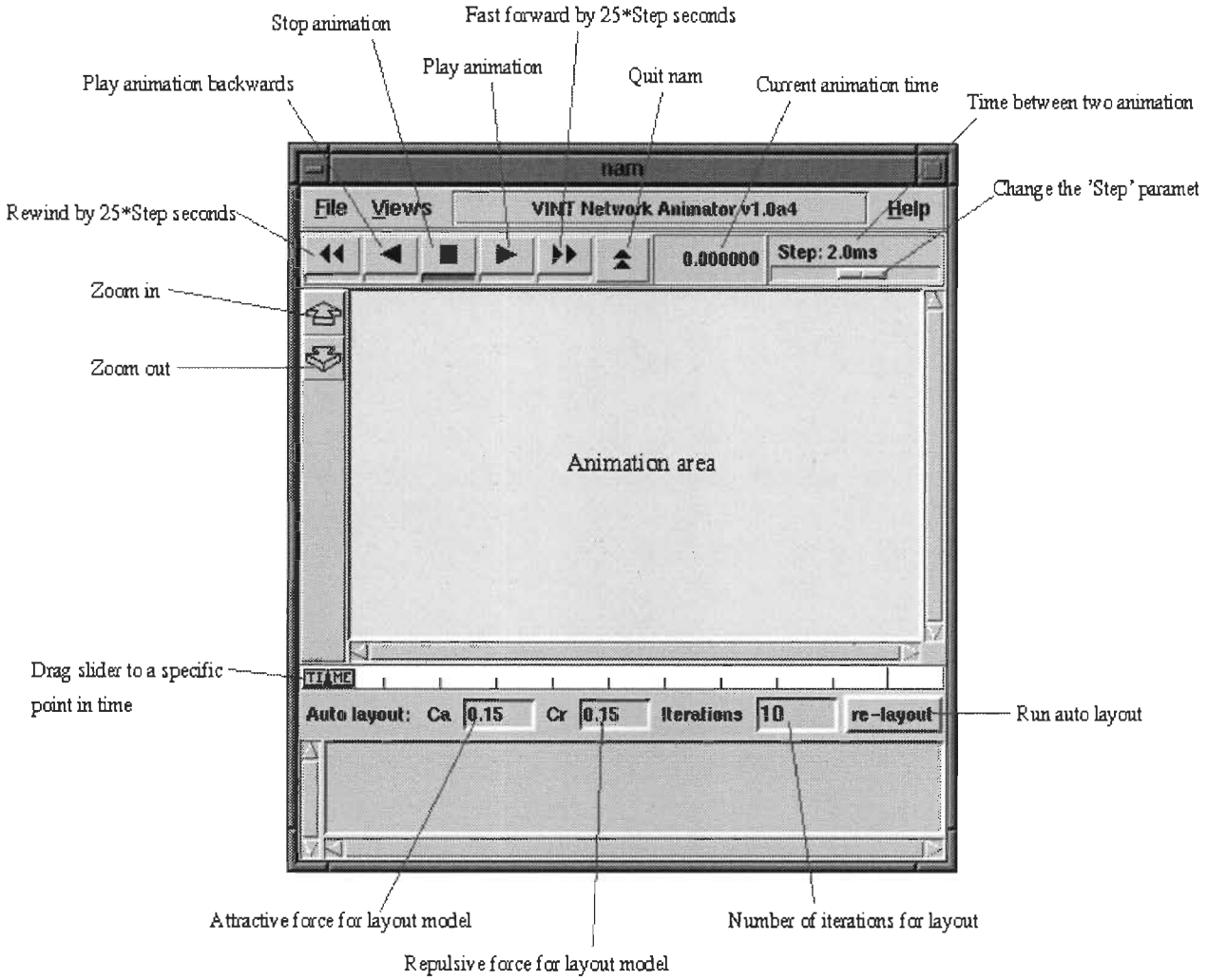

Figure 12- La fenêtre nam [44]

4.2.8 Le script TCL

Les étapes suivantes décrivent une topologie simple d'une communication entre deux nœuds liés par un lien simple.

- Au tout début du script, on crée un objet de simulation avec la commande suivante :

```
set ns [new Simulator]
```

- Par la suite, on ouvre un fichier de traçage des données qui sera utilisé par nam :

```
set nf [open out.nam w]
$ns namtrace-all $nf
```

La première ligne ouvre un fichier « out.nam » en mode écriture. La deuxième ligne indique au simulateur qu'un fichier a été créé. Celui-ci est utilisé pour l'écriture des données de la simulation qui seront utilisées par le nam.

- Créer une procédure « finish » qui ferme le fichier de trace et démarre NS2 par la suite :

```
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
```

- La prochaine ligne demande au simulateur d'exécuter la procédure finish après 5 secondes du temps de simulation :

```
$ns at 5.0 "finish"
```

- Enfin, la simulation démarre avec la commande suivante :

```
$ns run
```

- La prochaine étape consiste à créer 2 nœuds et un lien :

  On procède à la création de deux nœuds (n0 et n1) :

```
set n0 [$ns node]
set n1 [$ns node]
```

  Connecter les deux nœuds avec un lien en mode duplex avec un débit de 1Mb, un délai de 10ms, et une queue DropTail :

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

Après que fichier TCL ait été sauvegardé, nam peut être lancé donnant lieu à une sortie ressemblant à la figure ci-dessous.

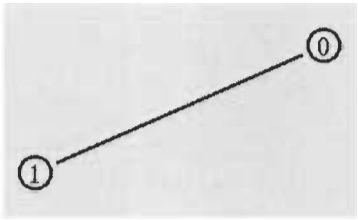

Figure 13- Exemple de topologie simple entre deux nœuds

- L'envoi des données du nœud n0 au nœud n1 se fait de la manière suivante :

Dans NS, la communication entre deux nœuds se fait entre deux objets appelés « agents » contenus dans chacun des deux nœuds. La première étape est donc de créer le premier agent UDP (*User Datagram Protocol*) et de l'attacher au nœud 0 avec la méthode de routage CBR (*Constraint Based Routing*). Les deux protocoles sont déjà installés dans NS2 :

```
#Créer un agent UDP et l'attacher au nœud n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
# Créer une source de traffic CBR et l'attacher à udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

La deuxième étape est de créer un agent Null et de l'attacher au nœud n1 :

```
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0
```

Maintenant les deux agents peuvent être interconnectés :

```
$ns connect $udp0 $null 0
```

A présent, la communication peut démarrer :

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
```

A noter qu'il est bien sûr préférable de mettre les lignes précédentes avant la ligne « `$ns at 5.0 "finish"` »

4.2.9 Un nouveau protocole pour NS2

La création d'un nouveau protocole dans NS2 se fait grâce aux étapes suivantes :

- Créer le fichier '.h' contenant la structure des données dans les paquets à envoyer, et la déclaration des fonctions de bases du protocole.
- Créer le programme '.cc' du protocole :
    - Définir les liens entre les variables utilisées par le programme C++ et le code Tcl.
    - Définir la fonction 'command ()' qui est appelée lorsque le code Tcl donne une commande à l'agent du nœud.
    - Définir la fonction 'recv ()' qui est appelée lorsqu'un paquet est reçu.
    - Si l'agent utilisé par le protocole n'existe pas dans NS, alors il faut le définir dans les fichiers source du logiciel :
        - Définir le nouveau paquet (dans 'packet.h'),
        - son nom (dans la fonction p_info()),
        - sa taille (ns-default.tcl),
        - ajouter une nouvelle entrée de ce paquet (dans ns-packet.tcl),
        - enfin dans le fichier 'Makefile', ajouter le nom du protocole '.o',

- Recompiler le logiciel afin de prendre en considération ces changements.

4.3 **Ecrire le code Tcl afin de définir le scénario de la simulation.**Création d'un nouveau scénario dans NS2

4.3.1 Le script TCL

Un scénario de simulation dans NS2 est créé en commençant par un script TCL. Ce script va regrouper tous les besoins d'un réseau comme paramètres du canal, des couches OSI, la taille du champ de la simulation, positions des nœuds, le modèle de propagation, le trafic (couche application), le traçage des évènements, et beaucoup d'autres caractéristiques du réseau. Dans l'annexe B de ce document se trouve notre script de simulation d'un scénario du protocole AODV.

Le tableau suivant liste tous les paramètres réglables d'un réseau typique de ce projet de maitrise. Voire l'annexe pour plus de détails.

Tableau 18- Définition d'un réseau dans NS2

Paramètre	Description
Le canal	Choisir un canal sans fil. Puisque NS2 simule tous les types de réseaux
Le modèle de propagation	Le choix entre trois types de modèles (Free Space, Two-Ray Ground, et Shadowing)
Type de la couche physique	Dans ce cas, c'est la couche physique du standard IEEE 802.15.4
Type de la couche MAC	Ça va être la couche MAC du standard IEEE 802.15.4 pour tous les scénarios
Type de la queue (buffer)	À savoir, une mémoire FIFO
Taille de la queue	Combien de paquets à bufférisér
Initialiser la couche	Couche LL de NS2
Protocole de routage	Choisir le protocole de routage
Type de l'antenne	Omni directionnelle
Taille du champ	mètres*mètres
Énergie initiale	En Joules (ex. Piles AA environ 15000 joules)
Consommation en modes TX, RX, veille et veille profonde	En watt
Fichiers de traçage	Envoyer le simulateur vers les fichiers de traçage des évènements et du traçage NAM
Fichier de topologie	Positions des nœuds
Le trafic	Dans tous les scénarios ça va être le trafic CBR (envoyer un paquet par un certain intervalle)
Définir la sensibilité du module radio	Le seuil de réception des signaux radio
Fréquence, puissance d'émission et hauteur d'antenne	Respectivement, en Hz, Watt, et mètres
NAM	Définition de la fenêtre d'animation
Définir la procédure d'arrêt et de démarrage	Selon cet ordre (arrêt ensuite démarrage)

4.3.2 Fichiers de traçage des évènements

C'est un fichier créé automatiquement par NS2 qui détaille TOUS les évènements qui se sont passés dans le réseau (généralement sous forme de fichier .txt). Un évènement est souvent lié à un paquet (reçu, transmis, ou échoué). Chaque ligne donne plusieurs informations sur l'évènement :

- s, r, d, ou f : respectivement, envoyé (*sent*), reçu (*received*), perdu (*dropped*), ou acheminé (*forward*).
- L'instant de l'évènement en secondes.
- L'adresse du nœud qui provoque cet évènement.
- Le numéro de la séquence de ce paquet.
- Le type du paquet (données, routage, acquittements,…).
- La taille du paquet en octets.
- L'état de l'énergie du nœud.
- Les adresses contenues dans ce paquet (source et destination).
- Nombre de hops.
- Certains protocole rajoutent des informations qui leurs sont spécifiques (ex. l'AODV rajoute le type du paquet : requête ou réponse).

4.3.3 Traitement des fichiers de traçage

Une simulation de 25 nœuds avec une source et une destination produit un fichier de plus de 10000 lignes d'évènements. Pour cela, si on veut estimer les performances du

réseau en termes de délais et autres critères, il n'y a pas de choix que d'automatiser cela avec, par exemple, le langage AWK [référence].

Connaissant le contenu des lignes du fichier de traçage, il est possible d'évaluer tous les critères de performance du réseau. Notre code AWK de la simulation de l'AODV est donné en annexe B.

4.3.4 Exemple de simulation d'un réseau sans fil avec NS2

La topologie consiste en deux nœuds mobiles dans un champ de 500m X 500m. Les deux nœuds se trouvent au début aux limites du champ, et commencent à se rapprocher l'un de l'autre, ensuite ils commencent à s'éloigner l'un de l'autre. La perte des paquets augmente plus on s'éloigne du rayon d'émission des nœuds et vice versa.

Le script Tcl décrivant le scénario est :

```
# ======================================================================
# Define options
# ======================================================================
set val(chan)         Channel/WirelessChannel  ;# Type du canal
set val(prop)         Propagation/TwoRayGround ;# Modèle de propagation
set val(ant)          Antenna/OmniAntenna      ;# Type d'antenne
set val(ll)           LL                       ;# Type de couche réseau
set val(ifq)          Queue/DropTail/PriQueue  ;# Type de la queue
set val(ifqlen)       50                       ;# max de paquets dans ifq
set val(netif)        Phy/WirelessPhy          ;# Type de réseau
set val(mac)          Mac/802_11               ;# MAC
set val(rp)           DSDV                     ;# Méthode de routage
set val(nn)           2                        ;# Nombre de nœuds
```

Création de la simulation :

```
set ns_    [new Simulator]
```

Permettre le traçage de toutes les informations de toutes les couches :

```
set tracefd     [open simple.tr w]
$ns_ trace-all $tracefd
```

Localisation des nœuds :

```
set topo        [new Topography]
```

Taille du champ :

```
$topo load_flatgrid 500 500
```

Créer un observateur de la simulation GOD (*General Operations Director*) qui sert à sauvegarder les informations sur le nombre de nœuds mobiles, le nombre minimum de hops pour atteindre un autre nœud,…

```
create-god $val(nn)
```

Désormais, il faut configurer les nœuds. Les valeurs par défaut sont :

```
# $ns_ node-config -addressingType flat ou hierarchical ou expanded
#                  -adhocRouting   DSDV ou DSR ou TORA
#                  -llType          LL
#                  -macType         Mac/802_11
#                  -propType        "Propagation/TwoRayGround"
#                  -ifqType         "Queue/DropTail/PriQueue"
#                  -ifqLen          50
#                  -phyType         "Phy/WirelessPhy"
#                  -antType         "Antenna/OmniAntenna"
#                  -channelType     "Channel/WirelessChannel"
```

```
#                   -topoInstance   $topo
#                   -energyModel    "EnergyModel"
#                   -initialEnergy  (en Joules)
#                   -rxPower        (en W)
#                   -txPower        (en W)
#                   -agentTrace     ON ou OFF
#                   -routerTrace    ON ou OFF
#                   -macTrace       ON ou OFF
#                   -movementTrace  ON ou OFF
```

La configuration des nœuds se fait comme ceci :

```
# Configure nodes
        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -topoInstance $topo \
                         -channelType $val(chan) \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF
```

Création de deux nœuds mobiles :

```
        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node ]
                $node_($i) random-motion 0    ;# désactiver les mouvements
aléatoires
        }
```

Définition des positions et de la nature du mouvement des nœuds :

```
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 390.0
$node_(1) set Y_ 385.0
$node_(1) set Z_ 0.0

# Mouvements des nœuds :
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 15.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 1.0"

# Node_(1) s'éloigne du node_(0) :
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 15.0"
```

Le premier nœud (0) commence à la position (5, 2, 0), le second nœud (1) à (390, 385, 0). Le nœud 1 commence son mouvement à l'instant 50s vers la position (25, 20, 0) avec une vitesse de 15m/s, le nœud 0 commence son mouvement à l'instant 10s vers la position (20, 18, 0) avec une vitesse de 1m/s. Enfin le nœud 1 commence à s'éloigner du nœud 0 à l'instant 100s vers la position (490, 480, 0) avec une vitesse de 15m/s.

Par la suite, il faut définir les agents (TCP) responsables du trafic :

```
# Connexions TCP entre node_(0) and node_(1)

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

```
$ns_ at 10.0 "$ftp start"
```

La fin de la simulation :

```
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 150.0001 "stop"
$ns_ at 150.0002 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns_ tracefd
    close $tracefd
}
```

Finalement, la commande de démarrage de la simulation est :

```
puts "Starting Simulation..."
$ns_ run
```

4.4 Conclusion

L'environnement de simulation est désormais défini, plusieurs outils offerts par NS2 vont nous permettre de simuler le comportement d'un nœud de capteur dans le réseau en prenant en compte sa couche physique, le protocole de communication (couche MAC et réseau), l'application et l'environnement de propagation. Dans ce dernier, il est important de trouver des topologies de déploiement du réseau qui modélisent le monde réel en ce qui concerne les distances entre les capteurs, leurs positions, et la mobilité des nœuds.

Chapitre 5 - Synthèse des résultats

5.1 Introduction

Dans ce chapitre, les scénarios de simulation sont tout d'abord décrits. Leur choix s'est fait de telle façon à augmenter la complexité de la topologie progressivement en intégrant la mobilité et en finissant par une topologie dans un environnement fermé. Les résultats des simulations des scénarios définis sont donnés par la suite, avec une étude comparative entre les différentes méthodes simulées.

5.2 Les scénarios de simulation

Afin d'effectuer une étude comparative entre les différents protocoles de routage, 8 scénarios ont été simulés. En commençant par le plus simple qui est de faire communiquer 2 nœuds avec une source et une destination jusqu'à simuler le comportement d'un réseau dans un environnement fermé. Ce dernier simule le comportement dans une maison avec plusieurs pièces; avec plusieurs sources et destinations dont une mobile.

Tous les scénarios ont été testés sur les deux modèles de propagation : Le Two-Ray Ground et le Shadowing :

- Les nœuds communiquent à 1 paquet de 32 octets par seconde durant 100 secondes.
- L'énergie initiale est de 15000 Joules ou 4.166 Wh
- La puissance de transmission est de 2.2 mW

- La sensibilité du module radio dans le cas du Two-Ray Ground a été choisie plus élevée que pour celle du modèle de propagation Shadowing afin de garantir un rayon de transmission quasiment identique. De cette manière les distances entre les nœuds restent de l'ordre du faisable.

- Les paramètres du modèle de propagation Shadowing sont : une déviation du Shadowing σ_{dB} égale à 7 (environnement fermé avec séparations rigides [référence]) et un exposant de l'atténuation β égale à 5 (environnement interne avec visibilité obstruée [référence]).

5.3 Scénario 1

Cela consiste en deux nœuds se trouvant à une distance de 12 mètres.

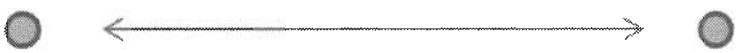

Figure 14- Topologie à 2 nœuds

5.4 Scénario 2

Figure 15- Topologie à 3 nœuds

5.5 Scénario 3

Figure 16- Topologie à 5 nœuds

La distance est de 12 m entre chaque deux capteurs consécutifs pour le modèle Two-Ray Ground et de 10 m pour le Shadowing afin de garantir une meilleure couverture pour ce dernier.

5.6 Scénario 4

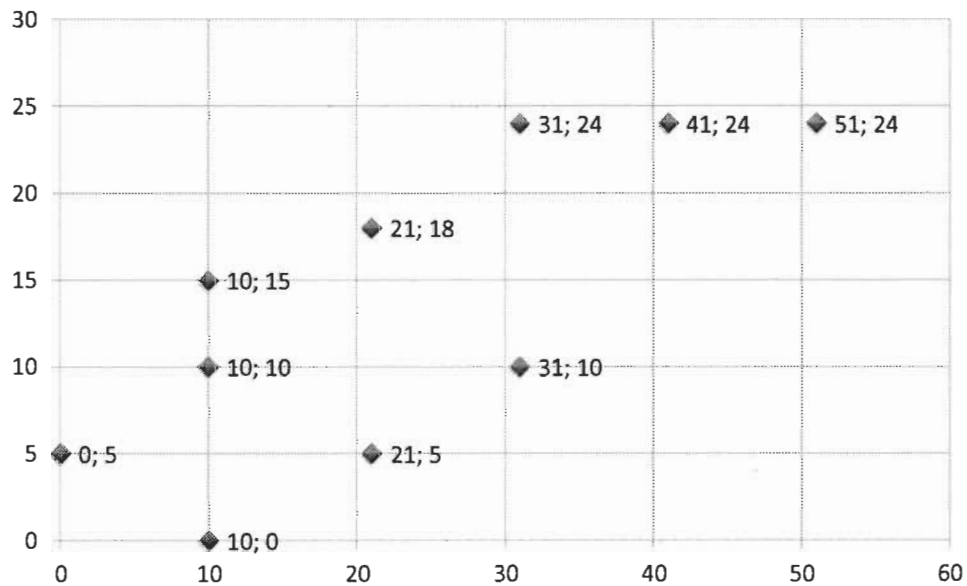

Figure 17- Topologie à 10 nœuds

C'est un scénario à dix nœuds. Les mêmes considérations de couverture ont été prises entre les deux modèles de propagation.

5.7 Scénario 5 et 6

Dans ces deux scénarios, le réseau se compose de 25 nœuds répartis aléatoirement. Pour le scénario 6, l'émetteur suit une trajectoire à une vitesse de 15 m/s autour du réseau, comme montré par la figure suivante (la ligne rouge a été dessinée manuellement pour montrer la trajectoire) :

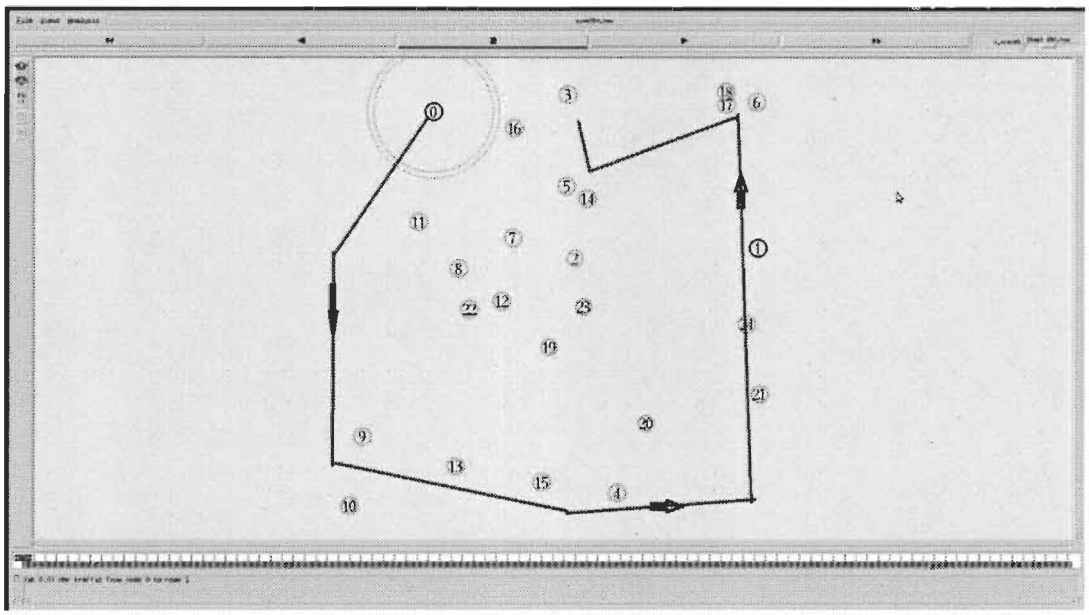

Figure 18- Topologie à 25 nœuds

5.8 Scénario dans un environnement interne sans, ensuite avec mobilité

Le scénario sans mobilité a été, à la fois, simulé pour tous les protocoles de routage (AODV, DSR et DSDV) et émulé pour le protocole AODV. En effet, Synapse Électronique dispose de modules intégrant le protocole AODV et le standard IEEE 802.15.4. Ce qui nous permet de valider nos résultats de simulation.

Dans le cas de la mobilité, le module d'adresse D1 se déplace à 0.8m/s tout au long de l'appartement (ce cas a été simulé seulement), comme montré dans la figure ci-dessus, où,

les flèches grises déterminent la source et la destination, les flèches bleues déterminent le mouvement du nœud D1.

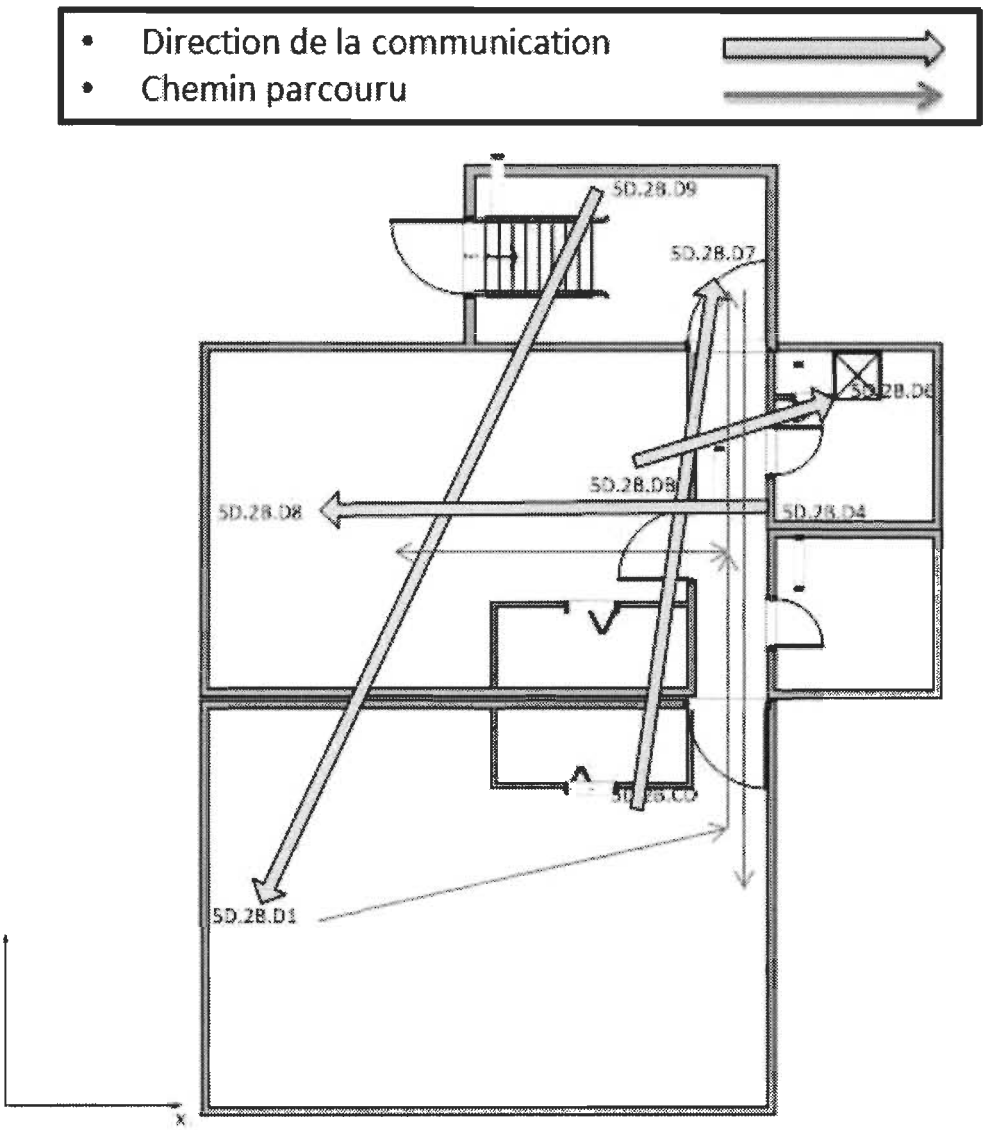

Figure 19- Topologie du réseau dans un environnement fermé

5.9 Critères d'évaluation des méthodes de routage

Trois méthodes de routage ont été simulées dans NS2, le protocole AODV, le protocole DSR et le protocole DSDV. La comparaison est effectuée en se basant sur quatre critères d'évaluation :

- **Le temps de découverte de la route :** Ce critère est applicable uniquement aux protocoles AODV et DSR. En effet, ce sont les seuls à entreprendre une découverte de route, alors que le DSDV entretient une table de routage au niveau de chaque nœud. Cette table est mise à jour à chaque 3 secondes d'intervalle (paramètre par défaut de l'algorithme). Donc, ce critère calcule l'intervalle entre le moment où la couche de réseau (routage) du nœud génère une requête, et le moment où la réponse est reçue par la même couche.

- **Le délai des paquets de données :** c'est le temps moyen que mettent tous les paquets transmis d'une source à une destination.

- **Le taux de réussite des paquets de données :** c'est le nombre de paquets reçus par la couche application de la destination sur le nombre de paquets envoyés par la couche d'application de la source.

- **La complexité de l'algorithme :** C'est le nombre de paquets de contrôle (de routage) nécessaires à l'envoi d'un paquet de données.

5.10 Résultats de simulations avec le modèle Two-Ray Ground

Voici les résultats des simulations des trois protocoles suivant les 6 scénarios de ce modèle de propagation.

- **Le temps de découverte de la route :**

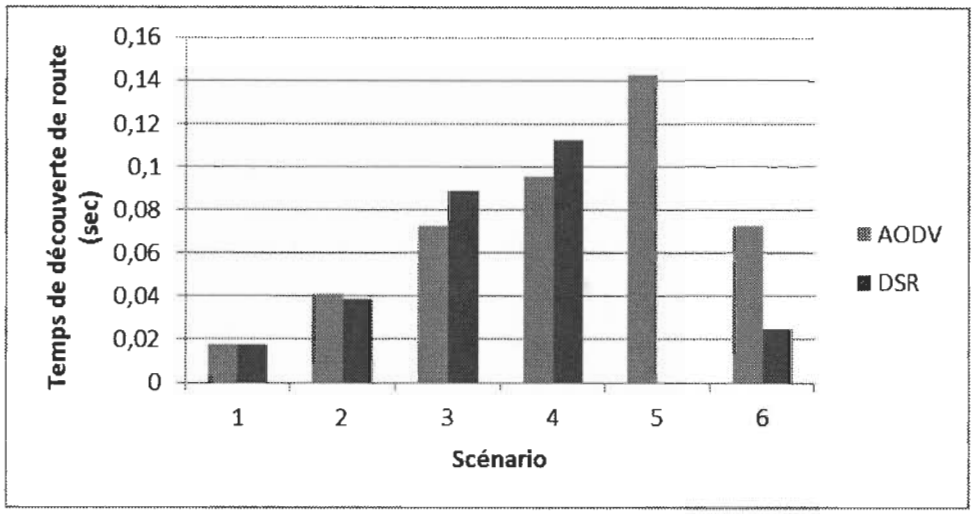

Figure 20- Le temps de découverte de la route (Two-Ray Ground)

Pour les deux premiers scénarios, ce temps est quasiment identique pour les deux protocoles. Dès que le nombre de nœuds augmente, à savoir les scénarios à 5 et 10 nœuds, ce temps est faible pour le protocole AODV. Pour le scénario à 25 nœuds immobiles (scénario 5), le protocole DSR n'entreprend aucune découverte de route.

Si on se réfère à la description du protocole dans le troisième chapitre de ce document, le protocole DSR génère une requête. Au fur et à mesure que cette requête est propagée dans le réseau, chaque nœud rajoute son adresse à l'entête du paquet. Ainsi, la taille du paquet augmente avec le nombre de hops. Or, puisque les paquets de la couche MAC du standard IEEE 802.15.4 ont un maximum de 128 octets (ce qui est relativement faible), un nombre de hops important génère des débordements dans la taille du paquet; et le protocole MAC ne peut plus supporter les paquets générés par la couche routage. C'est pour cela que pour le scénario à 25 nœuds, la méthode DSR ne trouve aucune route. De son côté, la méthode AODV a trouvé une route à 7 hops.

Pour le 6e scénario (25 nœuds avec mobilité de source), on peut remarquer que le DSR a trouvé une route vers la source, car selon la figure décrivant la nature du mouvement, la source se rapproche de la destination. Le temps de découverte est, à priori, à l'avantage du DSR. Cependant, ce temps représente la moyenne des temps de découverte des routes. Au début du mouvement, le DSR n'a trouvé aucune route, alors que l'AODV avait déjà une route avec un temps de découverte assez élevé. Avec la mobilité, l'AODV entreprend une nouvelle découverte à chaque fois qu'il perd sa route; et le DSR continue à chercher une route. Le temps de découverte de route de ce dernier est faible, car quand celui-ci a commencé à trouver des routes, la source était déjà proche de la destination.

- **Le délai des paquets de données :**

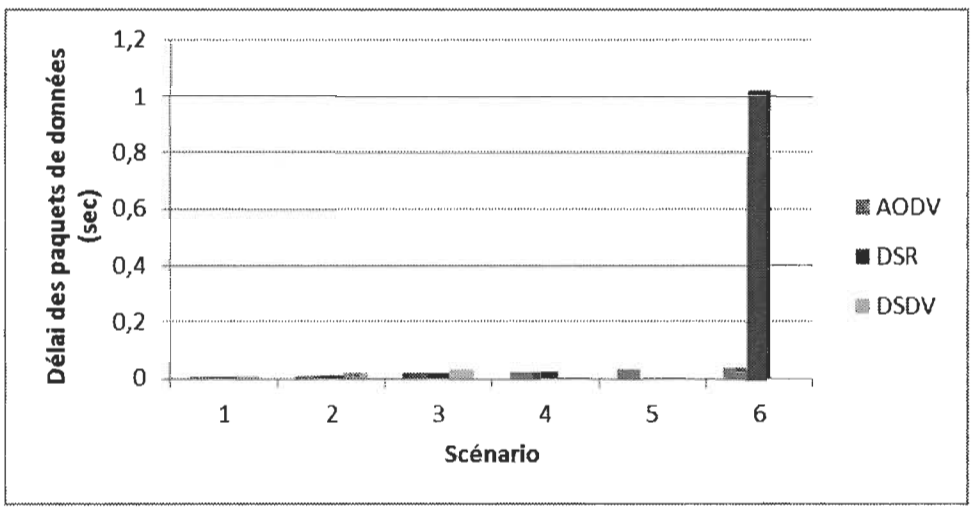

Figure 21- Délai des paquets de données (Two-Ray Ground)

On remarque que les délais de transmission des paquets sont relativement identiques pour les trois premiers scénarios. Pour le scénario à 10 nœuds (scénario 4) le DSDV ne peut plus envoyer de paquets, car les messages de mise à jour des tables de routage de chaque nœud dépassent la taille permise par le standard IEEE802.15.4 qui est de 128 octets.

Pour le scénario 5 le DSR n'a pas pu trouver de route vers la destination alors il n'y a aucune transmission de paquets. Pour le 6^{e} scénario, on remarque un très grand avantage dans le délai des paquets de données pour le protocole AODV.

- **Le taux de réussite des paquets de données :**

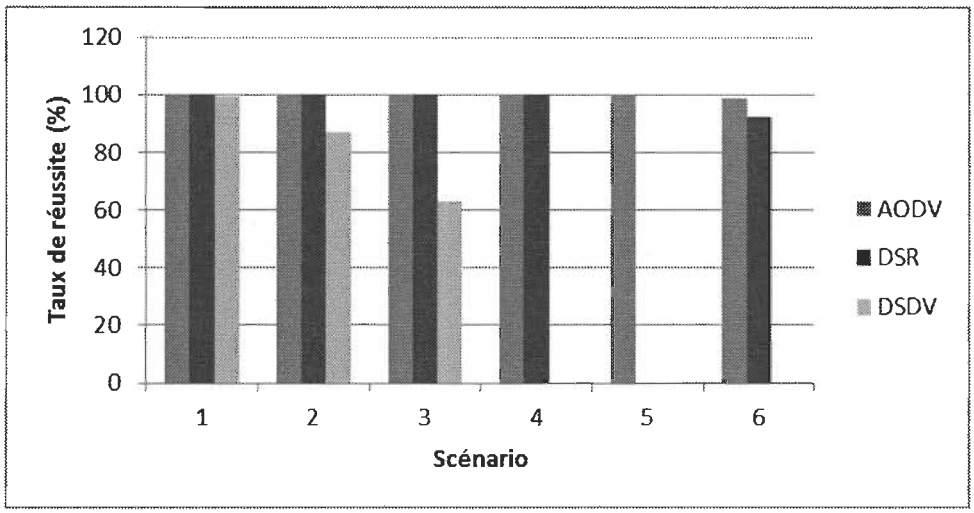

Figure 22- Le taux de réussite des paquets (Two-Ray Ground)

La même remarque que précédemment est à faire sur les scénarios 4 et 5 (DSDV n'a pas pu transmettre de paquets dans le 4^{e} scénario, ensuite le DSR dans le 5^{e} scénario). Pour le reste des scénarios, le taux de réussite est à 100% pour l'AODV pour les 5 premiers scénarios et à 98,6% pour le dernier, ce qui lui donne un avantage sur le reste des protocoles.

- **La complexité des algorithmes :**

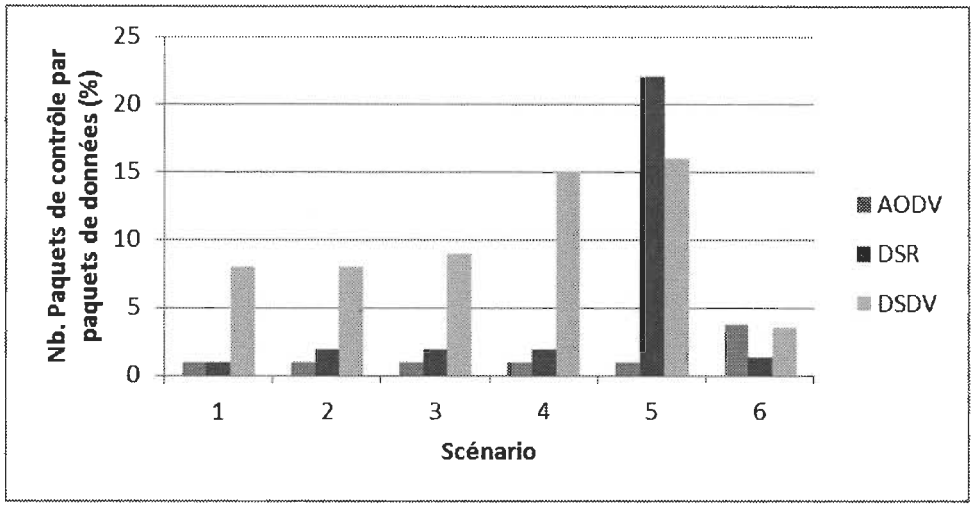

Figure 23- Complexité des algorithmes (Two-Ray Ground)

Pour les 5 premiers scénarios le protocole AODV produit le même nombre de paquets de contrôle pour envoyer 100 paquets (1 pour 100), indépendamment du nombre de hop, car la topologie ne change pas.

Le DSR garde aussi le même nombre de paquets de contrôle, mais seulement à partir de 3 nœuds. Pour le 5 e scénario, ce nombre explose, car le DSR continue de chercher une route pour les paquets produits par la couche d'application de la source.

La complexité du protocole DSDV est élevée, car elle dépend du nombre de nœuds dans le réseau, et non pas de la recherche de route puisque ce concept n'existe pas dans cette méthode.

5.1 Résultats de simulations avec le modèle Shadowing

La simulation du modèle de propagation Shadowing se rapproche du comportement réel d'un réseau sans fil. Selon la formule mathématique de ce modèle de propagation, l'estimation de la puissance d'un signal radio à un point donné dépend d'une variable

aléatoire. Ceci donne un aspect quasiment aléatoire de la propagation des signaux. En effet, pour avoir une tendance des critères d'évaluation d'un protocole, il faut répéter le même scénario plusieurs fois afin de décrire au mieux cette tendance.

Comme le montre la figure suivante, le temps de découverte de route du protocole AODV pour les quatre premiers scénarios (2, 3, 5, et 10 nœuds) sur 20 essais présente plusieurs variations.

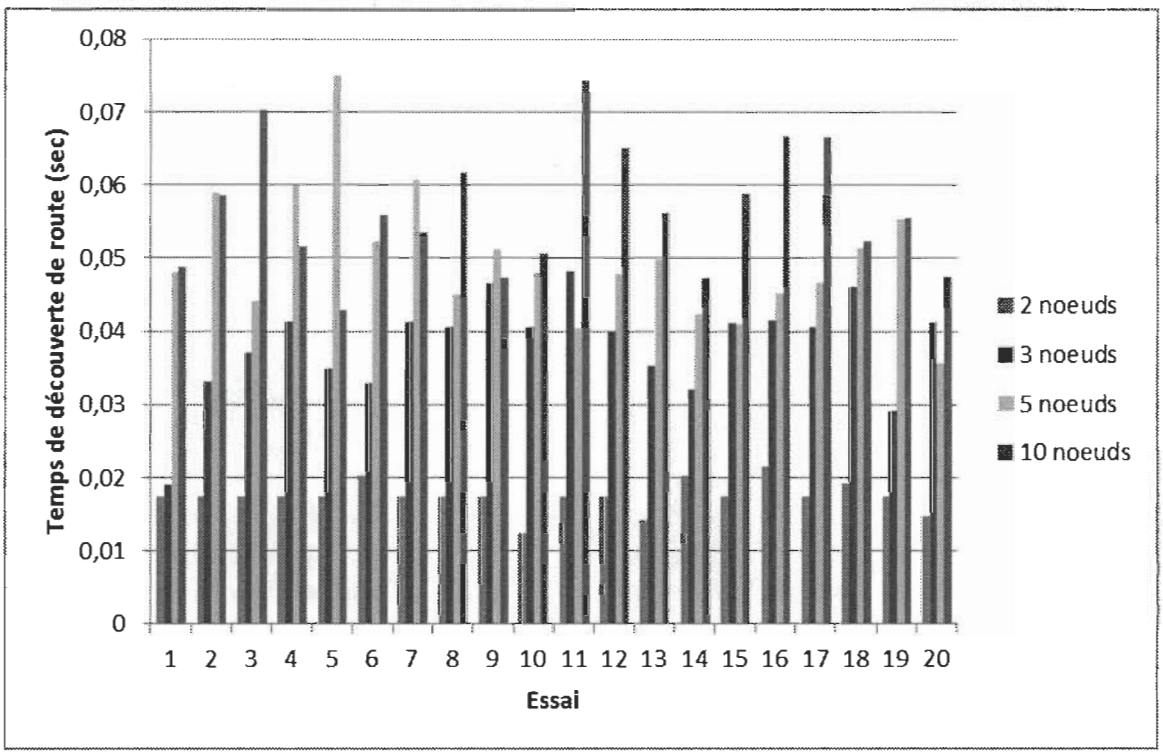

Figure 24- Temps de découverte de la route (AODV) (Shadowing) sur 20 essais

Pour chaque protocole, une série de tests a été effectuée (20 tests pour chaque critère) afin de sortir une moyenne de variation de chaque critère sur les 20 essais pour les 4 premiers scénarios.

- **Moyenne des temps de découverte de la route :**

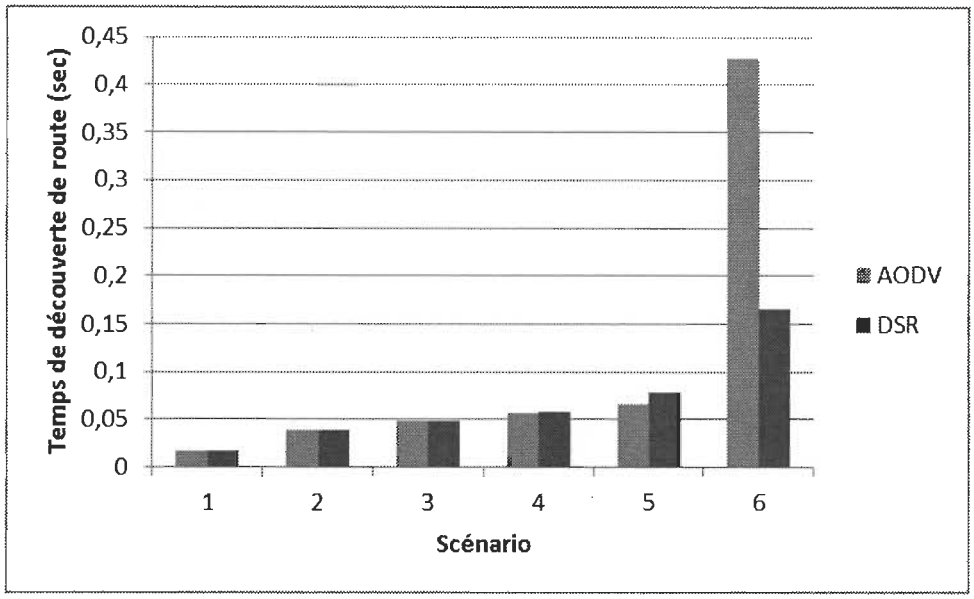

Figure 25- Moyenne des temps de découverte de route (Shadowing)

Cette moyenne est quasiment identique entre les deux protocoles et légèrement à l'avantage de l'AODV pour le scénario jusqu'au scénario à 10 nœuds. Pour les deux scénarios à 25 nœuds, la différence entre les deux protocoles est beaucoup plus visible. Le DSDV n'est pas concerné par la recherche de route.

Pour les deux scénarios à 25 nœuds, un comportement du réseau assez intéressant est à noter; la disposition des nœuds (la topologie) influence de façon énorme les performances à estimer. Ceci se voit même si on changeait la position d'un seul capteur de quelques centimètres. À cause de la nature aléatoire du canal de propagation, il est quasiment impossible de créer une topologie d'un réseau de capteurs sans fil disposant d'un grand nombre de nœuds (dans notre cas 25 nœuds) sans passer par un outil de modélisation de la propagation beaucoup plus avancé, qui prend en compte les paramètres de la propagation et ceux de la couche physique des nœuds de capteurs (puissance, fréquence,…).

- **Moyenne des délais des paquets des données :**

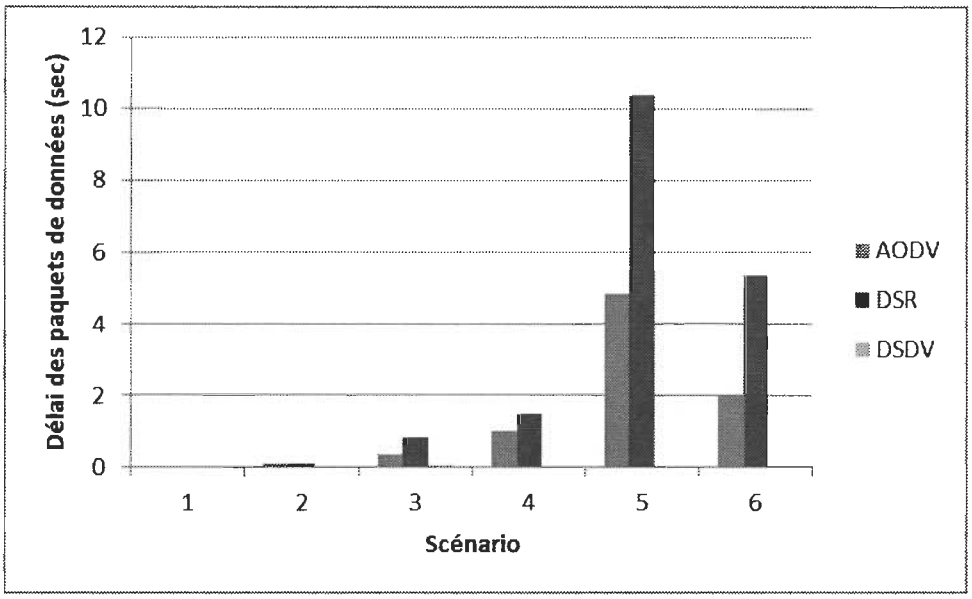

Figure 26- Moyenne des délais des paquets de données (Shadowing)

De première vue, le DSDV pourrait être considéré comme ayant le meilleur délai pour les scénarios 3, 4, 5 et 6. Cependant, si on se réfère au taux de réussite des paquets (graphe suivant), on remarque que le DSDV a envoyé peu de paquets par rapport aux autres. Donc pour le peu de paquets qu'il a réussi à envoyer, la moyenne des délais est relativement faible. En plus, sur les 20 essais, 14 n'ont présenté aucune transmission pour le DSDV (0% de taux de réussite). Pour les deux autres protocoles, on remarque un avantage considérable pour le protocole AODV.

- **Moyenne des taux de réussite des paquets :**

L'AODV présente un avantage clair lorsque le nombre de nœuds augmente en comparaison avec le DSR. Le DSDV présente des débordements dans la taille des paquets lorsque la taille du réseau augmente. Ceci explique la chute du taux de réussite des paquets.

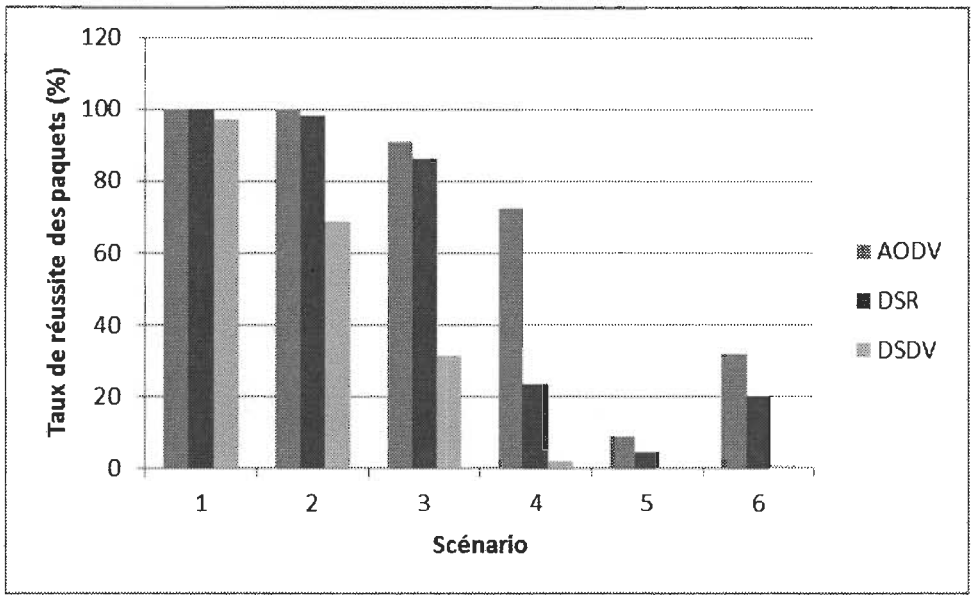

Figure 27- Moyenne des taux de réussite des paquets (Shadowing)

- **Complexité moyenne des algorithmes :**

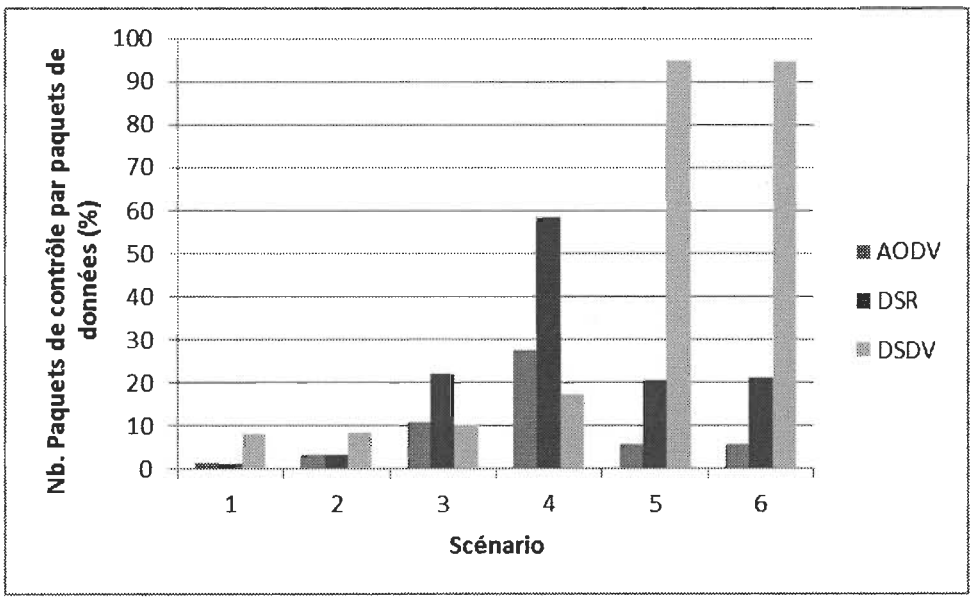

Figure 28- Complexité moyenne des algorithmes (Shadowing)

La différence de complexité entre le DSR et l'AODV se voit à partir du 3^{e} scénario; alors que la complexité moyenne du protocole DSDV augmente avec l'augmentation de la taille du réseau.

5.2 Résultats de simulations dans un environnement interne

Jusqu'à présent tous les réseaux simulés disposent d'une seule source et une seule destination; or dans la réalité ceci est rarement le cas. Dans ce genre de cas une source peut se transformer en un relai pour une autre source. Ceci la force à reconsidérer ses priorités (relayer ou envoyer ses paquets); et risque d'augmenter les délais de transmission ainsi que la complexité du routage.

Pour ces simulations, le modèle de propagation reste le Shadowing avec 20 essais pour chaque scénario et pour chaque critère. Dans les cas qui vont suivre, le scénario 7 est le scénario de maison sans mobilité, le scénario 8 est celui avec mobilité.

- **Temps de découverte de route :**

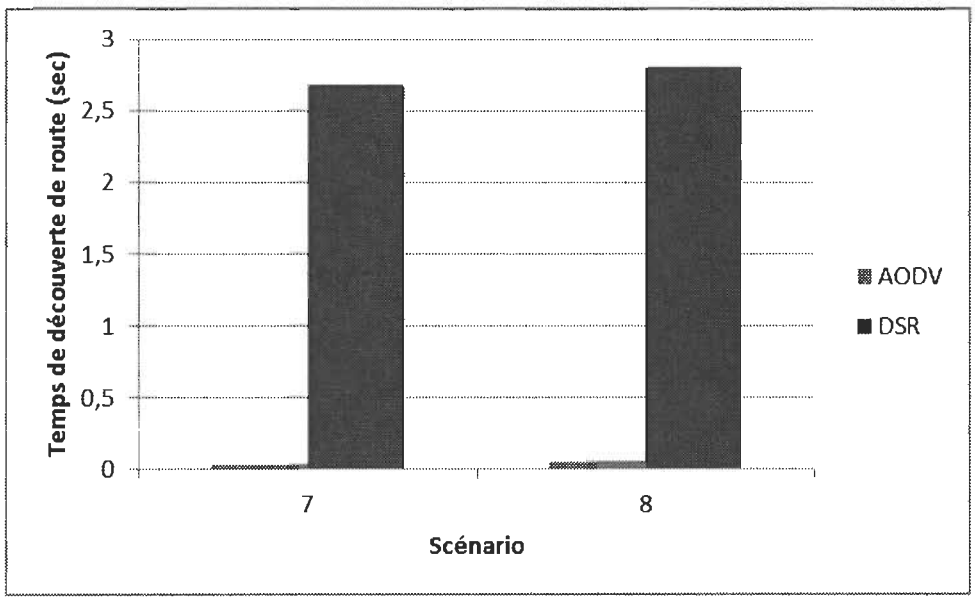

Figure 29- Temps de découverte de route (environnement interne)

La différence est considérable entre le protocole AODV et le protocole DSR.

- **Délai des paquets de données :**

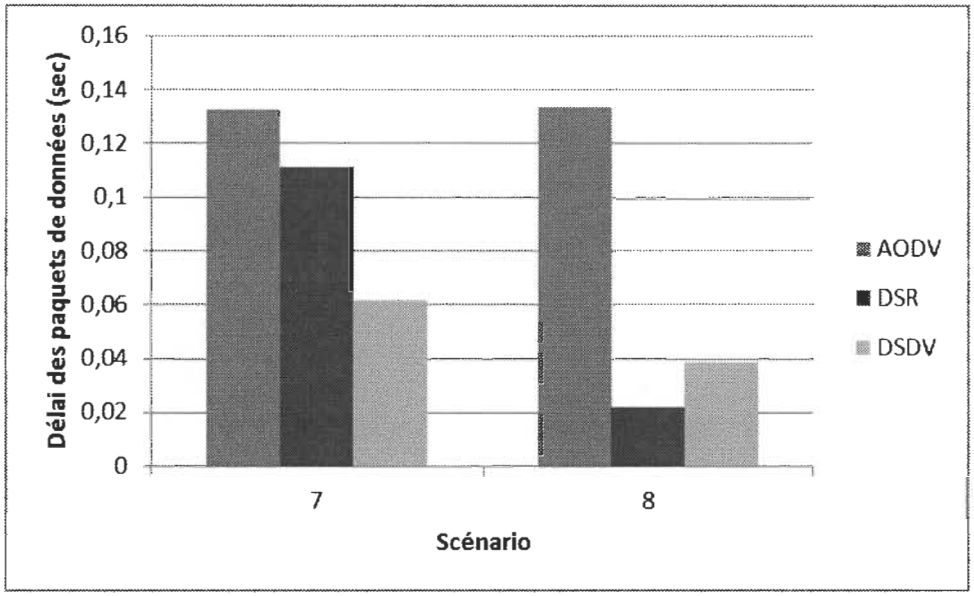

Figure 30- Délai des paquets de données (environnement interne)

Dans ce cas le DSR et le DSDV présentent un avantage certain par rapport à l'AODV, avec quand même un taux de réussite de paquets assez élevé pour le DSDV et le DSR. Comme cela a été dit au début, avant ces deux derniers scénarios, l'AODV présentait les meilleures performances par rapport aux autres protocoles. Or, à la présence de plusieurs sources et plusieurs destinations, la complexité de l'algorithme AODV augmente.

Selon le fonctionnement théorique de l'AODV, quand une source désire envoyer un paquet à une destination à une distance qui nécessite plusieurs hops, la source propage une requête dans le réseau. Le premier chemin qui aboutit à la réception de cette requête par la destination est suivi pour répandre la réponse. Dans ce chemin précisément, chaque nœud sauvegarde une liste de prédécesseur et de suiveur pour cette route. Autrement dit, chaque nœud dans cette route connait le hop précédent et le hop suivant pour cette route (cette route est identifiée par la source, la destination, et le numéro de séquence, voire chapitre 3

pour plus de détails). À ce point-là, on peut dire que la source ne connait pas toute la route vers la destination mais c'est les nœuds intermédiaires qui s'en occupent. Ceci présente un avantage pour les réseaux simple (avec une faible densité de trafic). Mais dès qu'il y a plusieurs sources et plusieurs destinations, les nœuds peuvent être l'intermédiaire de plusieurs routes (encore une fois, chaque route est identifiée et sauvegardée au sein du nœud intermédiaire, par la source, la destination et le numéro de séquence). Donc un nœud intermédiaire doit entretenir une table de prédécesseurs et de suiveurs pour plusieurs routes et doit mettre à jour cette table-là régulièrement, et surtout lors des erreurs de routage. En effet, un nœud intermédiaire doit avertir « en personne » la source lors d'une rupture de liaison dans un routage, car la source ne connait pas la route et n'a aucune idée du déroulement du routage.

Tout ceci augmente la complexité de l'algorithme et les délais de transmission. Car non seulement il faut propager les requêtes, les réponses, et les données, mais aussi, il faut constamment vérifier l'état des routes et propager une erreur vers la destination au besoin; et par la suite remettre à jour sa table de prédécesseurs et de suiveur pour cette route.

Pour le DSR, la route est sauvegardée dans l'entête des paquets. Le DSDV, quant à lui, sauvegarde une table de routage qu'il remet à jour régulièrement. C'est un inconvénient dans les réseaux à forte densité mais pour un faible nombre de nœuds et une forte complexité de trafic, ces inconvénients peuvent se transformer en avantages par rapport à l'AODV, car le routage est entretenu seulement au niveau des sources.

- **Taux de réussite des paquets :**

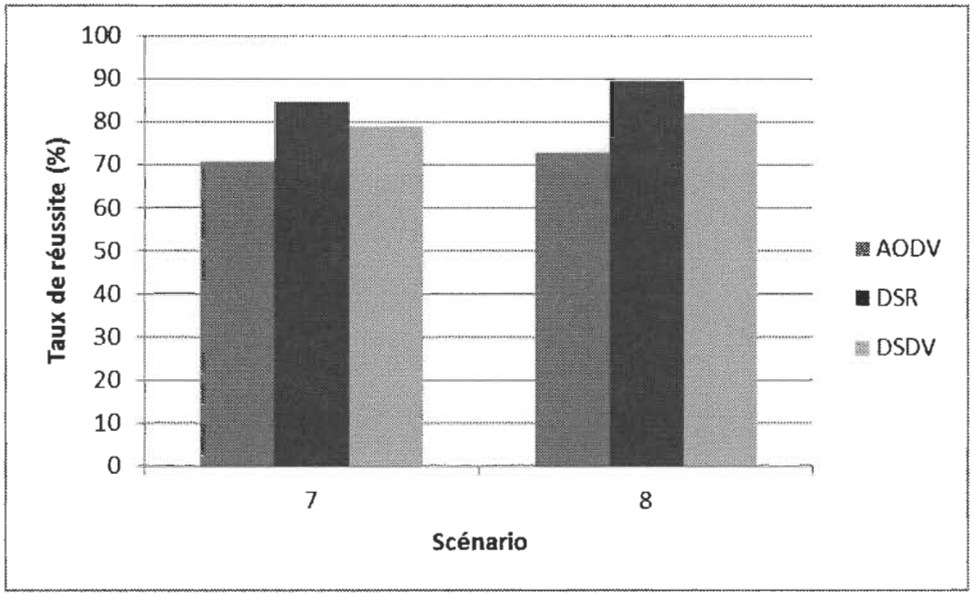

Figure 31- Taux de réussite des paquets (environnement interne)

Les remarques faites sur les délais des paquets entre les trois protocoles se vérifient de plus en plus avec les taux de réussite. En effet, il existe un léger avantage pour le DSR et le DSDV.

- **Complexité des algorithmes :**

La complexité de l'AODV est importante pour les mêmes raisons que précédemment. Celle du DSDV l'est parce que chaque nœud doit mettre à jour sa table de routage régulièrement et donc génère des paquets de routage de façon considérable. Le DSR s'adapte mieux aux petits réseaux complexes.

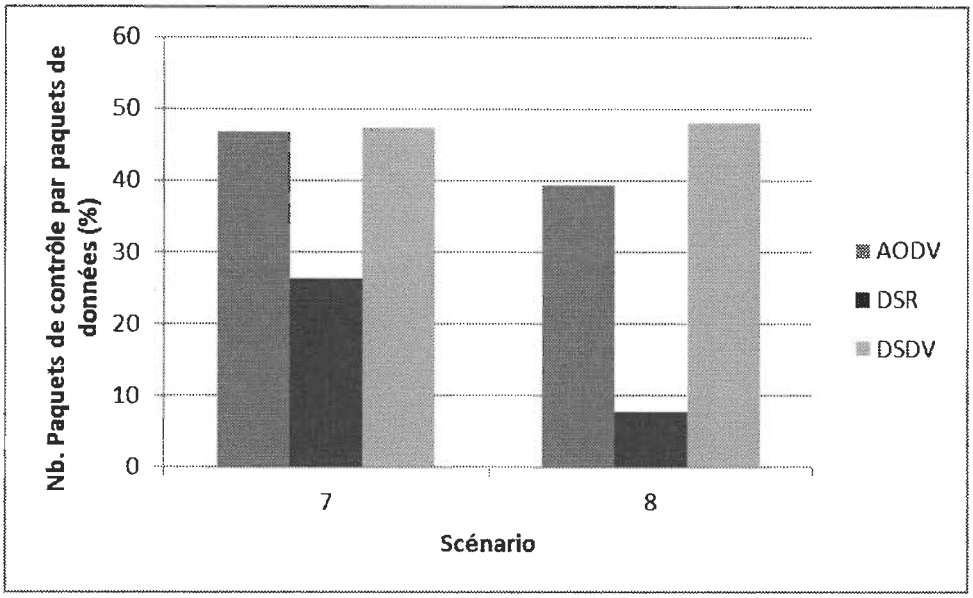

Figure 32- Complexité des algorithmes (environnement interne)

5.3 Émulation de l'AODV dans un environnement interne et comparaison avec la simulation

L'émulation de l'AODV a été effectuée par un autre étudiant en se basant sur ses études du matériel fourni par Synapse Électronique et les études et simulations réalisées dans ce projet. Ces émulations vont permettre de valider la simulation par et vice versa.

Deux scénarios ont été créés pour cette comparaison, et sont donnés respectivement dans les deux figures ci-dessous. Le premier scénario représente les communications locales, où l'émetteur et le récepteur se trouvent dans la même pièce. Le second scénario représente le cas où l'émetteur et le récepteur se trouvent dans des pièces différentes.

L'émulation s'est faite à un paquet par seconde, les instants de démarrage et d'arrêt de chaque communication dans la simulation ont été adaptés à ceux de l'émulation. L'émulation a été refaite 20 fois afin d'avoir une meilleure représentation des tendances de changements des performances du réseau. Les résultats sont donnés dans 20 fichiers de 600

lignes (12000 lignes !). À notre connaissance, il est impossible d'automatiser l'évaluation des critères de performance du protocole pour le type de fichiers générés par le logiciel de Synapse.

Pour le délai des paquets de données, il faudrait vérifier ligne par ligne les presque 12000 lignes manuellement les paquets envoyés et reçus afin de calculer les délais pour chaque couple de capteurs et enfin de calculer une moyenne, or il est clair que c'est une tâche énorme (à savoir que pour les résultats du simulateur, c'est le programme AWK qui s'en occupe).

Les paramètres de propagation du modèle Shadowing ont été variés pour vérifier l'influence de ces changements sur les critères de performances, et ainsi de nous permettre de trouver la meilleure combinaison du Beta et du Sigma pour mieux modéliser l'environnement de propagation en fonction des résultats de l'émulation.

Le premier cas : Beta = 5 et Sigma = 7; le deuxième cas : Beta = 4 et Sigma = 7; et le troisième cas : Beta = 1.7 et Sigma = 5. Le chapitre 2 décrit la signification de chaque cas.

5.3.1 Communications locales

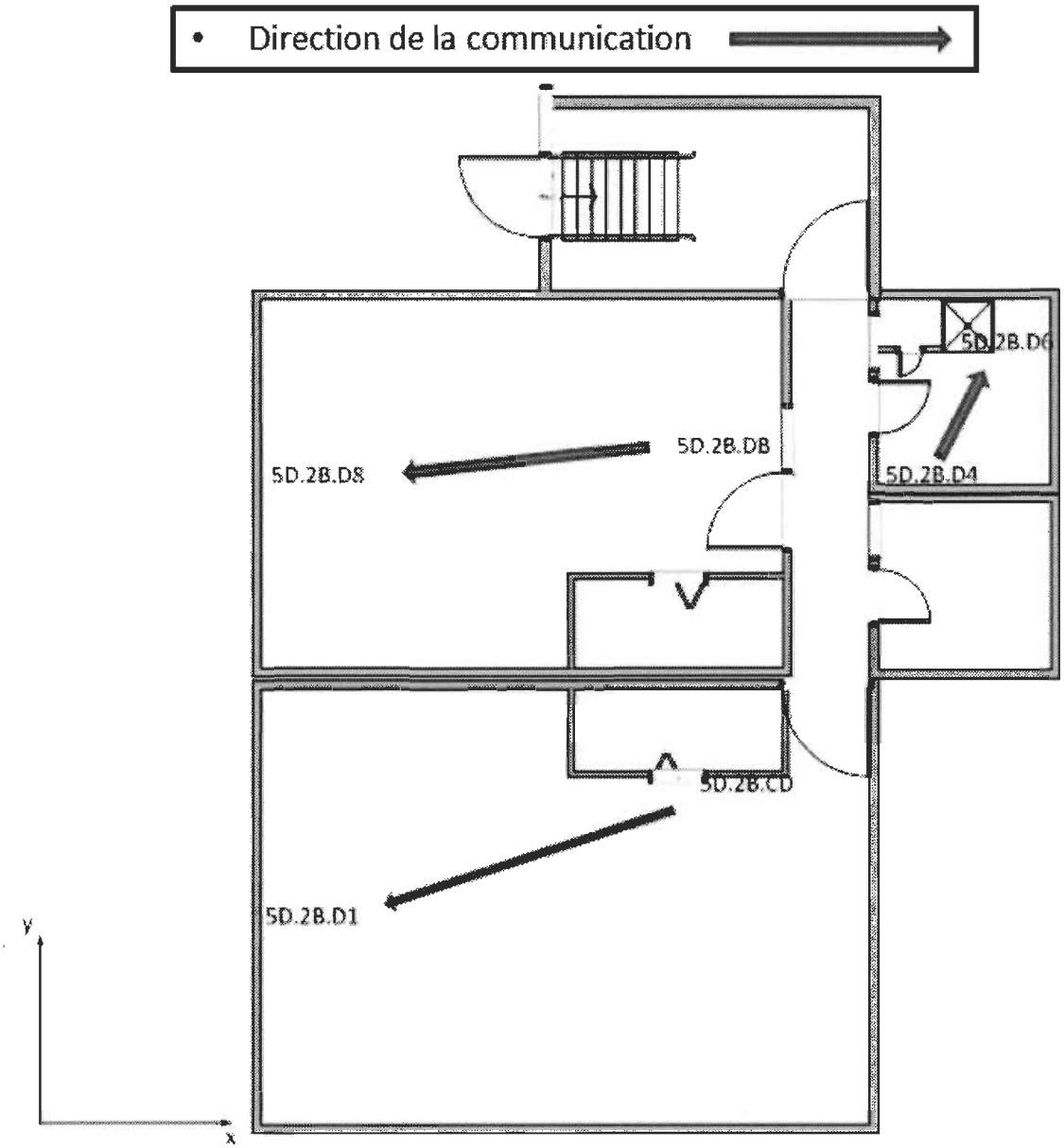

Figure 33- Plan des communications locales

- **Temps de découverte de route :**

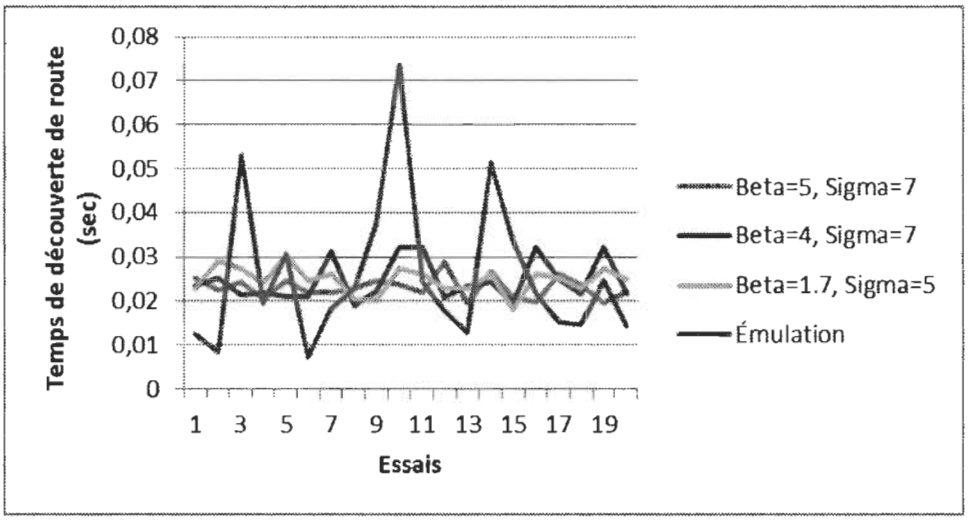

Figure 34- Temps de découverte des routes (Simulations Vs. Émulation)

Le changement des paramètres de propagation influence peu le temps de découverte des routes dans les simulations. L'émulation présente quasiment la même moyenne que la simulation (cas 1 : 0.023ms, cas 2 : 0.024 ms, cas 3 : 0.024 ms, Émulations : 0.025 ms).

- **Délai des paquets de données :**

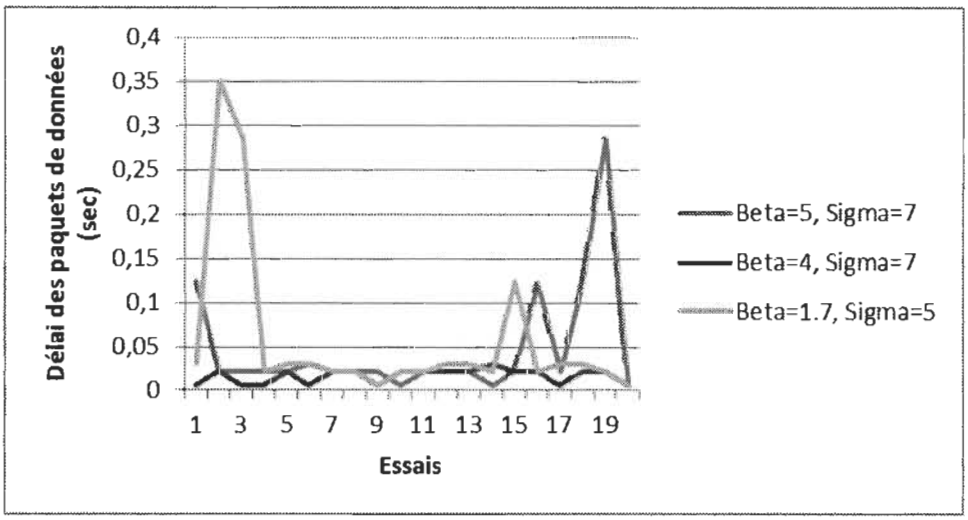

Figure 35- Délais des paquets de données pour la simulation des communications locales

Pour l'émulation, le maximum ne dépasse pas les 2 ms et le minimum est de 1ms pour les paquets de données. Pour les paquets de données de la simulation, le cas 2 (Beta = 4, Sigma = 7) présente quelques valeurs des délais proches de 5 ms avec une moyenne de 16 ms pour les 20 tests. Les autres cas sont loin de l'émulation (48ms pour le cas 1, et 58 ms pour le cas 3). On peut conclure qu'il faudrait une meilleure estimation des critères de propagation afin d'approcher au mieux les performances de l'émulation.

Taux de réussite des paquets :

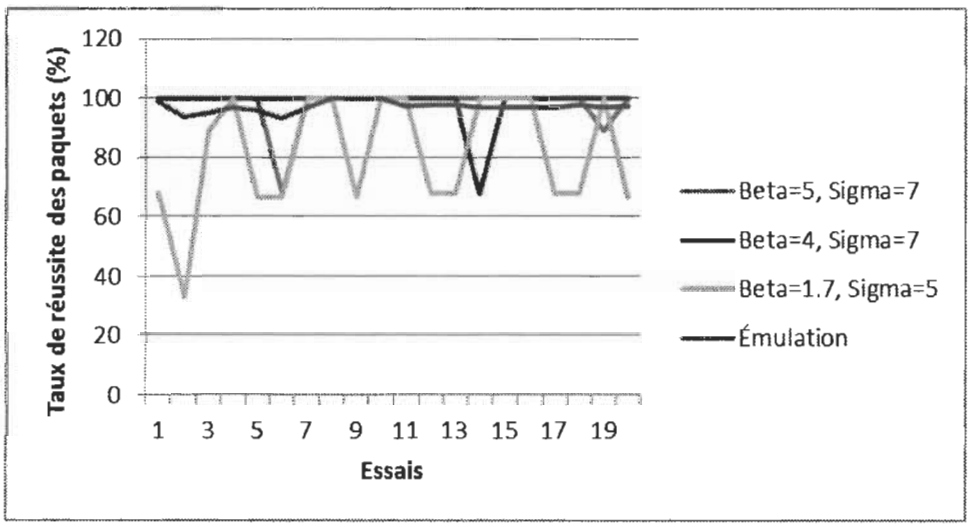

Figure 36- Taux de réussite des paquets de données (Simulations Vs. Émulation)

Le taux de réussite des paquets de données de l'émulation est très élevé et suit la tendance du cas 2 de la simulation.

5.3.2 Communications distantes

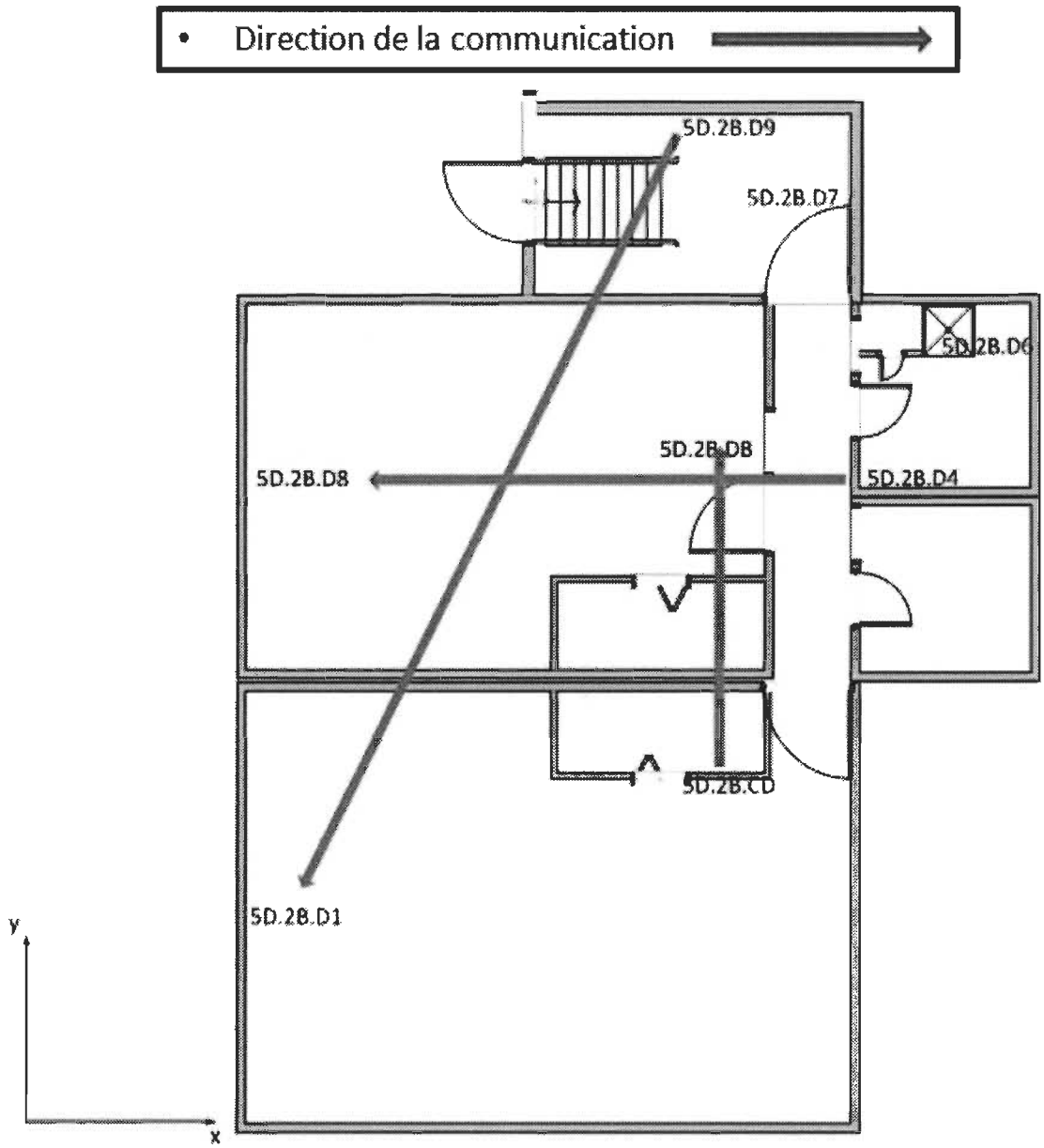

Figure 37- Plan des communications distantes

- **Temps de découverte de route :**

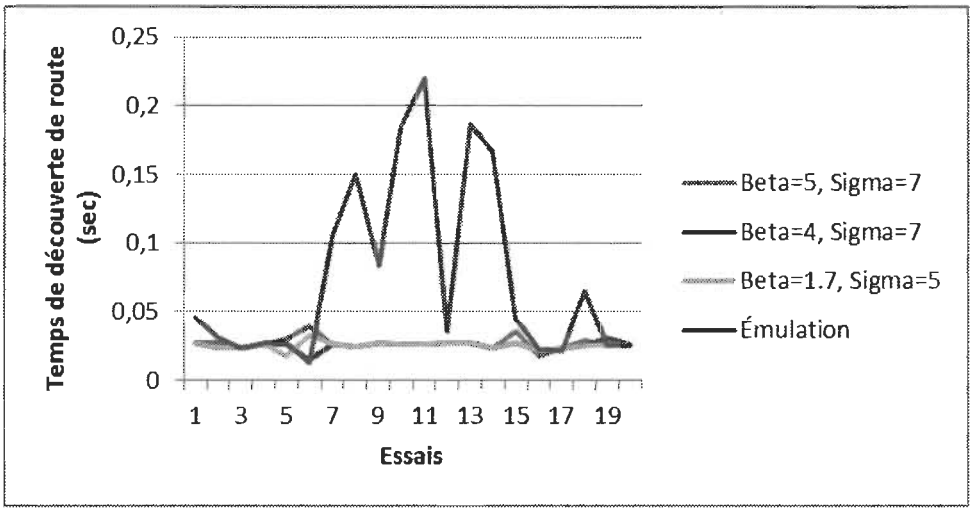

Figure 38- Temps de découverte des routes (Simulations Vs. Émulation)

Le cas présent confirme à travers la figure ci-dessus la nécessité d'avoir un modèle de modélisation de la propagation qui permet à la simulation de refléter le cas réel et vice versa.

- **Délai des paquets de données :**

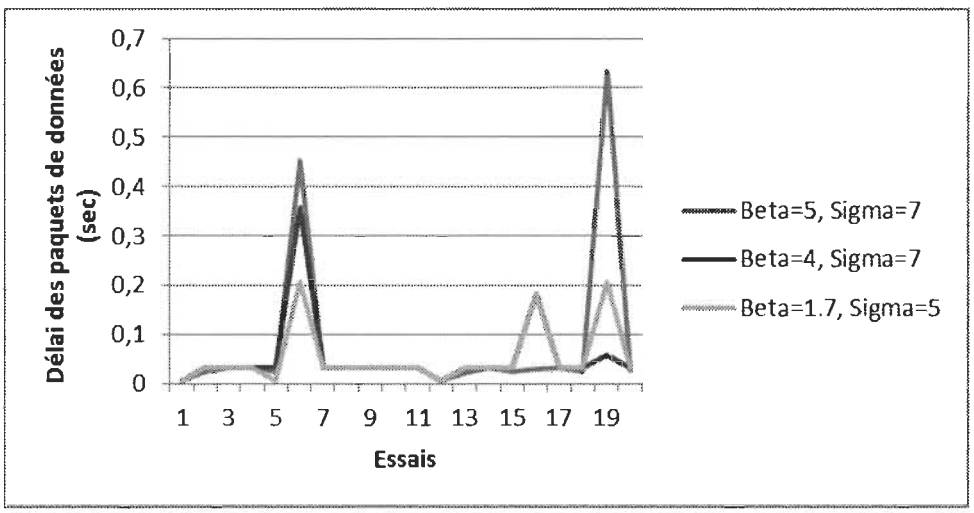

Figure 39- Délais des paquets de données pour la simulation des communications distantes

Les délais des paquets de données pour l'émulation varient entre 1 et 20 ms. La moyenne des paquets de données des trois cas de la simulation sont, respectivement, 77 ms, 53 ms, et 52 ms.

Taux de réussite des paquets :

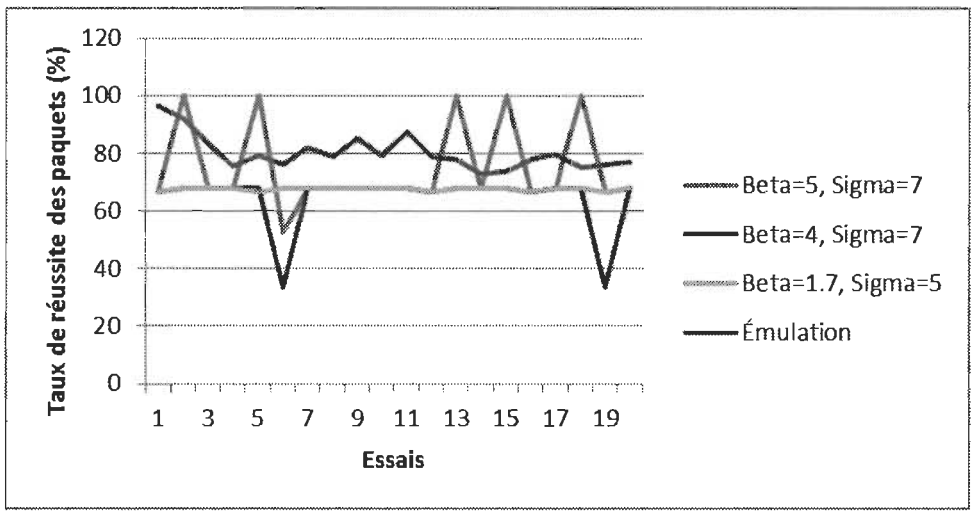

Figure 40- Taux de réussite des paquets de données (Simulation Vs. Émulation)

Les taux de réussites de l'émulation avec une moyenne de 80.2% restent proches du cas 1 qui a un taux moyen de 77.2%, pour les deux autres cas les taux moyens sont respectivement 64.1% et 67.49%.

5.4 Conclusion

Les différentes simulations réalisées ont révélé des différences remarquables dans le comportement des différents protocoles. Concernant les critères de performance choisis dans ce projet, et dans des réseaux simples à une source et une seule destination (c.à.d. le flux des paquets est faible), l'AODV se démarque par sa rapidité dans la découverte des routes et dans la transmission d'un maximum de paquets de données. Sa complexité est plus faible par rapport aux deux autres protocoles. Le DSR et le DSDV présentent le défaut

de débordement de la taille des paquets lorsque le nombre de sauts dépasse une certaine limite, ce qui est lié à leurs fonctionnements. Ceci est bien expliqué dans la description théorique dans le chapitre 3. Lorsque le réseau devient plus complexe (le flux est plus important), le protocole AODV montre des lacunes dans la maintenance des routes. Ceci s'explique par le fait que c'est aux nœuds intermédiaires de signaler à la source la rupture des liens, ce qui peut provoquer des délais et une surcharge du canal avec les messages d'erreurs. L'entretien des routes dans le DSR et le DSDV se fait au niveau de la source elle-même, ce qui la libère de la congestion du réseau. Ceci se voit clairement dans l'estimation des critères de performances des trois protocoles.

Concernant l'environnement de simulation, NS2 offre une modélisation du canal de propagation qui reste très limitée lorsque l'environnement présente des obstacles. Le simulateur NS2 (ou tout autre simulateur *open source*) n'offre pas d'outils pour créer la meilleure topologie pour un réseau à grand nombre de nœuds, ceci nous pousse à essayer de créer manuellement une topologie et de modifier la position de chaque nœud pour assurer la meilleure couverture. Or, lorsque le réseau est énorme, ceci peut être très approximatif comme procédure.

La simulation nous a permis d'établir une série de recommandations aux contraintes imposées par le projet. En effet, les deux premières parties des simulations (*Two Ray Ground* et *Shadowing*) ont montré que le fonctionnement de l'AODV en terme de délai d'établissement du routage, de transmission de paquets, de taux de réussite et de complexité, reste le meilleur compromis comparé avec le DSR et le DSDV. Cependant, l'émulation montre que la simulation doit franchir un cap important afin de passer à la réalisation d'un réseau réel. Au-delà du côté fonctionnel du protocole, la prise en compte et

la modélisation de l'environnement de déploiement est une étape essentielle et primordiale pour pouvoir passer à l'industrialisation du réseau. La simulation nous a permis donc de faire fonctionner le réseau selon les critères de performance, la modélisation de l'environnement de propagation des signaux radio en prenant en compte l'architecture et les obstacles est la prochaine voie de recherche pour ce projet.

Chapitre 6 - Conclusion générale

Une étude comparative entre plusieurs protocoles de routage fonctionnant sous la couche MAC du standard IEEE802.15.4 a été réalisée dans ce travail de recherche. Les conditions de simulation considérées ont été les plus proches possible des conditions réelles. Pour cela, la maitrise théorique, d'abord, des différentes fonctionnalités des protocoles des couches OSI concernées ont été étudiées de façon exhaustive.

Le projet s'attaque à un problème de recherche avec une application industrielle réelle. Des scénarios de simulations réalistes ont été considérés à travers les contraintes matérielles et de propagation. Une validation par émulation a également été faite. En se basant sur ces contraintes, il est possible de présenter les recommandations adéquates à cette problématique. À ce propos, il est possible de constater que, globalement, l'algorithme AODV a présenté des résultats satisfaisants par rapport aux autres protocoles. Les autres protocoles présentent le défaut de se saturer lorsque le nombre de sauts dans une route dépasse une certaine limite, ce qui n'est pas le cas de l'AODV. Par contre, les délais et la complexité des autres protocoles sont nettement meilleurs lors des réseaux à faible nombre de capteurs, mais complexes (plusieurs sources et plusieurs destinations). Malgré cela, l'AODV reste le favori, du fait que la topologie du réseau de capteurs ne peut être prédite et donc le nombre de sauts dans un réseau reste imprévisible lors de l'implantation d'un réseau dans un environnement inconnu. De cette manière, en dépit d'un certain retard dans

les communications et d'une certaine complexité par rapport aux autres protocoles, le protocole ne se saturera pas.

Plusieurs protocoles ont été étudiés, simulés et comparés dans des scénarios proches de la réalité ce qui a donné une vision assez claire sur les contraintes et les limites de la fidélité de ces simulations. En effet, les comparaisons entre la simulation avec l'émulation montrent que la propagation des signaux radio dans un RCSF est l'aspect de simulation le plus difficile à simuler, due à la nature complètement aléatoire de la propagation, surtout en environnement fermé. Plusieurs modèles mathématiques ont été développés, certains ont été utilisés dans ce travail, mais cela ne prend pas en compte l'architecture de l'environnement interne (l'emplacement des murs et des obstacles, les matériaux de construction,…).

À un certain degré de la simulation du comportement d'un RCSF, le manque de réalisme dans la modélisation de la propagation dans les logiciels de simulation peut se révéler être une voie d'exploration possible pour de futurs projets. Pour cela, on pourrait penser à un logiciel prenant en compte tous les aspects de l'environnement (architecture, obstacle, matériaux,…) dans la modélisation de la propagation.

Ce travail ouvre la voie pour le développement d'un nouveau protocole de communication dans les réseaux de capteurs sans fil pour des applications résidentielles, et ce, en se basant sur les limites des autres protocoles existants.

Bibliographie

[1] I. F. Akyildiz. (2010). *Wireless sensor networks.*

[2] Q. Wang and I. Balasingham, "Wireless Sensor Networks - An Introduction," in *Wireless Sensor Networks: Application - Centric Design*, G. V. Merrett and Y. K. Tan, Eds., ed: InTech, 2010.

[3] C. Chee-Yee and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE,* vol. 91, pp. 1247-1256, 2003.

[4] M. Vieira, A. da Cunha, and D. da Silva, "Designing Wireless Sensor Nodes Embedded Computer Systems: Architectures, Modeling, and Simulation." vol. 4017, S. Vassiliadis, S. Wong, and T. Hämäläinen, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 99-108.

[5] P. Zhongmin, D. Zhidong, Y. Bo, and C. Xiaoliang, "Application-oriented wireless sensor network communication protocols and hardware platforms: A survey," in *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on*, 2008, pp. 1-6.

[6] M. R. Akhondi, A. Talevski, S. Carlsen, and S. Petersen, "Applications of Wireless Sensor Networks in the Oil, Gas and Resources Industries," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 2010, pp. 941-948.

[7] M. A. Hussain, P. khan, and S. Kwak kyung, "WSN research activities for military application," in *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, 2009, pp. 271-274.

[8] H. Furtado and R. Trobec, "Applications of wireless sensors in medicine," in *MIPRO, 2011 Proceedings of the 34th International Convention*, 2011, pp. 257-261.

[9] Z. Dai, S. Wang, and Z. Yan, "BSHM-WSN: A wireless sensor network for bridge structure health monitoring," in *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*, 2012, pp. 708-712.

[10] K. Sukun, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks," in *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, 2007, pp. 254-263.

[11] M. Z. A. Bhuiyan, C. Jiannong, and W. Guojun, "Deploying Wireless Sensor Networks with Fault Tolerance for Structural Health Monitoring," in *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, 2012, pp. 194-202.

[12] J. K. Notay and G. A. Safdar, "A Wireless Sensor Network based Structural Health Monitoring system for an airplane," in *Automation and Computing (ICAC), 2011 17th International Conference on*, 2011, pp. 240-245.

[13] D. Angela, M. Ghenghea, and I. Bogdan, "Supporting environmental surveillance by using wireless sensor networks," in *Electrical and Electronics Engineering (ISEEE), 2010 3rd International Symposium on*, 2010, pp. 216-219.

[14] H. Huiping, X. Shide, M. Xiangyin, and X. Ying, "A Remote Home Security System Based on Wireless Sensor Network and GSM Technology," in *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, 2010, pp. 535-538.

[15] H. Mong-Fong, S. Chien-Chou, H. Wen-Hsiung, and L. Li-Chen, "A Temperature Surveillance System Based on Zigbee Technology for Blaze Detection," in *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, 2009, pp. 1277-1280.

[16] D. Baghyalakshmi, J. Ebenezer, and S. A. V. Satyamurty, "WSN based temperature monitoring for High Performance Computing cluster," in *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, 2011, pp. 1105-1110.

[17] G. H. E. L. de Lima, L. C. e Silva, and P. F. R. Neto, "WSN as a Tool for Supporting Agriculture in the Precision Irrigation," in *Networking and Services (ICNS), 2010 Sixth International Conference on*, 2010, pp. 137-142.

[18] O. Sekkas, S. Hadjiefthymiades, and E. Zervas, "A Multi-level Data Fusion Approach for Early Fire Detection," in *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, 2010, pp. 479-483.

[19] W. Charfi, M. Masmoudi, and F. Derbel, "A layered model for wireless sensor networks," in *Systems, Signals and Devices, 2009. SSD '09. 6th International Multi-Conference on*, 2009, pp. 1-5.

[20] "IEEE Standard for Information technology-- Local and metropolitan area networks-- Specific requirements-- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1-320, 2006.

[21] L. Ruizhong, W. Zhi, and S. Youxian, "Energy efficient medium access control protocols for wireless sensor networks and its state-of-art," in *Industrial Electronics, 2004 IEEE International Symposium on*, 2004, pp. 669-674 vol. 1.

[22] Y. Tselishchev, A. Boulis, and L. Libman, "Experiences and Lessons from Implementing a Wireless Sensor Network MAC Protocol in the Castalia Simulator,"

in *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, 2010, pp. 1-6.

[23] M. A. Labrador and P. M. Wightman. (2009). *Topology Control in Wireless Sensor Networks*.

[24] I. F. Akyildiz, T. Melodia, and K. R. Chowdury, "Wireless multimedia sensor networks: A survey," *Wireless Communications, IEEE,* vol. 14, pp. 32-39, 2007.

[25] T. V. Project, *The ns Manual*, 2011.

[26] *Synapse Électronique*. Available: http://www.synapseelectronique.com

[27] F. Yu, "Self organization in medium access control for wireless ad hoc and sensor networks," Ph.D. 3348237, Michigan State University, United States -- Michigan, 2008.

[28] T. Camilo, J. S. Silva, and F. Boavida, "Assessing the use of ad-hoc routing protocols in Mobile Wireless Sensor Networks."

[29] G. Sriram and D. S. Rao, "Performance Analysis of Ad Hoc Routing Protocols in WSNs," *International Journal of Computer Science & Engineering Technology,* vol. 2, 5 May 2011 2011.

[30] D. Goyal and M. R. Tripathy, "Routing Protocols in Wireless Sensor Networks: A Survey," in *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on*, 2012, pp. 474-480.

[31] L. Abusalah, A. Khokhar, and M. Guizani, "A survey of secure mobile Ad Hoc routing protocols," *Communications Surveys & Tutorials, IEEE,* vol. 10, pp. 78-93, 2008.

[32] T. P. Lambrou and C. G. Panayiotou, "A Survey on Routing Techniques Supporting Mobility in Sensor Networks," in *Mobile Ad-hoc and Sensor Networks, 2009. MSN '09. 5th International Conference on*, 2009, pp. 78-85.

[33] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing draft-ietf-manet-aodv-13," INTERNET DRAFT17 February 2003 2003.

[34] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) draft-ietf-manet-dsr-10," INTERNET-DRAFT2004.

[35] V. Park and S. Corson, "draft-ietf-manet-tora-spec-01 Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification," 1998.

[36] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *SIGCOMM Comput. Commun. Rev.,* vol. 24, pp. 234-244, 1994.

[37] A. Amuthan and D. N. Abirami, "Multicast Security Attacks and Its Counter Measures for Puma Protocol," *IJCTA,* vol. 2, pp. 594-600, 2011.

[38] R. K. Nagra, J. S. Gurm, and G. S. Grewal, "Simulation based Analysis of AODV, BABEL and PUMA Protocols for Adhoc Network," presented at the IJCA, 2012.

[39] R. Vaishampayan, "EFFICIENT AND ROBUST MULTICAST ROUTING IN MOBILE AD HOC NETWORKS," DOCTOR OF PHILOSOPHY, UNIVERSITY OF CALIFORNIA, SANTA CRUZ, 2006.

[40] A. Banerjee and P. Dutta, "A Survey of Multicast Routing Protocols For Mobile Ad Hoc Networks," *IJEST,* vol. 2, pp. 5594-5604, 2010.

[41] N. I. Australia. Available: http://www.nicta.com.au/

[42] A. Stetsko, M. Stehlik, and V. Matyas, "Calibrating and Comparing Simulators for Wireless Sensor Networks," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, 2011, pp. 733-738.

[43] L. Peng, J. Zeng, H. Yuan, and H. Li, "WSM: Introduction, Design and Case Study," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 2007, pp. 2580-2583.

[44] *Tutorial for the Network Simulator ns.* Available: http://www.isi.edu/nsnam/ns/tutorial/index.html

[45] OMNeT++. Available: http://omnetpp.org/

[46] A. Boulis, "Castalia A simulator for Wireless Sensor Networks and Body Area Networks Version 3.2 User's Manual," NICTA, Ed., ed, 2011.

[47] R. Ogier, F. Templin, and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)," 2004.

Annexe A – Le standard IEEE 802.15.4 (couches PHY et MAC)

La couche Physique (PHY)

Un paquet de la couche PHY est appelé PPDU (PHY Protocol Data Unit)

		Octets		
		1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Figure 41- Structure de la trame de la couche PHY

La trame PPDU comporte quatre parties :

- SHR : *Synchronisation Header*, permet au dispositif de se synchroniser. Ce sont les 5 premiers octets de la trame et correspondent à une entête de synchronisation (*Preamble + Start of packet delimiter*). L'octet « *Start of packet delimiter* » permet de spécifier la fin du préambule.
- PHR : *PHY Header*, contient les informations sur la longueur de la trame
- PHY payload : contient la cargaison de la trame de la couche MAC.

La couche MAC

Il existe 4 structures de paquet au niveau MAC :

- Trame de données

- Trame « Beacon » (trame phare)
- Trame d'acquittement (ACK)
- Trame de commande

Les trames de données et de Beacon proviennent ou sont destinées aux couches supérieures. Les deux autres structures de trames sont générées et ne sont utilisées que par la couche MAC.

Trame de données

Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Data Payload	FCS
MHR				MAC Payload	MFR

Figure 42- Structure de la trame de données

La trame complète est de 127 bytes. Elle contient un en-tête (MHR), des données provenant des couches supérieures (MSDU) et une fin de séquence (MFR).

- Frame Control (2 octets)

Commun à tous les types de trames et sert à spécifier la structure et le contenu du reste de la trame.

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figure 43- Structure du champ Frame Control

- Frame type (3 bits) : Ce champ sert à définir le type de trame.

Frame type value $b_2\ b_1\ b_0$	Description
000	Beacon
001	Data
010	Acknowledgment
011	MAC command
100–111	Reserved

Figure 44- Différents types de trames

- Security enabled (1 bit) : activation de la sécurisation des trames
- Frame pending (1 bit) : Quand ce bit est actif, cela indique au destinataire que des données sont encore présentes pour lui.
- Ack. Request (1 bit) : Demander au récepteur d'acquitter les données reçues.
- PAN ID compression (1 bit) : Ce bit permet d'indiquer si la trame doit être envoyée dans le même PAN (intra PAN) ou sur un autre PAN (inter PAN).
- Source addressing mode (2 bits) et Destination addressing mode (2 bits) : Définissent le mode d'adressage : court (16 bits) ou étendu (64 bits).

- Sequence Number (1 octet)

Ce champ définit une numérotation de trames sur 8 bits, afin de connaître quelles trames ont été acquittées pour éviter la duplication des paquets et ainsi de ne pas recevoir le même paquet plusieurs fois.

- Addressing Fields (4 à 20 octets)

Définit les adresses de la source et du coordinateur de cette source, et les adresses de la destination et du coordinateur de cette destination.

- Data Payload (cargaison)

- Frame Check Sequence (2 octets) : Aucune correction n'est faite si des bits erronés sont détectés, il doit y avoir retransmission.

Trame « Beacon »

Octets: 2	1	4/10	0/5/6/10/14	2	variable	variable	variable	2
Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Superframe Specification	GTS fields (Figure 45)	Pending address fields (Figure 46)	Beacon Payload	FCS
MHR				MAC Payload				MFR

Figure 45- Structure de la trame Beacon

Les champs MHR et MFR sont identiques à la trame des données.

Les trames « Beacon » ne peuvent être transmises que par un dispositif possédant toutes les fonctionnalités (FFD). Les informations qu'elles contiennent servent à la gestion du réseau en décrivant les caractéristiques du PAN (adresse, allocation des champs GTS, début des communications…), ce qui va servir à la synchronisation du réseau. C'est-à-dire que le coordinateur du PAN envoie régulièrement des Beacons et que les dispositifs utilisent la réception régulière des Beacons pour tenir une base temps commune. Les Beacons peuvent aussi servir aux transmissions indirectes, c'est-à-dire que le Beacon peut contenir des adresses de dispositifs qui indiquent que des données sont en attente chez le coordinateur du PAN et qu'il faut aller les récupérer.

- **Superframe Specification (2 octets)**

Définit le mode de communication appelé Superframe. Plus de détails vont être donnés par la suite.

- **GTS Fields (k octets)**

Les champs GTS (*Guaranteed Time Slot*) permettent d'allouer un intervalle temporel à un ou plusieurs dispositifs pour leurs communications. Plus de détails vont être donnés par la suite.

- **Pending Address Fields (m octets)**

Ce champ sert à connaître les adresses des dispositifs qui ont des données en attente chez le coordinateur, il est

- Beacon Payload (n octets) : Cargaison de la trame

Trame d'acquittement (ACK)

Ces trames servent à acquitter les trames de données, et ne contiennent pas de cargaison.

Trame de commande

Frame Control	Sequence Number	Addressing fields	Auxiliary Security Header	Command Frame Identifier	Command Payload	FCS
MHR				MAC Payload		MFR

Figure 46- Structure de la trame de commande

Les parties MHR et MFR sont identiques à celles de la trame de données

Command Frame Identifier : Permet de spécifier le type de la commande :

- **Association Request :** Cette commande permet à un dispositif de faire une demande d'association à un coordinateur.
- **Association Response :** Cette commande permet au coordinateur de communiquer les résultats de l'association au dispositif qui a fait la demande.
- **Disassociation Notification :** Cette commande permet de savoir pourquoi il y a eu une dissociation.
- **Data Request :** Cette commande est utilisée dans un réseau « Beacon Enabled », elle est envoyée par un dispositif lorsqu'il a détecté par un Beacon reçu que des données sont en attente pour lui chez le coordinateur. Cette trame de commande est aussi utilisée lorsque le réseau est « Non Beacon Enabled » et que la couche supérieure du dispositif indique à la couche MAC de demander des données. Et enfin, elle est employée durant la procédure d'association d'un dispositif au PAN.
- **PAN ID Conflict Notification :** Cette commande est envoyée par un dispositif au coordinateur lorsqu'un conflit d'identifiant PAN est détecté.
- **Orphan Notification :** Cette commande est utilisée par un dispositif associé qui a perdu la synchronisation avec son coordinateur.
- **Beacon Request :** Cette commande est utilisée par un dispositif FFD afin de localiser les coordinateurs.
- **Coordinator Realignment :** Cette commande est envoyée par un coordinateur soit après avoir reçu une commande « Orphan notification » de la part d'un dispositif ou soit après un changement de plusieurs caractéristiques du PAN. Cette commande permet au dispositif orphelin d'être de nouveau associé au coordinateur et à tous les dispositifs de réactualiser les informations à propos du PAN (envoi par Broadcast).

- **GTS Request :** Cette commande est utilisée par un dispositif associé qui fait une demande pour allouer un nouveau GTS ou pour arrêter d'allouer un GTS existant.

Méthodes d'accès au canal

La norme contient deux méthodes d'accès :

- CSMA-CA qui est implémentée en deux versions, l'une non synchronisée (utilisé dans un réseau « Non Beacon Enabled ») et une synchronisée sur des *Backoffs* (utilisé dans un réseau « Beacon Enabled »).
- La deuxième méthode d'accès est une forme de polling par réservation de temps qui est obtenu grâce à un mode de transmission spécifique que l'on appelle Superframe. Cette dernière méthode ne peut être obtenue que dans un réseau synchronisé par un coordinateur (réseau « Beacon Enabled »).

La méthode CSMA-CA

Le principe du CSMA-CA (Carrier Sense Multiple Access - Collision Avoidance) est de détecter l'activité du canal avant de transmettre, afin d'éviter des collisions. Si le canal de transmission n'est pas libre, le dispositif attend qu'il se libère. Il existe deux versions de l'algorithme du CSMA-CA l'une est synchronisée (*slotted*) pour l'accès au canal, et l'autre ne l'est pas (*unslotted*)

La version « slotted » est utilisée lorsque le réseau est « Beacon Enabled », tandis que la version « unslotted » est utilisée lorsque le réseau est « Non Beacon Enabled ». Dans les deux cas, le CSMA-CA est uniquement utilisé pour les transmissions de trames de données et de commande.

Structure de la Superframe

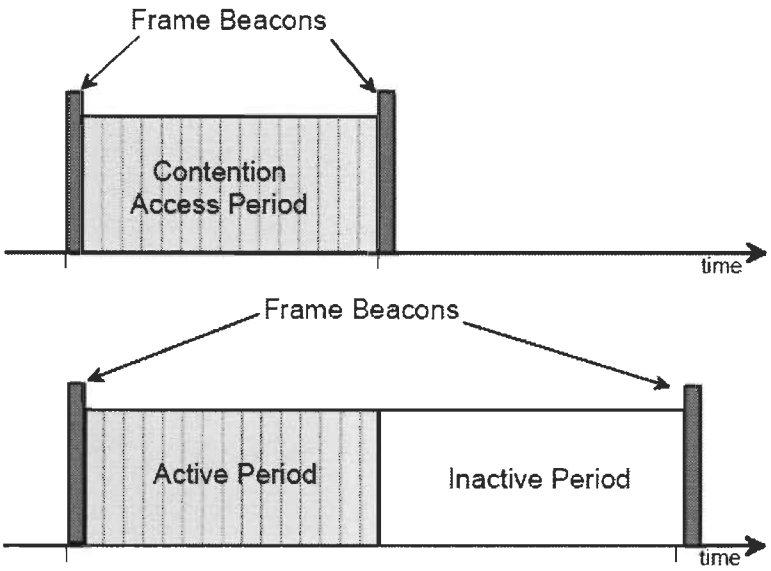

Figure 47- Structure de la Superframe

Le format de la Superframe est déterminé par le coordinateur. La Superframe comporte deux parties : Période active CAP (*Contention Access Period*) et période inactive où le coordinateur et tout le reste de son réseau passent en mode faible consommation. La Superframe est délimitée par deux trames Beacon, et est divisée en 16 périodes égales (période Beacon comprise).

Tous les dispositifs désirant communiquer durant la période CAP doivent entrer en compétition pour accéder au canal avec la méthode CSMA-CA.

Le coordinateur peut aussi allouer des plages temporelles aux dispositifs juste après la période CAP, à l'aide des champs GTS.

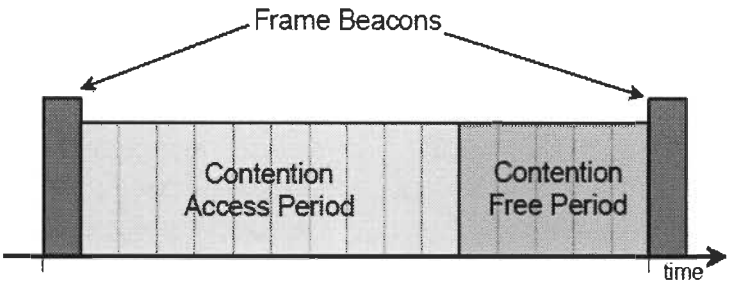

Figure 48- Périodes CAP (sans GTS) et CFP (avec GTS) de la Superframe

- **Transfert de données au coordinateur :**

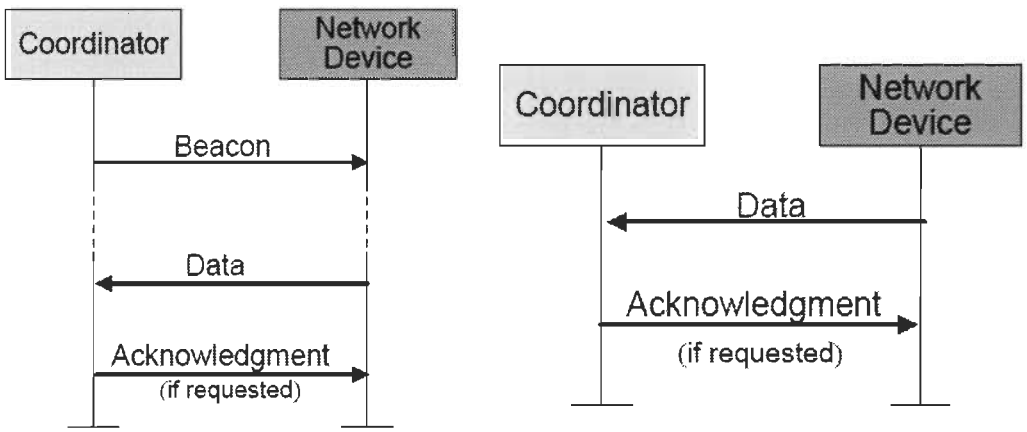

Figure 49- a) Transmission au coordinateur avec Beacon, b) Transmission au coordinateur sans Beacon

Dans le cas de la communication à base de Beacon, c'est la trame Beacon qui définira la base temporelle commune pour la communication. Dans le deuxième cas (pas de trame Beacon), c'est le dispositif du réseau qui prend l'initiative de la communication à chaque fois qu'il a des données pour le coordinateur.

- **Transfert de données à partir du coordinateur :**

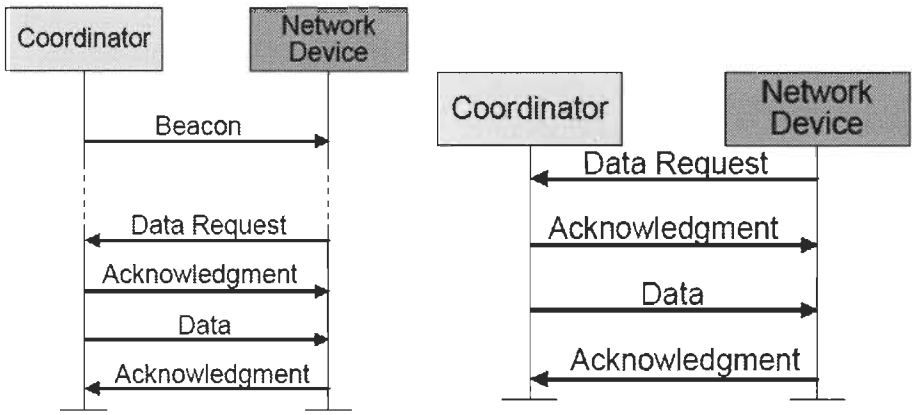

Figure 50- a) Transmission du coordinateur avec Beacon, b) Transmission du coordinateur sans Beacon

Dans le premier cas, le coordinateur envoie ces trames Beacon contenant le champs Frame Pending à 1, indiquant ainsi aux dispositifs concernés, qu'ils ont des données en attente chez le coordinateur et qu'il faut aller les chercher avec une requête de données (Data Request) envoyée au coordinateur. Pour le deuxième cas, c'est le dispositif qui envoie une requête de données pour savoir s'il a des données chez le coordinateur.

- **Pour les transmissions P2P :**

Les dispositifs peuvent envoyer leurs données avec la méthode CSMA-CA sans synchronisation. Garantir la synchronisation pour le P2P dépasse du cadre de la norme (plusieurs travaux ont été effectués à ce propos).

Valeurs par défaut de la Superframe

- L'unité de temps de base utilisée dans le standard est le **symbole**.
- La superframe est décrite par les valeurs *macBeaconOrder* et *macSuperframeOrder*.
- Le paramètre *macBeaconOrder* (BO) sert au calcul de l'intervalle entre deux beacons (BI) :

BI = *aBaseSuperframeDuration* * 2^*BO* symboles, avec $0 \leq BO \leq 14$

- Le paramètre *macSuperframeOrder* (SO) sert au calcul de la durée de la partie active de la superframe (SD) :

  SD = *aBaseSuperframeDuration* * 2^*SO* symboles, avec $0 \leq SO \leq BO \leq 14$

- La portion active de la superframe est divisée en *aNumSuperframeSlots* plages égales de durée 2^*SO* * *aBaseSlotDuration* chacun. L'ensemble de ces plages forme les trois parties de la partie active de la superframe : un Beacon, un CAP et un CFP

Le Beacon est envoyé durant la plage 0 en broadcast et sans CSMA-CA. La partie CAP commence juste après le Beacon. Le CFP arrive après le CAP s'il y a des GTS alloués. Les coordinateurs qui veulent utiliser des superframes choisissent un BO entre 0 et 14 (inclusivement), et un SO entre 0 et BO (inclusivement).

En choisissant le mode superframe ($0 \leq SO \leq BO \leq 14$), le réseau est Beacon Enabled. Par ailleurs, si BO = 15, la superframe n'est plus utilisée et le réseau est Non Beacon Enabled. Un réseau Non Beacon Enabled utilise le CSMA-CA unslotted pour accéder au canal, sachant que les GTS ne sont pas appliqués.

Chronologie des évènements

- Le coordinateur est actif durant SD et inactif durant BI-SD.
- Chaque nœud se réveille pour écouter la trame Beacon (ou bien en restant en veille mais en utilisant *macRxOnWhenIdle*). Le réveil se fait en passant du mode en veille au mode réception en allumant le module radio (MAC donne l'ordre à la PHY).
- Les nœuds concernés par la trame Beacon entrent en compétition sur le canal avec la méthode CSMA-CA, afin de transmettre leurs données.

- Les nœuds qui ont envoyé leurs données retournent en mode veille jusqu'à la prochaine BF (ils connaissent maintenant la durée du BI).

Base de temps

L'unité de temps de base utilisée dans le standard est le **symbole**.

La durée d'un symbole est directement liée au débit fixé par la couche physique.

EXEMPLE : Si la couche physique possède un débit fixe de 9.6 kb/s, et en sachant que le codage Manchester utilise 2 bits pour coder un symbole d'information, on obtient un débit de symbole (D) de moitié moins que le débit binaire : **D=4.8 kBauds.** La durée d'un symbole est donc

1 *symbol* = 1/D = 1/4800 = 0.208ms

Annexe B - Scripts et codes

Code TCL pour la simulation du protocole AODV sous la couche MAC IEEE802.15.4

Erreur ! Nom de fichier incorrect.

Attribution des positions de 2 nœuds

Erreur ! Nom de fichier incorrect.

Script AWK pour l'AODV

Erreur ! Nom de fichier incorrect.

Annexe C – Castalia Simulator

Le simulateur Castalia Simulator

Installation du logiciel Castalia Simulator

La version Castalia 3.2 est basé sur le simulateur de réseaux OMNeT++ [45], version 4.0 ou 4.1. Le système d'exploitation Linux est recommandé pour l'utilisation de Castalia.

Après avoir téléchargé Castalia du site http://castalia.npc.nicta.com.au/, faire entrer les commandes suivantes :

- Extraire le dossier Castalia :

$ tar –xvzf Castalia-3.0.tar.gz

- Compiler Castalia :

$ cd Castalia-3.0/

$ ./makemake

$ make

Caractéristiques du simulateur

Le simulateur Castalia dispose de plusieurs fonctionnalités qui lui permettent de modéliser, à travers la simulation, un comportement proche de la réalité d'un nœud de

capteur. Ainsi, Castalia propose une modélisation avancée du canal de propagation en se basant sur des données mesurées par l'équipe de développement du simulateur à travers :

- Une définition d'une cartographie de perte de signaux.
- Un canal variant dans le temps.
- La possibilité d'avoir des nœuds mobile sur un plan 3D.
- Plusieurs méthodes de calcul des interférences.

Castalia propose aussi une modélisation quasi-réaliste des communications radio à faibles énergies :

- Réception des signaux radio basée sur le SINR, la taille des paquets, la modulation,…
- Définition de plusieurs modes (RX, TX, et en veille) et les énergies et délais de transitions entre ces modes.

Comme le rôle des capteurs est de contrôler les processus physiques (température, humidité, vibrations,…), Castalia propose une modélisation avancée de ces processus par plusieurs méthodes (assignation de valeurs directement, une variation de valeurs, ou bien une équation basée sur la propagation du phénomène physique à contrôler).

Composition d'un nœud de capteur selon Castalia Simulator

Castalia définit le nœud de capteur comme décrit par la figure ci-dessous. Le nœud y est défini comme une série de blocs interconnectés définissant tous les processus qu'un nœud de capteur utilise. En effet, on retrouve le modèle OSI décrit auparavant (couches

radio, MAC, routage et application) ainsi que d'autres blocs liés à la gestion des ressources, le canal, la mobilité, la gestion du capteur, et le processus physique.

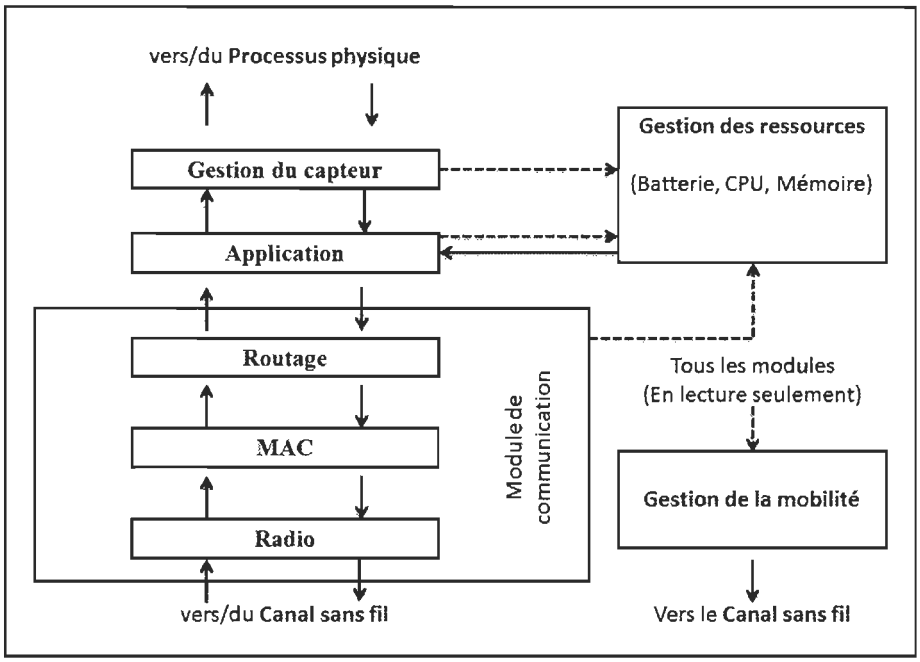

Figure 51- Composition d'un nœud de capteur selon Castalia Simulator

Les fichiers de configuration (.NED et .INI)

Le fichier .NED

Castalia offre une structure basée sur des modules interconnectés. Ceux-ci peuvent être configurés par un fichier spécial de forme .NED (*Network Definition*). Le fichier .NED d'un module donné établit sa structure en définissant les entrées/sorties et leurs paramètres de configuration.

Par exemple, si on veut définir le protocole TMAC pour le module MAC, il faut faire appel au fichier de configuration .NED qui va définir les différents paramètres (que l'utilisateur peut régler à sa guise) du protocole TMAC, comme montré par la prise d'écran suivante.

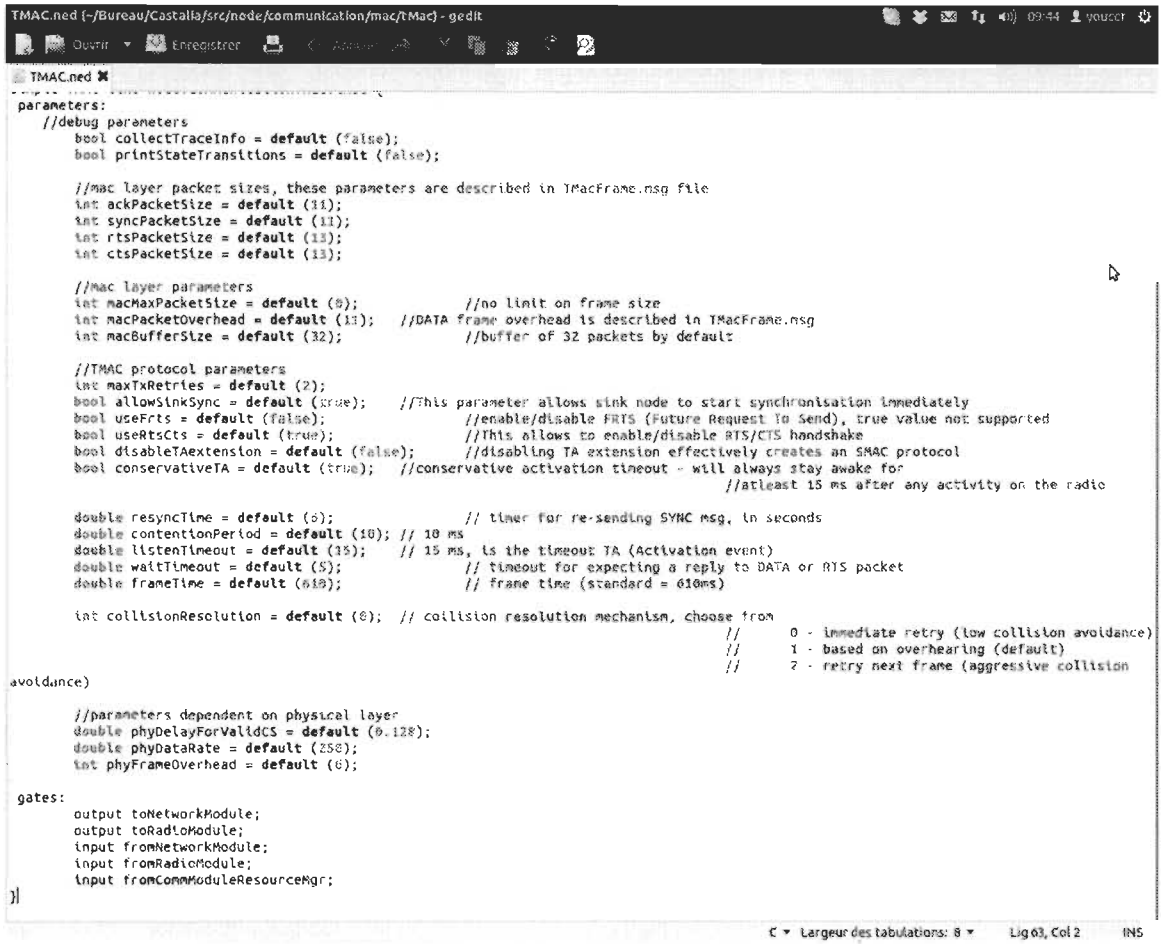

Figure 52- Exemple de fichier .NED du protocole TMAC

Ainsi, il est possible de définir par exemple la taille des différents paquets du protocole, les différents modes de retransmission, les intervalles de synchronisation et d'attente avant la retransmission,…

À la fin du fichier .NED, il faut définir les entrées et sorties (*gates*) de ce module, de cette manière Castalia saura de quel module s'agit-il, et avec quel module va-t-il communiquer.

Bien évidemment, le fonctionnement du protocole et l'utilisation de ces paramètres de configuration sont compilés par un code en C++ dans le dossier spécifique.

Le fichier .INI

Le fichier .INI réside dans le dossier directeur de la simulation et sert à définir les paramètres globaux de la simulation et d'assigner ou de changer les valeurs des paramètres de chaque module. La prise d'écran suivante montre un exemple de simulation d'un réseau de 8 capteurs dans un champ de 600m*600m utilisant :

- un module radio de type TI CC2420,
- le protocole TMAC,
- un protocole d'application spécifique,
- un protocole de routage donné dans Castalia,
- et un canal idéal

Figure 53- Exemple d'un fichier .INI

Le module Canal sans fil

Castalia offre une modélisation réaliste du canal sans fil basée sur des mesures empiriques du canal réalisées par le laboratoire NICTA [41] (environnement variable, nœuds mobiles, communication large bande).

La perte de puissance des signaux est modélisée par l'équation suivante :

$$PL(d) = PL(d_0) + 10.\eta.\log(\frac{d}{d_0}) + X_\sigma \quad (4)$$

Les pertes de puissance dépendent de la distance entre les deux nœuds (d), d'un exposant η, et d'une variable aléatoire gaussienne X_σ.

Castalia offre une cartographie de la perte de puissance de tout le réseau. Ainsi, il est possible de modéliser l'atténuation des signaux entre chaque couple de capteurs. Comme les capteurs peuvent se déplacer dans un plan 3D, cette cartographie des pertes de puissance des signaux s'adapte à ces mouvements.

Le simulateur offre à l'utilisateur de définir une variation temporelle du canal certaines expérimentations réalisées par l'équipe de développement [46].

Les signaux qui transitent dans le module du canal sans fil contiennent des informations sur le type de modulation, la largeur de bande, la fréquence de la porteuse, et l'intensité du signal en dBm. Ainsi, le module du canal sans fil envoie ce signal radio au module radio pour calculer le SINR afin de décider de recevoir le signal ou pas selon un seuil préétabli. En effet, les paramètres liés à la réception des signaux peuvent être ajustés :

- Le seuil de réception (dBm).
- Possibilité de rendre le canal « idéal » : transmission A→B identique à B→A, et qu'à l'intérieur du cercle de transmission, la réception et idéale, et à l'extérieur aucune réception n'est possible.

Le module Radio

Les principaux paramètres du module que l'utilisateur peut customiser sont :

- Plusieurs états : RX (réception), TX (transmission), en veille (plusieurs niveaux).
- Définition des délais et consommation d'énergie lors des transitions entre les états.
- Plusieurs niveaux d'énergies de transmission et de consommation.
- Définition du type de modulation, du débit, de la largeur de bande, seuil de bruit,…
- Calcule continu du paramètre RSSI (indicateur d'intensité du signal reçu).
- Évaluation du paramètre CCA (Channel Clear Assessment) qui indique si le canal est libre ou pas.
- Plusieurs Modèles d'interférences (additive, simple, et sans interférences).
- Les paramètres peuvent être changés de façon dynamique.

Le module MAC

Castalia Simulator offre quatre protocoles implémentés :

- Le *tunable* MAC.
- Le TMAC.
- La couche MAC du protocole IEEE 802.15.4.
- La couche MAC du protocole IEEE 802.15.6.

Le module Routage

Ce module permet de définir le protocole de routage à utiliser dans la simulation. Deux algorithmes de routage sont implémentés dans le Castalia.

Le module Application

Castalia Simulator offre à l'utilisateur la possibilité de définir sa propre Application, qui va servir à définir le comportement et le rôle de chaque nœud dans le réseau. Plusieurs applications sont offertes par le simulateur.

Le module de gestion du capteur

Ce module fait le lien entre le processus physique et le module d'application. Ainsi, il est possible de définir :

- La consommation du capteur en mJoules/échantillon.
- Le nom du processus de détection (par exemple, température).
- La source physique correspondante.
- Maximum du débit de détection des échantillons par seconde.
- Le bruit et la distorsion dans les échantillons détectés.
- La résolution des lectures.
- La saturation.
- La sensibilité aux variations.
- ...

Le module Processus physique

Le rôle de base d'un capteur est de détecter une source physique (température, son, vibration,...) et de l'émettre vers une station de base. Les autres simulateurs de réseaux de capteurs proposent des sources physiques basées sur un générateur de chiffres aléatoires, ce

qui est loin de modéliser une vraie variation d'une source réelle. Castalia offre jusqu'à 5 sources et une meilleure représentation des processus physiques en proposant une formule customisée qui dépend de la valeur de la source à l'instant *t*, distance du point par rapport à la source, la diffusion de la valeur de la source, et une valeur aléatoire gaussienne.

Le module Gestion de la mobilité

Ce module permet de :

- Spécifier l'équation du mouvement des nœuds dans l'espace.
- Déterminer la période de mise à jour de la position.
- Déterminer la vitesse du mouvement.

Le module Gestion des ressources

Les paramètres réglables sont :

- Énergie initiale (en Joules) : ex. Piles AA = 18720 Joules.
- Période de calcule de l'énergie consommée.
- L'horloge du CPU du capteur.

Annexe D – Le protocole de routage TBRPF (Topology broadcast based on reverse-path forwarding)

Introduction au protocole TBRPF

Le protocole TBRPF est un protocole de routage proactif destiné aux réseaux Ad-hoc mobile (MANET). Contrairement aux autres protocoles proactifs, le TBRPF dispose d'un algorithme pour le maintien d'une table de routage de façon à réduire la taille des entêtes des paquets. Ce protocole dispose de deux modules : un module de découverte du voisinage et un module de routage [47].

Le module de découverte du voisinage

Ce module consiste en un algorithme appelé TND (*TBRPF Neighbor Discovery*) qui permet à chaque nœud *i* d'établir un lien bidirectionnel avec un autre nœud (voisin) *j* (le lien est appelé (I, J)). De cette manière, chaque nœud peut détecter la perte de ce lien. L'algorithme TND communique avec le module de routage grâce à la table de routage établie par le TND ainsi que trois commandes qui sont : LINK_UP (I, J), LINK_Down (I, J), et LINK_CHANGE (I, J) qui annoncent, respectivement, un nouveau lien, la rupture

d'un lien et un changement dans un lien. La table de routage du TND est établie en utilisant les messages HELLO.

Les messages HELLO et la table de voisinage

Les messages HELLO sont utilisés pour trois tâches différentes : requête de voisin, réponse de voisin, ou perte de voisin. Grâce à ses messages HELLO, chaque nœud maintien une table de voisinage qui sauvegarde l'état de chaque voisin à partir duquel un message HELLO a été reçu. Les informations, qu'un nœud *i* maintiens d'un voisin *j*, sont :

- L'adresse de ce voisin.
- L'état du lien qui peut être : rompu, 1-chemin, ou 2-chemin, dans ce dernier, les deux nœuds ont reçu les messages HELLO dans les deux sens.
- Le temps avant de changer l'état du lien à l'état 'rompu' si aucun message HELLO n'est reçu. Ce temps est remis à jour à chaque réception d'un message HELLO de ce voisin.
- Le numéro de séquence du dernier HELLO reçu. Ce champ sert à calculer le nombre de HELLO perdus.
- Le nombre de messages (requête, réponse, perte) à envoyer au voisin *j*.
- Le nombre de HELLO reçus à partir du voisin *j*.
- Un champ optionnel concernant la qualité du lien (entier entre 1 et 255, la meilleure qualité étant 1). Ce champ peut être programmé par l'utilisateur pour correspondre à ses critères de qualité des liens (la qualité du signal, le nombre de HELLO reçus

durant un certain intervalle, la stabilité du lien,…), ceci peut servir comme seuil avant de déclarer que le lien a été perdu avec ce voisin.

Dans la table de routage du nœud *i*, si aucun HELLO n'est reçu à partir du voisin *j* durant un certain intervalle temporel, son entrée est supprimée.

Le module de routage

Ce module d'occupe de la découverte de la topologie et du calcul des routes. Chaque nœud garde – en plus des informations données par le TND – une Table de Topologie (TT) contenant l'information sur chaque nœud et lien connus dans le réseau.

La table de routage

La table de routage contient une liste d'informations sous la forme suivante : (rt_dest, rt_next, rt_dist, rt_if_id), qui sont, respectivement, la destination, le prochain hop, la longueur de la route (hops), et l'adresse de l'interface contenant le TND du prochain hop.

Le message de maintenance de la topologie

Ce message peut être sous une des trois formes suivantes :

- FULL : un nœud envoi ce message pour déclarer l'intégralité de tous ses liens.
- ADD : utilisé par un nœud pour déclarer les liens ajoutés à sa table de routage.
- DELETE : utilisé par un nœud pour déclarer les liens supprimés de sa table de routage.

Les opérations de routage

Les opérations de routage du protocole TBRPF sont résumées dans les points suivants [47] :

- Traitement périodique général : chaque nœud exécute une procédure de mise-à-jour complète Update_all (). Cette procédure vide la liste des interfaces voisines, envoie un message HELLO à chaque voisin, et remet à jour la liste des voisins. La table de routage est aussi remise à jour en prenant en compte les liens expirés.
- Mise-à-jour de l'arbre source et du graphe de topologie : cette procédure est appelée pour calculer les chemins les plus courts en fonction d'un paramètre de coût de lien.
- Mise-à-jour de la table de routage : la table de routage est remise-à-jour en fonction du résultat des deux dernières procédures.
- Sélection des voisins privilégiés (ensemble RN) : cette procédure complexe est décrite dans [47] et permet de choisir parmi les voisins d'un nœud l'ensemble des nœuds utiles pour avoir les chemins les plus courts.
- Générer des mises-à-jour partielles selon un intervalle temporel fixé.
- Plusieurs autres opérations concernant le traitement des mise-à-jour, l'annonce de la rupture des liens et de leur formation, sont données dans [47].