République Algérienne Démocratique et Populaire

Ministre de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira-Bejaïa

Faculté des Sciences Exactes

Département d'informatique

Mémoire de Master Professionnel

En Informatique

Option

Administration et Sécurité des Réseaux Informatiques

Thème

Adaptation du protocole CSMA/CA pour la localisation des mobiles en temps réel.

Présenté par :

- CHEKROUNE Ridha
- CHELIK Mourad

Devant le jury composé de :

Présidente	Melle SABRI Salima
Examinateur	Melle GHANEM Souhila
Examinateur	Mr. MOHAND Yazid
Encadreur	Mr. ACHROUFEN Achour

Promotion 2012 – 2013

Je dédie ce modeste travail à :

Mes très chers Parents

Mes frères et ma sœur

Mes grands parents

Mes tantes, mes oncles et leurs femmes

Mes cousins et cousines

Tous mes adorables amis

Toutes les personnes que j'aime en général, Amel en particulier

Tout mes enseignants

Ceux qui cherchent leurs noms ici.

Redha Chekroune

Je dédie ce modeste travail à :

Ma très chère mère

Mes grands parents

Mes tantes, mes oncles et leurs femmes

Mes cousins et cousines

Tous mes adorables amis

Tout mes enseignants

Ceux qui cherchent leurs noms ici.

Mourad Chelik

Quelques lignes ne pourront jamais exprimer la reconnaissance que nous éprouvons envers tous ceux qui, de près ou de loin, ont contribué, par leurs conseils, leurs encouragements ou leurs amitiés à l'aboutissement de ce travail.

Nous remercions DIEU tout puissant de nous avoir donné la force, la santé, le courage et la patience pour pouvoir accomplir ce travail.

Un grand merci à toutes nos familles surtout nos parents pour leurs incessants encouragements et leur suivi avec patience du déroulement de notre projet.

Nous tenons à remercier vivement notre promoteur Monsieur A. Achroufene pour nous avoir orienté, dirigé, soutenu, et être disponible tout au long de notre projet.

Nos sincères remerciements s'adressent aussi la communauté open source qui grâce à elle nous n'avons pas manqué d'outils nécessaires pour la réalisation de ce travail.

Enfin, nous tenons aussi à remercier tous les membres de jury pour avoir bien examiner notre travail et statuer sur notre candidature.

A tous merci de fond du cœur.

Table des matières

Liste des figures

Liste des tableaux

Liste des Acronymes

ACK	ACKnowledgment
AOA	Angle Of Arrival
AP	Access Point
API	Application Programming Interface
APS	Ad hoc Positioning System
APS_{AOA}	Ad hoc Positioning System AOA
BS	Base Station
BSS	Basic Service Set
CA	Collision avoidance
CAN	convertisseur analogique numérique
CD	Collision Detection
CS	Carrier Sense
CSMA	Carrier Sense Multiple Access
CTS	Clear To Send
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	Distributed Inter-Frame Spacing
DQDB	Distributed Queue Dual Bus
DSSS	Direct Sequence Spread Spectrum
DV-HOP	Distance Vector-HOP
EIFS	Extended IFS
ESS	Extended Service System
FHSS	Frequency Hopping Spread Spectrum
GPS	Global Positioning System
HTRefine	Hot Top Refine
IBSS	Independent BSS
ID	Identification
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFS	Inter-Frame Spacing
IGMP	Internet Group Multicast Protocol
IP	Internet Protocol
IR	Infra-Red
ISO	International Organization for Standardization
K-NN	K-Nearest Neighbor
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MANTIS	MultimodAl system for NeTworks of In-situ wireless Sensors
MAX	Maximum
MIN	Minimum
NAV	Network Allocation Vector
NM	Nœud Mobile
NR	Nœud de Référence
NTP	Network Time Protocol

OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open System Interconnection
PCF	Point Coordination Function
PDA	Personal Digital Assistant
PHY	PHYsical layer
PIFS	Point Coordination Inter-Frame Spacing
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
QoS	Quality of Service
RAM	Random Access Memory
RCSFs	Réseaux de capteurs sans fil
RFCs	Request For Comment
ROM	Read-Only Memory
RSSI	Received Signal Strength Indicator
RTS	Request To Send
SIFS	Short IFS
SYNC	Synchronization
TCP	Transport Control Protocol
TDOA	Time Difference Of Arrival
TDMA	Time Division Multiple Access
TOA	Time Of Arrival
TOF	Time of Flight
TinyOS	Tiny Operating System
UDP	User Datagram Protocol
Wi-Fi	Wireless-Fidelity
WLAN	Wireless Local Area Network
WSN	Wireless Sensors Network

Introduction générale

Depuis quelques décennies, le besoin d'observer, de surveiller à distance et de récupérer les données d'un environnement complexe et distribué s'accroit rapidement surtout avec les récentes avancées des domaines de réseaux et des applications sans fil.

Parmi les technologies, en vogue, répondant à ces domaines, on cite entre autres les réseaux de capteurs sans fil. Un réseau de capteurs sans fil peut être défini comme un ensemble de composants miniatures (capteurs) capable de capter des grandeurs physiques à partir d'un environnement donné et de transformer ces données en grandeurs numériques dans le but d'établir une communication avec les autres capteurs du réseau et d'acheminer la somme des données collectées vers une station de base.

En l'absence d'information sur la position des éléments d'un réseau de capteurs sans fil au sein de l'environnement où ils sont déployés, les données récoltées peuvent s'avérer d'une utilité limitée. Une étape préalable à tout traitement consiste donc à estimer la position de ces nœuds, à partir de mesures de portée inter-capteurs tel que les RSSI (Received Signal Strength Indication), et de la position supposée connue d'une fraction de capteurs appelés *ancres*. Ce problème de localisation à fait et fait encore l'objet de nombreux travaux de recherche dans divers domaine tels que la santé (surveillance de patient à domicile par exemple), le militaire (la poursuite de cible), l'environnement, etc.

Ce travail rentre dans le cadre de la localisation indoor où l'on cherche à estimer la position d'un mobile se déplaçant dans un environnement connu dans lequel est déployé un réseau de capteurs sans fil. La position du mobile doit être estimée à chaque instant Δt, ce qui signifie que l'application à réaliser doit répondre aux contraintes du temps réel. C'est-à-dire, l'exactitude des résultats n'est pas la seule priorité, ces derniers doivent également être disponibles dans des délais bien déterminés.

Pour la manipulation et la programmation des capteurs dans notre travail nous utilisons le système d'exploitation Contiki dédié aux réseaux de capteurs. Ce système propose une implémentation du protocole d'accès au support CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) qui n'est pas adapté pour les applications en temps réel en général et de localisation en particulier. C'est ce que nous allons démonter, en proposant de nouvelles

implémentations de ce protocole permettant de pallier ce problème, par un ensemble de tests, que ce soit en simulation dans un premier temps et en pratique avec des capteurs matériels dans un deuxième temps.

Ce mémoire propose une étude sur le protocole d'accès au support utilisé par les réseaux de capteurs sans fil à savoir CSMA/CA. Cette étude a pour objectif d'adapter ce protocole aux applications temps réel telle que la localisation de mobiles dans notre cas. Par ailleurs, la technique de localisation adoptée repose sur le concept de la multi-latération, qui consiste à déterminer la position d'un mobile à l'aide au moins de trois ancres où les distances entre le mobile et les ancres sont estimées à partir de la puissance du signal reçu RSSI.

Pour cela notre mémoire est organisé comme suit :

- Le **premier chapitre** donne un aperçu général sur les réseaux de capteurs sans fil, présente ses caractéristiques, son architecture, ses systèmes d'exploitation et ses différents domaines d'applications.

- Le **deuxième chapitre** décrit le standard IEEE 802.11, en mettant l'accent sur la couche MAC (Medium Access Control) et donne une description détaillée du protocole d'accès au support CSMA/CA.

- Le **troisième chapitre** passe en revue les différentes techniques de localisation et de positionnement qui peuvent être utilisées dans les réseaux de capteurs, et étudie quelques algorithmes de localisation dans les réseaux de capteurs statiques.

- Le **dernier chapitre** présente nos propositions qui consistent en l'amélioration de CSMA/CA pour les applications en temps réel. En outre, il décrit l'application de la localisation dans les réseaux de capteurs sans fil. Egalement, il résume les différentes analyses et interprétations des résultats obtenus dans les divers tests effectués.

Enfin, comme pour tout travail, on termine le mémoire par une conclusion générale afin de synthétiser les objectifs atteints, les acquis sur les plans théorique et surtout pratique, ce qui reste à faire et quelques perspectives en vue.

Chapitre 1

Introduction aux réseaux de capteurs

I. Introduction

Il est inenvisageable que l'être humain effectue par lui-même certaines tâches de surveillance et dans la plus part des cas c'est même absurde, soit parce que la plus part des phénomènes surveillés sont imperceptibles pour nos cinq sens, ou que la surveillance se fait dans un environnement hostile, ou encore, la quantité de travail est colossale. Toutes ces raisons soulignent le besoin d'utiliser une technologie pour réaliser la tâche de contrôle automatiquement, et dans ce cas c'est d'un *capteur* que nous avons besoin. Les capteurs sont des instruments qui détectent et récoltent des grandeurs physiques puis procèdent à leur numérisation, mais les capteurs récents ne se limitent pas à cette fonction basique et poussent le concept un peu plus loin en introduisant d'une unité de calcul appelée *microcontrôleur* [1] qui effectue les premiers traitements au lieu d'envoyer les informations comme données brutes.

Ce premier chapitre va s'attarder sur les Réseaux de Capteurs Sans Fil (RCSFs) ou WSN (*Wireless Sensor Network*) en partant de leur cellule élémentaire, le capteur sans fil.

II. Définition d'un capteur sans fil

Un capteur sans fil est l'élément de base dans un réseau de capteurs, c'est sur ce dernier que le système d'exploitation et les applications sont mis en œuvre. Nous présenterons dans cette section les principaux composants d'un capteur sans fil puis nous donnerons un exemple d'une plateforme matérielle commercialisée.

II.1 Architecture matérielle d'un capteur sans fil

Un capteur sans fil est le plus souvent composé d'un ou plusieurs instruments de mesure (capteurs ou sondes), une unité de traitement appelée communément microcontrôleur (µcontrôleur) et une source d'énergie. La figure 1.1 montre un schéma type d'un capteur sans fil.

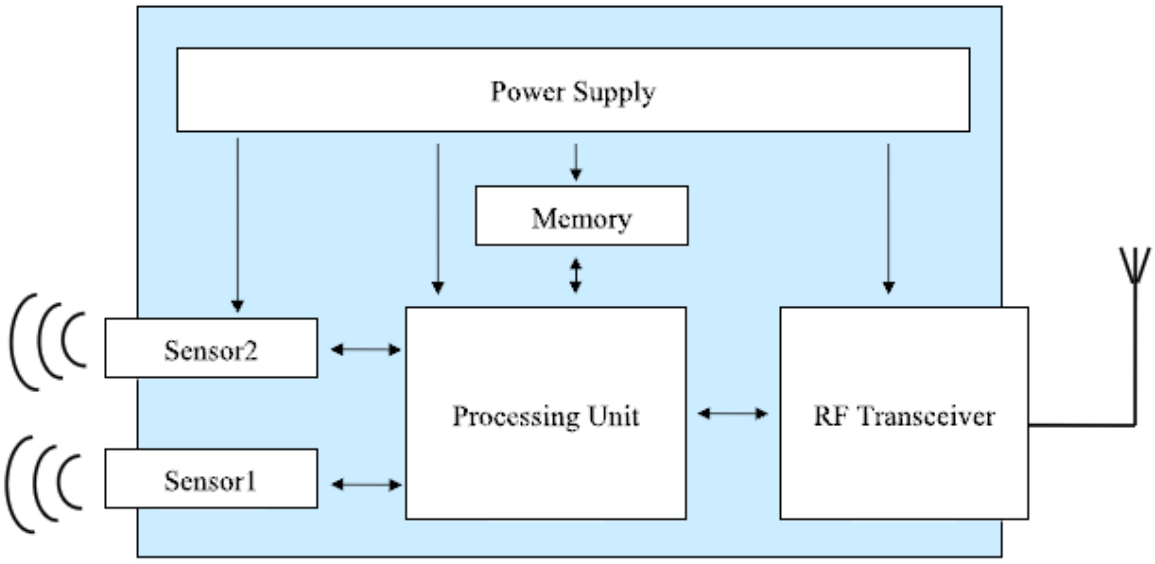


Fig. 1.1 : les principaux composants d'un capteur sans fil [64].

Sensor 1 et Sensor 2 dans la figure sont des instruments de mesure ou sondes qui mesurent des grandeurs physiques puis procèdent à leur conversion en un signal qui va être traité par l'unité de traitement (Processing Unit), plusieurs phénomènes physiques comme la température, le bruit sonore, les vibrations, l'humidité, la pression…, peuvent être mesurés [64].

Dans les applications informatiques industrielles, le nombre de circuits doit souvent être réduit pour limiter le coût et l'encombrement des systèmes [2]. Pour répondre à ce besoin, un nouveau type de microprocesseur est apparu, intégrant de la mémoire et de nombreuses interfaces d'entrées/sorties : *le microcontrôleur* [2] (Processing Unit dans la figure 1.1). Le µcontrôleur est un système à microprocesseur autonome contenu dans un seul boitier, il ressemble dans son fonctionnement au microprocesseur [2]. La largeur de bus de données peut être de 8,16 ou récemment de 32 bits [2], le µcontrôleurs intègre une mémoire interne : une ROM (*Read Only Memory*) pour stocker le programme à exécuter et une partie RAM (*Random Access Memory*) utilisée pour les données. Cette mémoire modeste peut être étendue de manière externe en cas de besoin [2].

Plusieurs choix de technologies de communication sans fil s'offrent aux concepteurs des capteurs sans fil. L'organisme de normalisation IEEE (*Institute of Electrical and Electronics Engineers*) consacre plusieurs groupes de travail pour la normalisation des réseaux sans fil, les principales normes sont IEEE 802.15 pour les petits réseaux personnels d'une dizaine de mètres de portée, et le IEEE 802.11 ou Wi-Fi (Wireless-Fidelity) pour les réseaux locaux sans fil.

Dans le groupe IEEE 802.15 trois sous-groupes normalisent des gammes de produits en parallèle :

Protocole	**Fréquence**	**Modulation**	**Débit maximal**
Bluetooth IEEE 802.15.1	2.4GHz	FHSS	1Mb/s
ZigBee IEEE 802.15.4	2.4GHz	O-QPSK	250kb/s
UWB IEEE 802.15.3	3.1-10.6GHz	PPM / BPM	100-500Mb/s

Tableau 1.1 : les différentes normes 802.15 [1]

Du côté de la norme IEEE 802.11 il existe plusieurs déclinaisons offrant des débits et des portées différents :

Protocole	**Fréquence**	**Modulation**	**Débit maximal**
IEEE 802.11a	5GHz	OFDM	54Mb/s
IEEE 802.11b	2.4GHz	DSSS/HR-DSSS	11Mb/s
IEEE 802.11g	2.4GHz	DSSS / OFDM	54Mb/s

Tableau 1.2 : les différentes normes 802.11 [1].

Le module de communication (RF Transceiver dans la figure 1.1) implémente soit des protocoles de la famille IEEE 802.11 ou IEEE 802.15 [1].

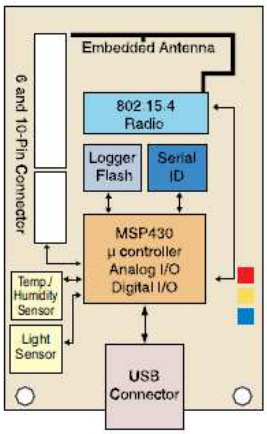
Le bloc d'alimentation (Power Supply) est un élément délicat dans un capteur, son choix doit corréler avec le type de µcontrôleur et le module de communication retenu, tout en gardant à l'esprit l'application pour laquelle notre capteur est destiné. La première idée consiste à s'orienter vers une source d'énergie naturelle et inépuisable comme celle fournie par les panneaux solaires. Cependant, dans la majorité des cas, cela n'est pas possible, parce que ce type d'équipement est fragile et encombrent [1]. Par conséquent, les constructeurs ont souvent recours à une ou plusieurs batteries de type AA comme source d'énergie [1]. Par ailleurs, des techniques économes comme l'inclusion d'un mode de veille par exemple peuvent être implémentées, à condition, que le passage d'un mode à l'autre ne pénalise pas la consommation d'énergie [1].

II.2 La plateforme hardware TelosB

La plate-forme open source *TelosB* a été développée et édité à la communauté de recherche par l'université de Californie, Berkeley et commercialisé à travers Crossbow Technology Inc. Cette plateforme favorise une basse consommation d'énergie. Le modèle TPR2420 montré dans la figure 1.2 est actionné par deux batteries de type AA, si le capteur est connecté à un ordinateur par le port USB l'énergie vient de cette ordinateur, s'il est toujours attaché à cette ordinateur aucune batterie n'est nécessaire. Ce capteur offre aux utilisateurs la

possibilité de connecter des dispositifs additionnels via deux interfaces de 6 et 10 pins. TPR2420 offre beaucoup de dispositifs, incluant [3]:

- Emetteur récepteur de radio fréquence IEEE 802.15.4 conforme ;
- La bande de fréquence utilisée est 2,4 à 2,4835 GHz ;
- Débit de 250 Kbps ;
- Antenne intégrée ;
- Un microcontrôleur TI MSP 430 8MHz, 16-bit RISC ;
- 10 Ko de RAM disponible;
- Une mémoire flash pour les programmes de 48 Ko ;
- Programmation et collection de données via USB ;
- Intégration de capteurs de lumière, température et humidité ;
- Une consommation de 1,8 mA en mode actif et 5,1 µA en mode veille.



III. Réseaux de capteurs sans fil

Les réseaux de capteurs ont donné naissance à de nombreuses perspectives de développements. En effet, là où un capteur seul a un rayon d'action limité à sa propre portée, un réseau de capteurs permet d'acheminer les données à un utilisateur plus éloigné et de couvrir, au même moment, une surface plus importante [1].

III.1 Réseaux Ad hoc

Les réseaux locaux sans fil peuvent accueillir plusieurs topologies réseau. Dans la description de ces topologies, la pierre angulaire de l'architecture de réseaux locaux sans fil IEEE 802.11 est l'ensemble de services de base (BSS) [65]. La norme définit l'ensemble de services de base comme un groupe de stations qui communiquent entre elles [65]. Les réseaux sans fil peuvent fonctionner sans infrastructure ; il s'agit dans ce cas d'une topologie ad hoc, la

norme IEEE 802.11 désigne un réseau ad hoc sous l'appellation IBSS (Independent BSS, ensemble de services de base indépendant) [65].

III.2 Réseaux de capteurs

Un RCSF est différent d'un réseau sans fil Ad Hoc, la différence se situe au niveau du nombre et des caractéristiques des nœuds qui forment le réseau [1]. Le réseau de capteurs sans fil est généralement caractérisé par un déploiement dense avec des centaines voire des milliers de nœuds. Les nœuds captent et communiquent leurs données (selon un protocole de communication) afin d'atteindre le nœud central de traitement (connu sous le nom de *Sink*). L'architecture du réseau est présentée dans la figure 1.3. Le WSN est caractérisé par sa capacité d'auto organisation, de coopération, de rapidité de déploiement, et de faible coût [4].

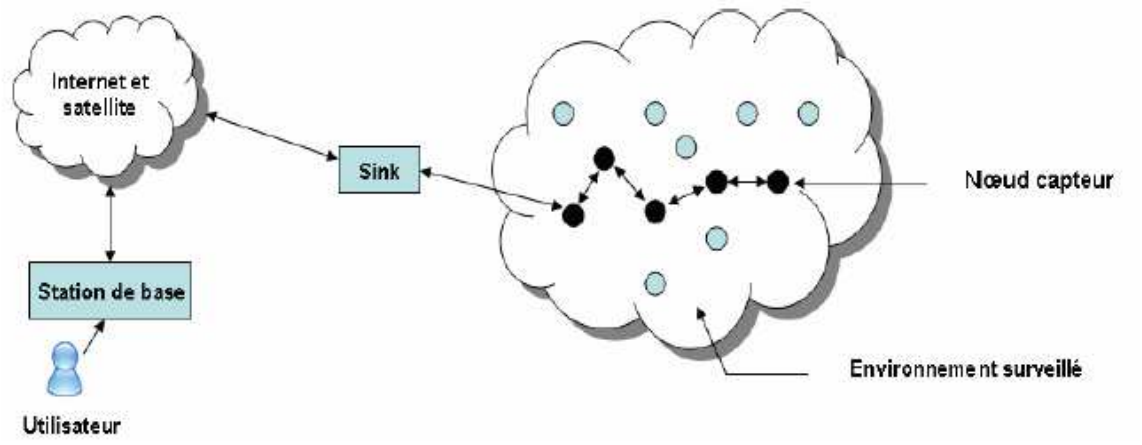


Fig. 1.3 : Architecture d'un réseau de capteurs [4].

IV. Topologies des réseaux de capteurs sans fil

Comme annoncé dans la section précédente, les réseaux sans fil peuvent accueillir plusieurs topologies. Dans cette section nous discutons de celles qui sont applicables aux RCSFs. Il existe plusieurs topologies pour les réseaux de capteurs [66] :

IV.1 Topologie en étoile

La topologie en étoile est un système uni-saut [66]. La station de base est capable de communiquer (émission et/ou réception) avec les autres nœuds, ces derniers ne peuvent émettre ou recevoir que vers ou depuis l'unique station de base [67]. Cette topologie est simple et elle demande une faible consommation d'énergie, mais la station de base est vulnérable et la distance entre les nœuds et la station est limitée [66].

IV.2 Topologie en grille

La topologie en grille est un système multi-saut [66]. Si un nœud destinataire est dans la portée radio d'un nœud émetteur, alors la destination est directement accessible, sinon

l'émetteur doit faire appel à des techniques de routage en passant par les nœuds voisins (intermédiaires) pour atteindre la cible [67]. Cette topologie a plus de possibilités de passer à l'échelle du réseau, avec redondance et tolérance aux fautes, mais elle demande une consommation d'énergie plus importante [66].

IV.3 Topologie hybride

La topologie hybride est un mélange des deux topologies ci-dessus. Les stations de base forment une topologie en grille et les nœuds autour d'elles sont en topologie étoile. Elle assure la minimisation d'énergie dans les réseaux de capteurs [66].

Le tableau 1.3 représente un récapitulatif des avantages est des inconvénients des trois topologies de RCSF :

Topologie	Routage	Portée	Redondance des chemins	Résistance aux pannes	Consommation d'énergie
Etoile	Non	Limitée	Non	Non	Moins importante
Grille	Oui	Grande	Oui	Oui	Plus importante
Hybride	Oui	Grande	Non	Oui	Moins importante

Tableau 1.3 : les principales différences entre les trois topologies.

V. Les systèmes d'exploitations pour les réseaux de capteurs

Les composants à l'intérieur d'un capteur sont de plus en plus diversifies, ce qui rend leur gestion complexe et difficile. Les plateformes hardwares existantes pour les capteurs sans fil utilisent des modules de communications sans fil différents et des dispositifs de captage distincts, la seule caractéristique qui est unifiée est l'architecture du processeur avec les données en RAM et le code de programme en ROM [5], d'où le besoin d'une couche logiciel qui interfacera le matériel et les programmes, cette couche s'appelle traditionnellement *système d'exploitation* et son premier rôle est d'abstraire les applications et donc les programmeurs des différents détails bas niveau liés au matériel [6].

V.1 définition d'un système d'exploitation

Un système d'exploitation peut être vu comme une machine virtuelle au-dessus du matériel plus facile d'emploi et plus conviviale. Vu d'en dessous le système d'exploitation est un gestionnaire de ressources, il prend à sa charge la gestion de ressources de plus en plus complexes et diversifies [7].

V.2 Topologie des systèmes d'exploitation embarqués

Un système d'exploitation embarqué est un système utilisé dans un équipement industriel ou un bien de consommation. La différence essentielle avec un système d'exploitation classique

tient à la complète intégration du système embarqué dans cet équipement : il n'a pas de raison d'être en dehors de l'équipement pour lequel il a été conçu. Il existe deux grandes familles de système embarqués ceux dit temps réels et les autres [7]. Nous allons expliquer dans ce qui suit les différents modes de partage de temps dans ces deux familles.

V.2.1 Temps partagé et temps réel

La gestion du temps est un des problèmes majeurs des systèmes d'exploitation, en effet, ces derniers sont multitâches, or il n'y a qu'un seul processeur, le système doit donc partager le temps du processeur entre ces processus et c'est la fonction de *l'ordonnanceur* (ou *scheduler*). Un système d'exploitation classique comme Unix, Linux ou Windows utilise la notion de *temps partagé ou quantum de temps*, par opposition *au temps réel*. Dans ce type de système, le but de l'ordonnanceur est de donner à l'utilisateur une impression de confort d'utilisation tout en assurant que toutes les tâches demandées sont finalement exécutées, mais la gestion des *interruptions* reçues par une tâche n'est pas optimisée dans le sens ou le temps de latence –soit le temps entre la réception de l'interruption et son traitement– n'est pas garanti par le système. Par comparaison, le temps de latence dans le cas d'un système temps réel est souvent inférieur à 100 microsecondes. Il existe un grand nombre de définitions d'un système dit *temps réel*. Une définition simple d'un tel système pourra être la suivante [7] :

- « Un système est dit temps réel lorsqu'il est soumis à des contraintes de temps et qu'il y répond dans un intervalle acceptable » ;

Ou bien :

- « Un système temps réel est une association logiciel/matériel où le logiciel permet, entre autres, une gestion adéquate des ressources matérielles en vue de remplir certaines tâches ou fonctions dans des limites temporelles bien précises ».

Ainsi, on pourra diviser les systèmes temps réels en deux catégories [7]:

- Les systèmes dits à contraintes *souples* ou *molles* (*soft real time*). Ces systèmes acceptent des variations dans le traitement des données de l'ordre de la demi-seconde (ou 500 ms) ou la seconde.
- Les systèmes dits à contraintes *dures* (*hard real time*) pour lesquels une gestion stricte du temps est nécessaire pour conserver l'intégrité du service rendu.

Un système temps réel n'est pas forcément plus rapide qu'un système à temps partagé. Il devra en revanche satisfaire à des contraintes temporelles prévues à l'avance.

V.2.2 Préemption et commutation de contexte

Le noyau (*kernel*) est le composant principal d'un système d'exploitation multitâche moderne. Dans un tel système, chaque tâche (ou processus) est découpé en *threads* (*processus léger* ou *tâche légère*) [7]; ce sont des éléments de programmes, chacun étant capable d'exécuter une portion de code dans un même espace d'adressage. Chaque thread est caractérisé par un *contexte* local contenant la *priorité* du thread, ses variables locales ou l'état de ses registres. Le passage d'un thread à un autre est appelé changement de contexte (*context switch*). Ce changement de contexte sera plus rapide sur un thread que sur un processus car les threads d'un processus évoluent dans le même espace d'adressage, ce qui permet le partage des données entre les threads d'un même processus. Dans certains cas, un processus ne sera composé que d'un seul thread et le changement de contexte s'effectuera sur le processus lui-même [7].

V.3 Les caractéristiques des systèmes d'exploitation pour les réseaux de capteurs

Les importantes caractéristiques à considérer dans un système d'exploitation pour capteurs sans fil sont les suivantes [68] :

- Temps réel : une grande partie des applications pour les réseaux de capteurs sont des applications dites *temps réel* (localisation, détection d'incendie, etc.), d'où la nécessité d'utiliser un système d'exploitation temps réel à contraintes dures ou souples car ces derniers sont plus prévisibles par rapport aux systèmes classiques ce qui nous assura qu'une tâche donnée terminera son exécution dans les délais fixés.
- L'empreinte mémoire : l'empreinte mémoire d'un système est la quantité de la mémoire occupée par ce dernier. A cause des ressources limitées sur un capteur les systèmes destinés aux réseaux de capteurs doivent avoir l'empreinte mémoire la plus petite possible.
- Le chargement et le déchargement dynamique : les capteurs sont souvent déployés à de grandes échelles dans des environnements très durs d'accès. Il est donc primordial d'implémenter au niveau du système d'exploitation des mécanismes qui permettront le chargement et le déchargement individuel des applications et des services sans avoir recours à envoyer une image binaire complète du système et de l'application qui est souvent beaucoup plus grande en taille que l'application et donc coûteuse en énergie pour la transmission.

V.4 Tour d'horizon des systèmes pour capteurs existants

Les systèmes pour capteurs peuvent être de trois modèles de programmations, le multithread traditionnel (*thread-driven*), un modèle piloté par les évènements (*even-driven*) et un hybride des deux premiers [68]. Dans la section suivante nous donnerons un exemple de système d'exploitation pour chacune de ces trois catégories.

V.4.1 MANTIS : un système multithreads

MANTIS (MultimodAl system for NeTworks of In-situ wireless Sensors) est un système d'exploitation multithread préemptif dédié aux RCSFs [8]. Il est développé à l'Université du Colorado avec le langage de programmation C. L'architecture complète (voir la Figure 1.4) du système d'exploitation MANTIS intègre différents éléments regroupés de la manière suivante :

- L'interpréteur de commandes (*shell*) ;
- la pile de protocole réseau ;

- L'API (*Application Programming Interface*) du système ;
- Le noyau et l'ordonnanceur ;
- Le module de communication (COMM) ;
- Le gestionnaire de périphériques (DEV).

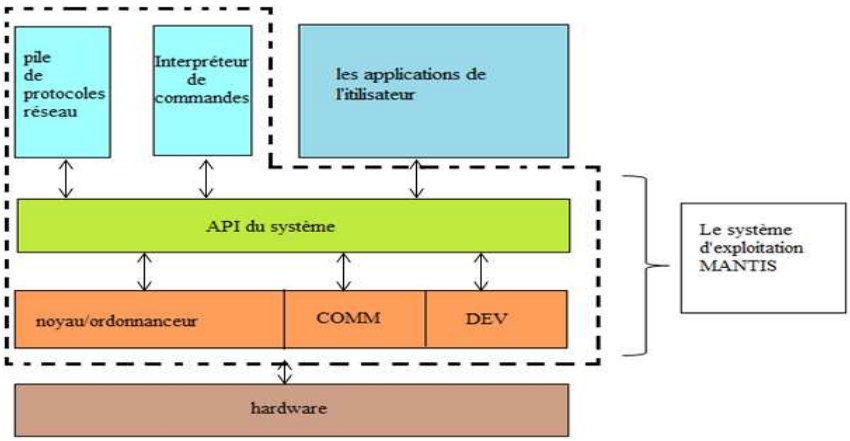


Fig. 1.4 : L'architecture de système MANTIS [8].

L'interpréteur de commandes (shell) offre à l'utilisateur un accès à distance au capteur pour changer sa configuration par exemple.

La communication dans MANTIS se fait par le biais, une pile de protocoles réseaux qui peut être représentée ainsi :

- la couche Application ;
- la couche Transport ;
- la couche Réseau ;
- la couche communication.

Les trois premières couches sont conformes aux recommandations du modèle OSI (reprennent les mêmes fonctionnalités) et restent au niveau logique de l'utilisateur, pour retrouver la couche communication par contre, il faut descendre jusqu'au niveau logique de noyau, le rôle principal de la couche communication est le contrôle d'accès au support des protocoles comme CSMA (Carrier Sense Multiple Access) et TDMA (Time Division Multiple Access) peuvent être utilisées.

Le MANTIS fournit un environnement commode pour créer des applications pour les RCSFs, grâce à L'API du système. Grâce à elle nous pouvons facilement créer des applications qui s'exécuteront comme des thread, une application peut créer un autre thread à l'aide de la primitive *thread_new()*, MANTIS initialise correctement d'autres thread du système, tels que les threads de la pile de réseau par exemple.

L'aspect multithreads et préemptif dans MANTIS est assuré par des ordonnanceurs (ou *schedulers*) de type UNIX [8], MANTIS attribue une priorité à chaque processus, les processus les plus prioritaires passent avant les moins prioritaires, dans une file de processus de même priorité l'algorithme *tourniquet avec quantum de temps* est appliqué. Pour la gestion des ressources critiques MANTIS propose les sémaphores (des composants utilisés pour la gestion des ressources partagées entre différentes tâches).

Les entités COMM et DEV offrent une interface aux threads pour communiquer avec les périphériques. Les entités COMM est en charge des communications synchrones et DEV des communications asynchrones.

V.4.2 TinyOS : un système piloté par les évènements

Les systèmes multithreads nécessitent l'utilisation de beaucoup de mémoire or la ressource mémoire est très limitée dans un capteur [5], d'où l'apparition des autres systèmes dit pilotés par les évènements comme TinyOS.

TinyOS a été développé à l'Université de Californie Berkeley avec le langage de programmation NesC, un dialecte de C. TinyOS est une plateforme de développement pour les réseaux de capteurs, autrement dit, TinyOS n'est pas un système d'exploitation au sens traditionnel, mais un ensemble de *composants* qui pourront être combinés pour former un système d'exploitation spécifique à chaque application [9].

Un programme TinyOS est un graphe de composants, chaque composant offre des interfaces pour l'interaction avec les autres composants. Un composant de niveau supérieur envoie des *commandes* à des composants de niveau inférieur et ces derniers répondent par des *évènements*. Dans un composant on trouve donc un gestionnaire de commandes, un gestionnaire d'évènements (*event-handlers*), les processus (ou les tâches) associes ainsi qu'un contexte d'exécution. Chaque composant possède un espace mémoire fixe.

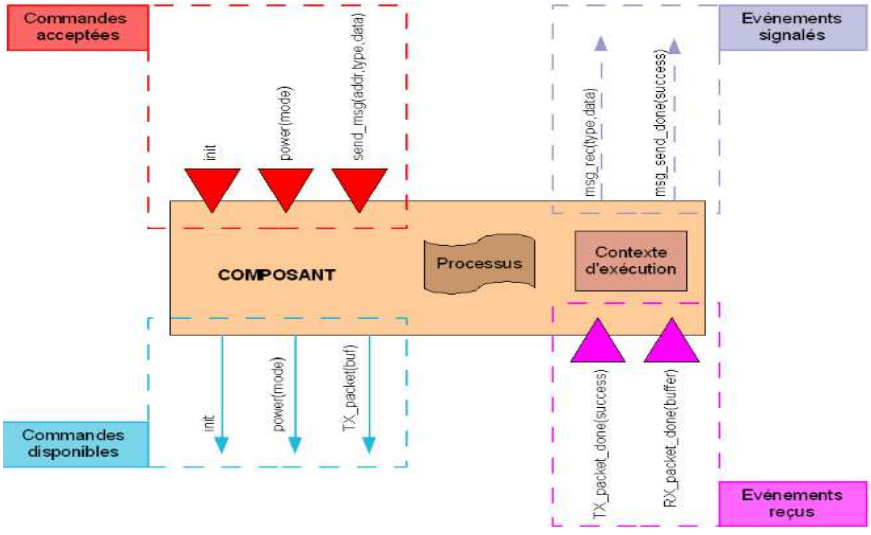


Fig. 1.5 : exemple d'un composant TinyOS avec ses interfaces [1].

Une comm rieur d'assurer un certain service, al e. Les évènements

peuvent aussi être signalés de façon asynchrone par exemple lors d'une interruption matériel ou lorsqu'un message arrive.

Les ressources matérielles sont abstraites dans TinyOS par les composants, qui offrent des interfaces (commandes) aux composants supérieurs, par exemple : un composant supérieur peut appeler la commande *getData()* sur le composant d'abstraction qui représente le dispositif de captage, ce qu'il lui fera signaler plus tard un évènement *dataReady()* [9]. Tous les traitements et l'algorithmique de l'application, sont contenus dans des composants de plus haut niveau logiciel.

Concernant la politique d'ordonnancement, TinyOS est un système piloté par les évènements (ou *event-driven*), lorsque le gestionnaire d'évènements d'un composant reçoit un évènement, les processus concernés dans ce composant vont s'exécuter successivement jusqu'à l'accomplissement (*run-to-completion*), contrairement à un système multitâche préemptif ou des processus peuvent interrompre d'autre processus. Ici seules les interruptions matérielles peuvent interrompre une tâche en cours d'exécution.

Ces caractéristiques prouvent que TinyOS est un système innovant est entièrement pensé pour les ressources limitées d'un capteur. En effet, puisque les différents processus ne s'interrompent pas mutuellement, le système n'a nulle besoin d'assigner une pile à chacun d'eux et de sauver leur contexte, au contraire il partage tous la même pile [5]. Mais un tel concept n'est pas totalement dépourvu de problèmes, tant que les tâches restent petites, elles s'exécutent assez rapidement et ne pénalisent pas celles qui sont en attente, mais on ne peut pas dire autant pour les grands processus comme les algorithmes cryptographiques par exemple qui monopoliseront le CPU [5]. Pour finir certaines applications ont été conçues pour tourner dans des environnements multitâches et leur portage sur un système pilotés par les évènements risque d'être compliqué.

V.4.3 Contiki : un Hybride

Les systèmes d'exploitation piloté par les threads ont l'inconvénient de consommer beaucoup de mémoire ce qui a conduit à l'apparition des systèmes piloté par les évènements qui ont pâlit le problème de la mémoire. Néanmoins, dans un système piloté par les évènements un processus long monopolisera le CPU et dans ce cas il est préférable d'avoir l'aspect préemptif des systèmes multitâches [5]. Cette constatation a encouragé l'institut Suédois de l'informatique SICS ainsi que plusieurs contributeurs menés par Adam Dunkels à concevoir un système d'exploitation dit *Hybride*, c'est-à-dire qu'il combine les deux techniques précédemment expliquées à savoir le thread-driven et l'even-driven [5].

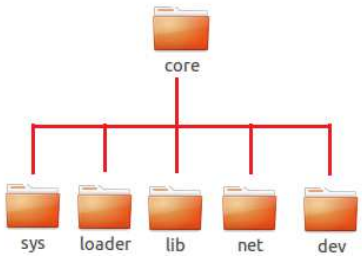
Dans sa configuration de base Contiki est un système piloté par les évènements, mais il supporte également le multithread préemptif, ce dernier n'est pas directement implémenté au niveau de noyau mais comme une bibliothèque qui peut être jointe à n'importe quel processus qui le demande explicitement [5]. Un processus peut être une application ou un service. Un service implémente une fonctionnalité utilisée par plusieurs processus. Tous les processus, les applications et les services, peuvent être dynamiquement substitués à d'exécution. Le noyau ne fournit pas une couche d'abstraction de matériel, mais laisse des pilotes de périphériques et des

applications communiquer directement avec le matériel [5]. La communication interprocessus est effectuée en postant des évènements.

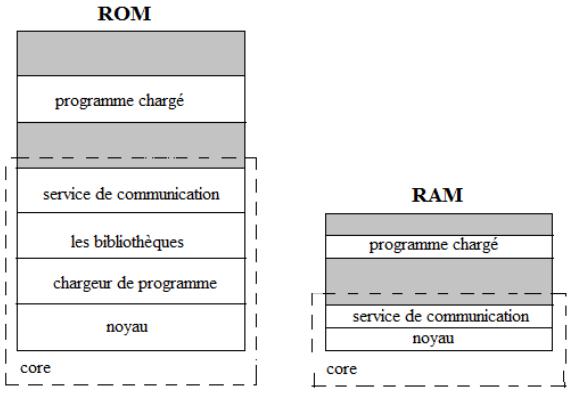
Typiquement, le cœur (*core*) de Contiki consiste en [5]:

- le noyau (*kernel*) de Contiki (*sys*);
- le chargeur de programme (*loader*);
- des bibliothèques (*lib*);
- une pile de communication (*net*) ;
- les pilotes des périphériques pour la communication hardware (*dev*).

Ces modules sont mis en évidences par la figure 1.6 qui représente l'arborescence de code source Contiki.



Contiki est divisé en deux parties, le cœur et les programmes chargés [5], ce partitionnement est effectué à la compilation. Sur le nœud qui exécute Contiki le *core* est compilé dans une image binaire simple qui est enregistrée dans les dispositifs avant le déploiement, comme le montre la figure 1.7. Cette même figure montre aussi le programme chargé, ce dernier est chargé en RAM par le chargeur de programme (*programs loader*) soit à partir de la mémoire de masse (ROM) ou encore via le réseau.



chargés [5].

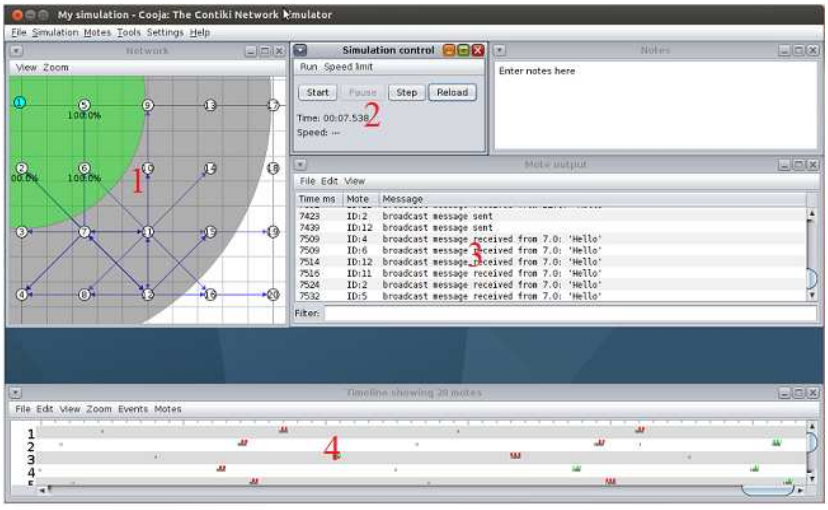
Le noyau Contiki consiste en un ordonnanceur (ou *scheduler*) léger d'évènements. L'éxécution de tout processus est déclenchée par un evènement acheminé par le noyau. Le noyau supporte deux types d'évènements : synchrones et asynchrones. Les évènements asynchrones sont mis dans une file est acheminés au processus cible un peu plus tard. Les évènement synchrones sont acheminé au processus émmidiatement. Le contrôle revient au processus qui a posté l'évènements mais seulement après que le processus cible l'aura traité. Ceci peut être vu comme des appels de procédure inter-processus [5].

Dans les système temps réel une intérruption doit être traitée dans les plus bref délais[5], or dans Contiki un évènement ne peut pas intérrempre un autre, pour que contiki soit en mesure de suppoter le temps réel, une intérruption ne déclanche pas un evènement mais positionne un drapeau et le système traitera cet intérruption plus rapidement lorsqu'il va prendre en compte le changement de ce drapeau [5].

Par ailleurs, Contiki fournit une large gamme de dispositifs pas nécessairement prévus dans une basse empreinte mémoire, comme un interpréteur de commandes interactif, un navigateur web et un gestionnaire de fichiers flash [5].

Plus important maintenant, la communication dans contiki est assurée par deux piles de protocoles : uIP (micro-IP) et Rime. uIP est une petite pile TCP/IP conforme aux RFCs (Request For Comment) d'IETF (Internet Engineering Task Force), elle offre des protocoles comme : DHCP, HTTP, TCP, UDP, ARP, etc. Cette pile rend Contiki capable de communiquer sur internet. Rime est une pile légère conçue pour les radio de basse puissance elle fournit une large gamme de primitives pour la tranmission réseau [5].

Contiki vient également avec un certain nombre de simulateurs de réseau, dont le simulateur basé sur java *Cooja* est le plus avancé [69]. Cooja permet de simuler un capteur qui exécute le système Contiki, Cooja supporte plusieurs plateformes, ESB, Sky, MicaZ, etc. le simulateur contient un certain nombre de modules, les plus important sont numérotés sur la figure 1.8 :



8 : le simulateur de réseau de capteur Cooja.

1. Network : la fenêtre Network permet de visualiser la topologie du réseau simulé, on peut voir les capteurs ainsi que leurs adresses, positions, portées et le trafic radio, etc.
2. Simulation control : comme son nom l'indique, cet outil permet de contrôler la simulation, on peut la stoper, la suspendre ou la recharger, on peut également changer la vitesse de la simulation en l'accélérant ou la ralantissant.
3. Mote output : ce module est très important car c'est ici que sont imprimées les sorties des capteurs.Un champ de text permet d'enter un filtre pour cibler un capteur ou un type de message en particulier.
4. TimeLine : affiche une ligne de temps pour chaque capteur dans la simulation. Sur la ligne de temps, l'état de puissance de l'émetteur récepteur par radio de chaque capteur est affiché dans le codes couleur suivant : blanc si la puce réseau est éteinte, gris s'il est allumé. Les transmissions et les réceptions par radio sont affichées dans la même ligne de temps : les transmissions sont bleues et les réceptions sont vertes, une collision est affichée en rouge.

Le tableaux suivant décrit un bref comparatif entre les 3 systèmes d'exploitation précedents :

Approches	Avantages	Désavantages
Thread-driven (MANTIS)	– Les processus lents ne monopolisent pas le CPU.	– Necessite plus de mémoire RAM pour fonctionner.
Even-driven (TinuOS)	– Necessite moins de mémoire RAM pour fonctionner.	– Les processus lents monopoliseront le CPU.
Hybride (Contiki)	– Les processus lents ne monopolisent pas le CPU. – Necessite moins de mémoire RAM pour fonctionner.	– Contiki reste, nativement, un système d'exploitation basé sur les événements. Pour obtenir le mode multitâche, une bibliothèque doit être installée.

Tableau 1.4 : les principales différences entre les trois OSs.

C'est le système Contiki et le simulateur Cooja qui ont été retenus pour réaliser notre travail, nous allons revenir dans le chapitre 4 plus en détails sur Contiki et l'annexe A détaille comment installer un envirenement de programmation Contiki sous Linux.

VI. Application des réseaux de capteurs

Les applications des réseaux de capteurs sont nombreuses et variées, elles peuvent être placées en cinq grandes familles d'applications [10] : militaire, habitat, environnementale, santé et les applications commerciales. Pour chaque application un cahier de charge est défini, il comprend le nombre et le type de capteurs approprie, la topologie adéquate, etc. Cette section propose de découvrir une partie de ces applications.

VI.1 Applications militaires

Les réseaux de capteurs sans fil se forment par le déploiement d'une centaine voir des milliers de capteurs peu coûteux, le réseau lui-même est tolérant aux pannes et capable de se réorganiser. Ces caractéristiques intéressent particulièrement les militaires pour remplacer les approches traditionnelles qui sont basées sur des dispositifs portant les mécanismes coûteux de protection.

L'application militaire typique des réseaux de capteur est : la surveillance des forces [5]. La surveillance des forces consiste en le suivi continu de statut et de placement des troupes, chaque troupe et véhicule peuvent être équipés de petits capteurs qui recueillent des informations et les envoient à un nœud spécial qui les enregistre et les organise. De cette façon, les commandants peuvent consulter cette base de données à jour.

Les champs de bataille peuvent être rapidement couverts par des capteurs car ces derniers peuvent être mis en place en les dispersant aléatoirement, les capteurs étant dotés de mécanismes d'auto-organisation, Chaque capteur se réveille, trouve ses voisins et coordonne avec eux afin de mettre l'endroit sous surveillance. La destruction d'un certain nombre de capteurs est tolérable, car les capteurs restant vont s'adapter au changement de la topologie.

VI.2 Applications commerciales

Se fiant à la supposition que n'importe quelle technologie peut être une issue commerciale, un réseau de capteur est une technologie tout à fait flexible qui permet d'introduire les améliorations en application commerciales et d'en tirer bénéfice [10]. Certaines des applications commerciales incluent : le contrôle de l'environnement dans des immeubles de bureaux, musées interactifs, détection industrielle.

Le contrôle de l'environnement dans des immeubles de bureaux : la climatisation et la chaleur de la plupart des bâtiments sont centralement commandées [10]. Ce qui peut se produire est que la température à l'intérieur de chaque salle peut varier par peu de degrés, cela est dû aux conditions particulières (exposition au soleil, fenêtres, isolement). Un réseau de capteurs sans fil peut apporter une haute efficacité dans la distribution de l'air conditionné dans un tel environnement uniforme. En effet en plaçant un capteur dans chaque salle on prélève la température exacte de façon à contrôler le flux d'air et la température dans différentes parties de l'immeuble.

Musées interactifs : ils existent des musées scientifiques qui offrent aux visiteurs la possibilité d'assister à certaines petites expériences ou l'observation de phénomènes. Le réseau de capteur peut aider à apporter l'interactivité entre les visiteurs et le musée et les laisser ainsi apprendre plus. Les réseaux de capteurs sans fil peuvent fournir la localisation à l'intérieur du musée. Le fait qu'un visiteur s'approche d'une expérience déclenche cette dernière et son éloignement l'arrête [10].

Détection industrielle : l'industrie est toujours intéressée à utiliser des capteurs en tant que moyen au coût inférieur pour la maintenance et en même temps l'amélioration de fonctionnement des machines. La « santé » des machines industrielles peut être surveillée par des

capteurs qui peuvent être insérés dans des placements au niveau des machines inaccessibles pour les humains. Par exemple pour déterminer les vibrations (cause principale des défaillances matérielles) ou des niveaux de lubrification. Cela permet assurer la conformité à de nouvelles directives de sûreté émises par l'autorité de l'État tout en maintenant des frais d'installation bas [10].

VI.3 Applications d'habitats

Le terme « habitat » peut être interprété de deux façons différentes. Il peut se rapporter à des bâtiments en général ou aux environnements naturels occupés par des animaux ou des plantes [10]. Les premiers sont généralement liés au concept des environnements intelligents. La seconde sont souvent une partie des intérêts des membres de la communauté des sciences de la vie. Nous discutons dans cette section les deux cas.

La raison principale qui fait que les capteurs ont un intérêt croissant pour les sciences de la vie est les incidences humaines sur les environnements naturels. L'approche traditionnelle employée par des chercheurs est de collecter des mesures en interagissant eux même avec l'écosystème [10]. Cette interaction humaine est nocive, et même parfois, elle a une modification complète de comportements des espèces étudiées. L'utilisation des réseaux de capteurs pour automatiser la collecte de ces informations réduit considérablement les risques de perturbation des milieux naturels.

Une place particulière dans laquelle les réseaux capteurs sans fil jouent un rôle important est ce que les chercheurs appellent les environnements intelligents [10]. Un environnement intelligent est un environnement qui peut identifier des personnes, interpréter leurs actions, et réagir convenablement. À mesure que le nombre des appareils numériques augmente dans notre environnement, la quantité d'attention que l'utilisateur peut assigner à chacun est nécessairement réduite, de ce fait, ces derniers doivent devenir de plus en plus «intelligents» afin d'anticiper et résoudre les problèmes de l'utilisateur avec un plus grand niveau autonomie.

VI.4 Applications environnementales

Détection d'incendie : puisque des nœuds capteurs peuvent se déployer aléatoirement et en masse dans une forêt, ils peuvent aider à déterminer l'origine exacte d'un feu avant que sa diffusion devienne incontrôlable. Un nombre très important de nœuds de capteur peut être déployé tout en étant équipés par exemple de panneaux photovoltaïques [10] pour être opérationnels pendant de longues périodes (des mois et même des années). Les nœuds capteurs collaboreront les uns avec les autres pour détecter et acheminer l'information à une station de base dans les plus brefs délais.

Agriculture de précision : en déployant des réseaux de capteur dans des zones agricoles, pour surveiller en temps réel plusieurs paramètres, tels que le niveau des pesticides dans l'eau potable, le niveau de l'érosion du sol et la pollution atmosphérique deviennent beaucoup plus facile à détecter [10].

V.5 Applications de santé

Les réseaux de capteurs peuvent contribuer à améliorer la santé des patients et le travail des médecins. Un patient dans son domicile ou au travail peut être équipé d'un capteur sans fil intelligent capable d'acquérir et d'analyser les signes vitaux de ce patient [10], par exemple ses signaux électrocardiogrammes (ECG) ou sa tension artérielle. Ces information sont soumises aux médecins dans des hôpitaux par internet ce qui permet au patient d'avoir une vie quotidienne plus au moins normale et au médecin de suivre la maladie de plus près et planifier une intervention rapide en cas de malaise [10].

VII. Les problématiques dans les réseaux de capteurs

Un grand chemin a été parcouru depuis l'apparition des réseaux de capteur sans fil, en effet, ces derniers sont utilisés dans de nombreux domaines d'applications et ils disposent maintenant de plusieurs systèmes d'exploitation et protocoles créés spécifiquement en tenant compte des ressources limitées d'un capteur. Mais il existe toujours des défis auxquels les chercheurs font face, comme la synchronisation, le routage, etc.

VII.1 La localisation

En l'absence d'information sur la position des nœuds d'un réseau de capteurs sans fil, au sein de l'environnement où ils sont déployés, les données récoltées peuvent s'avérer d'une utilité limitée [11]. Pour localiser un capteur nous pouvons dans un premier temps penser à lui intégrer un récepteur GPS (Global Positioning System) mais cette technologie et couteuses et consomme plus d'énergie [1]. Comme alternative les chercheurs ont pensé à des techniques se basant sur la puissance de signal reçu comme le RSSI mais là encore les obstacles et la qualité du module radio font que cette technique soit remise en question dans certains cas [11].

VII.2 Le routage

Dans un réseau Ad hoc l'isolement d'un nœud ou la désunion du réseau sont des évènements fréquents à cause de la mobilité des terminaux. Dans les RCSFs ce phénomène est accentué par le fait de la fragilité des capteurs et la faible portée de leur module radio. D'où la nécessité d'adapter les protocoles de routage pour les réseaux Ad hoc aux RCSFs. Un compromis doit être trouvé entre minimiser le nombre de capteurs hors service et ne pas faire peser la totalité des traitements sur les nœuds les plus valides [1].

Dans le cas où l'application n'exige pas que les capteurs soit mobiles, le protocole de routage peut se concentrer sur le partage de la charge sur les capteurs voisins mais il doit dans un premier temps partir à la découverte de ces derniers. Les techniques de localisation peuvent aider à améliorer les protocoles de routage, connaissant la position des capteurs le protocole peut choisir les nœuds intermédiaires les plus adaptés pour acheminer le paquet à la destination [1].

VII.3 La syncronisation temporelle

Pour certaines applications il est nécessaire que les capteurs disposent de la même horloge pour avoir une chronologie des évènements corrects. Par exemple dans une application de collecte de données avec une forte contrainte temporelle, une horloge avancée d'un capteur B par rapport au capteur A mènera le capteur B à supprimer des paquets reçu depuis A en les considérons comme obsolète alors qu'ils sont récent. Cette synchronisation peut être obtenue à l'aide du GPS [1]. Toutefois, comme pour la localisation, le coût d'un tel système aussi bien d'ordre financier qu'énergétique n'est pas anodin. Par conséquent, différents protocoles ont été développés pour répondre à cette problématique. Certains sont des adaptations de protocoles existants dans les réseaux filaires. Cette transition n'est pas toujours possible comme l'illustre la complexité voire l'impossibilité d'utiliser la protocole (S) NTP ((Simple) Network Time Protocol) car nécessitant une puissance de calcul importante [12].

VIII. Conclusion

Un capteur sans fil est un instrument doté de capacités de captage, traitement et transmission sans fil, ces derniers utilisent des concepts déjà maitrisés dans l'informatique en générale et dans les systèmes embarqués en particuliers, quoique, l'excès dans la miniaturisation pour réduire le coût limite fortement les ressources d'un capteur ce qui a mené à repenser les solutions déjà existantes pour les adapter aux RCSFs et d'en créer d'autre en exclusivité pour ces derniers.

Ce chapitre constitue le minimum à savoir à propos des RCSF. Nous avons abordé l'architecture matérielle d'un capteur, les topologies des RCSF, les domaines d'application et les systèmes d'exploitation pour les RCSF. Il manque encore un concept important dans les RCSFs, le concept de partage du support de communication. Nous consacrons le prochain chapitre pour parler sur le standard IEEE 802.11 qui s'articule sur la couche MAC du modèle OSI où décrit le protocole d'accès au support CSMA/CA utilisé par les réseaux locaux sans fil et c'est ce qui nous intéresse pour ce travail.

Chapitre 2

Le Standard IEEE 802.11

I. Introduction

En ce début du 21ième siècle, les réseaux locaux informatiques connaissent deux évolutions importantes. D'une part, l'utilisation courante du réseau local chez les particuliers, due en grande partie à internet, et d'autre part, l'arrivée en masse des ordinateurs et autres matériels mobiles. Pour cela il faut trouver une technologie permettant de simplifier le câblage du réseau chez un particulier et de préserver la mobilité des produits portables. Un seul principe permet de concilier les deux, le sans fil. [22]

IEEE 802.11 est un standard de réseau sans fil local proposé par l'organisme de standardisation Américain IEEE (Institute of Electrical and Electronic Engineers). La technologie 802.11 est généralement considérée comme la version sans fil de 802.3 (Ethernet). [23]

L'objectif de ce chapitre est de présenter en première lieu les différentes normes de l'IEEE en mettant le point sur les protocoles d'accès au support, en deuxième lieu nous détaillerons le standard 802.11 qui est le plus utilisé dans les réseaux locaux sans fil. Pour cela, nous commencerons par décrire les topologies suivant lesquels les WLAN 802.11 fonctionnent. Ensuite, nous présenterons les caractéristiques liées à l'architecture logique de la norme (couche physique et couche MAC).

II. Les Normes IEEE 802

II.1 Normalisation IEEE

Les réseaux informatiques doivent permettre à des applications informatiques de coopérer sans avoir à tenir compte de l'hétérogénéité des moyens et procédés de transmission (par exemple : de la topologie, des méthodes d'accès, des caractéristiques des équipements ou des supports, etc.). La normalisation est un ensemble de règles établies qui doivent être suivies par les entités désirant communiquer. En 1980, l'IEEE a créé le comité d'étude 802, chargé de définir des normes pour les réseaux locaux, afin d'assurer la compatibilité entre les équipements provenant de différents constructeurs. Une famille de normes a été élaborée.

Toutes les normes du comité IEEE 802 ont été reprises et complétées par l'ISO sous la désignation ISO8802, elles correspondent aux couches physique et liaison de données du modèle de référence OSI. L'IEEE a défini les fonctionnalités de la sous-couche LLC dans la norme 802.2 et celles de la sous-couche MAC (couche Physique) dans les normes 802.3, 802.4, et 802.5. L'architecture IEEE 802 est décrite par la figure 2.1 : [25]

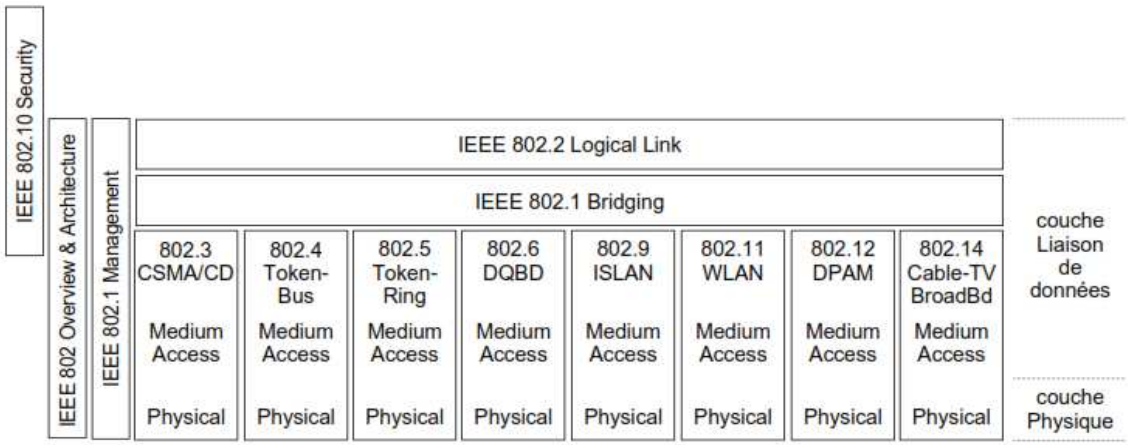


Fig. 2.1 Liste des principales normes IEEE 802 [25].

II.2 Les normes IEEE 802.x

Le comité 802 n'a pas donné naissance à unique standard, mais à une série de standards pour couvrir l'ensemble des besoins. C'est ainsi que plusieurs sous-comités ont été crées chacun traitant d'un sujet particulier dans lequel une norme est élaborée, voici une liste sélective des ces normes :

- **II.2.1 La norme IEEE 802.2**

  Elle est relative au contrôle de liaison logique (LLC "Logical Link Control"). La sous couche LLC offre la possibilité de définir des liaisons de données logiques entre toute

paire de nœuds, elle permet de masquer la méthode d'accès. Trois types de services ont été prévus : [26]

- LLC 1 : sans connexion, non fiable,
- LLC 2 : avec connexion, conservation de l'ordre d'émission, fiable,
- LLC 3 : sans connexion, la récupération des erreurs est décidée par l'émetteur.

➢ II.2.2 La norme 802.3

La norme 802.3 définit des réseaux locaux dont le support choisi est un bus logique auquel sont connectés tous les éléments actifs et utilisant le protocole d'accès au support CSMA/CD (*Carrier Sense Multiple Acces/ Collision Detection*), tel que tous les ordinateurs du réseau 'écoutent' le support afin de savoir s'il y a du trafic sur le réseau :

- Un ordinateur détecte que ce câble est libre (il n'y a pas de trafic),
- L'ordinateur peut émettre des données,
- Si les données se trouvent déjà sur le câble, aucun autre ordinateur ne peut transmettre tant que les données n'ont pas atteint leur destination et que le câble n'a pas été libéré.
- Si deux (ou plusieurs) ordinateurs envoient des données sur le câble au même instant, il y aura collision. En pareil cas, les ordinateurs cessent de transmettre pendant une durée aléatoire, puis ils essaient de reprendre la transmission. Après 16 tentatives d'émission d'un même message, l'émetteur abandonne l'émission. [27]

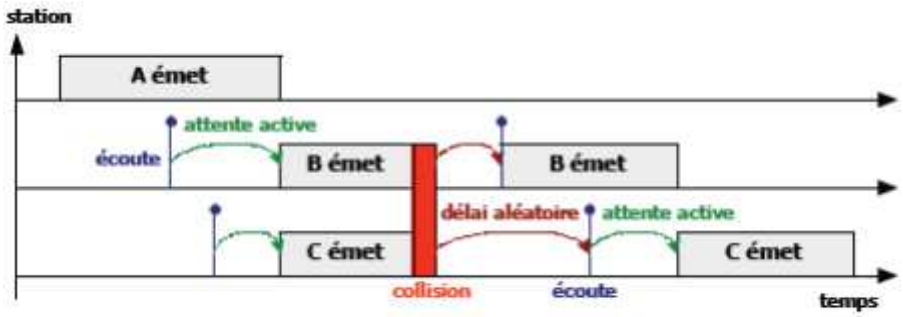


Fig. 2.2 Principe du CSMA/CD [27].

➢ II.2.3 La norme 802.4

La norme 802.4, apparu en 1985, définit des réseaux de type bus à jeton [59].

La gestion du bus est confiée à un élément actif particulier appelé superviseur. Ce dernier s'occupe de la création du jeton à l'initialisation du réseau et de sa circulation entre les ordinateurs, Une station qui reçoit et reconnaît le jeton émis par la station précédente peut alors accéder au support de transmission. Le superviseur détermine aussi un ordre cyclique de passage du jeton entre tous les postes connectés.

La figure 2.3 décrit le principe générale d'accès par jeton :

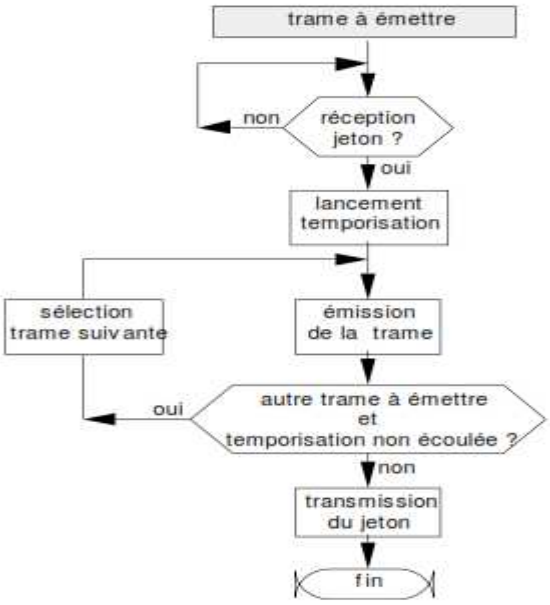


Fig. 2.3 Principe d'accès par jeton.

➢ II.2.4 La norme 802.5

Au milieu des années 80, IBM crée une architecture de type anneau à jeton (Token Ring), l'IEEE normalise les réseaux en anneau à jeton sous la norme 802.5 [60].

Le fonctionnement d'un réseau de type anneau à jeton est le même que celui d'un réseau de type bus à jeton, mais le support physique en bus est remplacé par un anneau.

➢ II.2.5 La norme 802.6

La norme IEEE 802.6 [61] définit la méthode d'accès DQDB (*Distributed Queue Dual Bus*) pour les réseaux métropolitains (MAN). La topologie générale repose sur deux bus unidirectionnels (câble coaxial ou fibre optique), de sens inverses, auxquels sont rattachés les nœuds. Chaque bus est muni d'un générateur de trames produisant les trames. La norme IEEE 802.6 est maintenant une ancienne technologie elle est remplacée particulièrement par le Gigabit Ethernet [25] utilisé dans de nombreux MAN.

➢ II.2.6 La norme 802.11

L'EEE a normalisé une architecture de réseaux locaux sans fil basée sur une technologie radio utilise CSMA/CA (*Carrier Sense Multiple Acces/ Collision Avoidance*), comme protocole d'accès au support. Cette norme a été nommée 802.11 [18] qui sera détaillée prochainement.

La norme IEEE 802.11 se décompose en plusieurs normes de transmission, offrant chacune des caractéristiques différentes en terme de fréquence, de débit ou de portée du signal radio.

- **II.2.7 La norme 802.12**

La norme IEEE 802.12 utilise le mécanisme de demande de priorité pour l'accès aux médias à 100 Mbit/s. Elle est aussi appelée 100VG-Any-LAN [28].

Où,

- VG (Voice Grade) : Priorité à la demande.
- Any-LAN : accepte tous les réseaux (Trame Ethernet ou Token Ring).

La topologie physique est en étoile hiérarchique [28].

III. Le standard IEEE802.11

III.1 Généralités

IEEE802.11 définit une architecture cellulaire dont les terminaux munis d'une carte d'interface réseau 802.11, s'associent à un point d'accès (mode infrastructure) ou entre eux (mode ad-hoc), la zone occupée par un terminal est dénommée cellule.

Le standard IEEE802.11 définit les deux premières couche (basses) du model OSI, à savoir la couche physique et la couche liaison de données qui est divisée en deux sous-couches : la sous-couche MAC (*Medium Accès Control*), dont le rôle est de permettre le partage du support, c'est-à-dire celui de la bande passante, par plusieurs utilisateurs en utilisant le protocole CSMA/CA et la sous-couche LLC (*Logical Link Control*). L'une des caractéristiques essentielles est qu'il définit une seule couche MAC commune à toutes les couches physiques. Ainsi, différentes couches physiques peuvent être développées sans qu'il soit nécessaire de modifier le protocole d'accès au réseau. Ainsi, depuis la première version de la norme 802.11 proposée en 1997 [62], plusieurs extensions présentant principalement des améliorations de la couche physique ont été proposées.

III.2 L'Architecture réseau

Le principe de la cellule est au centre de l'architecture 802.11. Une cellule est la zone géographique dans laquelle une interface 802.11 est capable de dialoguer avec une autre interface 802.11.

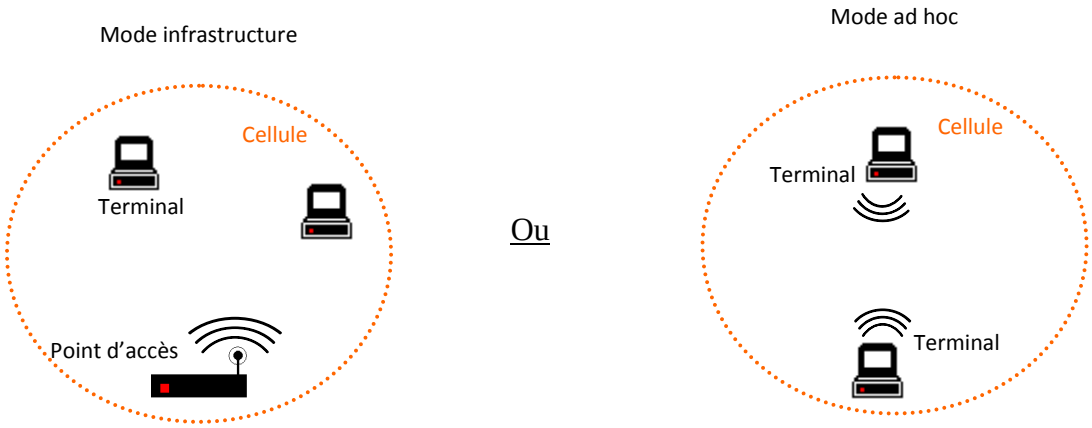


Fig. 2.4 : La notion de cellule en 802.11

Une vue complète des éléments architecturaux proposés par l'IEEE 802.11 peut se résumer par le schéma suivant :

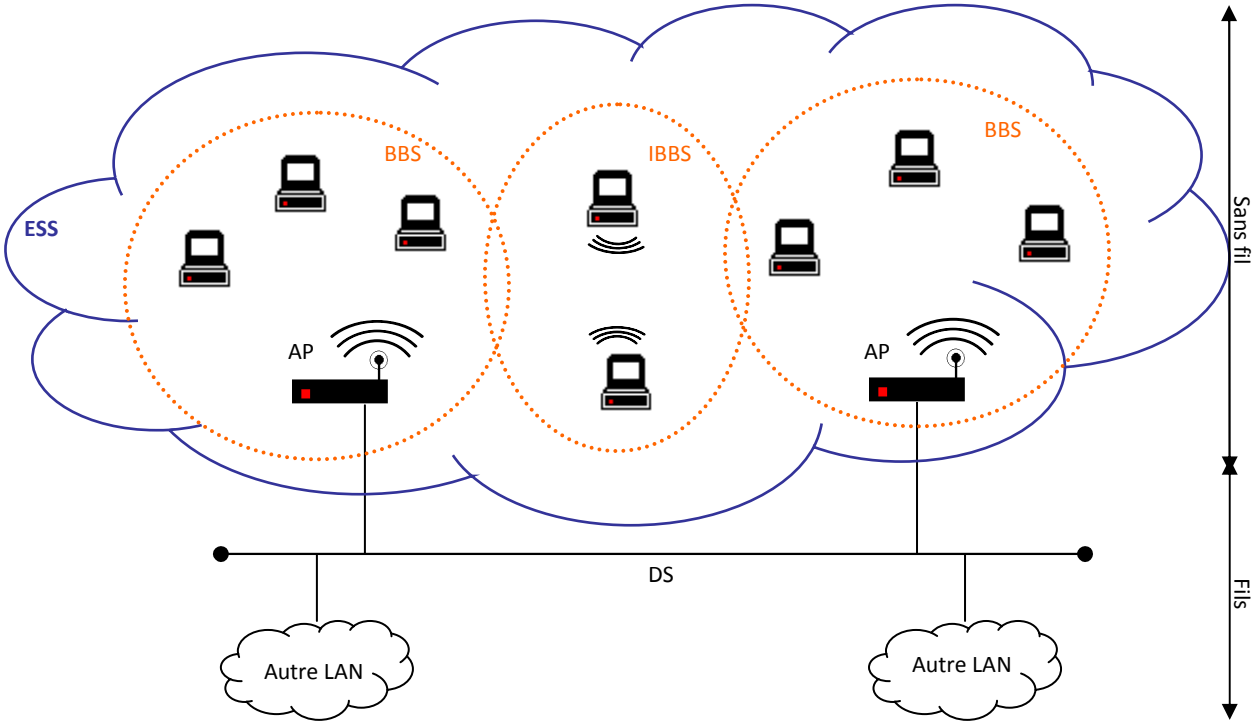


Fig. 2.5: Architecture type d'un WLAN **802.11**

On distingue les éléments suivants :

- la cellule de base appelée BSS (Basic Service Set)
- la cellule de base en mode ad-hoc appelée IBSS (Independant Basic Service Set)
- le point d'accès appelé AP (Acces Point)
- l'ensemble du réseau sans fil appelé ESS (Extented Service Set)
- l'épine dorsale appelée DS (Distribution System)

III.3 L'architecture en couches

La norme IEEE 802.11 couvre les couches : physique et liaison de données. Le schéma suivant présente les couches en question, positionnées par rapport au modèle de référence OSI de l'ISO : [24]

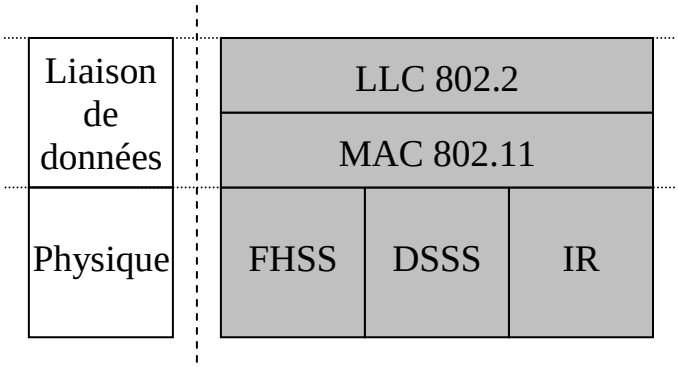


Fig. 2.6 : Situation de la norme 802.11

III.3.1 La couche physique

La couche physique a pour but de véhiculer le flux binaire de la station émettrice jusqu'à la station réceptrice. Ainsi, selon l'extension de la norme employée, ce n'est pas le même type de couche de physique qui est utilisé. Une architecture de couche physique définit : le format des trames de transmises sur le canal, la technique de transmission et le type de modulation utilisé. Deux types de sous-couches ont été définis [14]:

- **PMD (*Physical Medium Depender*)** : gère l'encodage des données et la modulation.
- **PLCP (*Physical Layer Convergence Protocol*)** : s'occupe de l'écoute du support, elle est directement reliée à la couche MAC pour lui signifier que le support de transmission est libre ou pas.

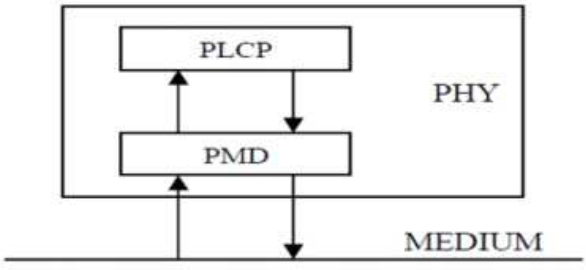


Fig. 2.7 : Les deux sous couches physique du standard 802.11 [14].

Le standard 802.11 d'origine a défini trois couches physiques de base, FHSS (Frequency Hopping Spread Spectrum), DSSS (Direct Sequence Spread Spectrum), IR ((Infra Rouge), auxquelles ont été rajoutées trois nouvelles couches physiques Wifi (avec deux variantes au sein de la solution 802.11b) et Wi-Fi5 (802.11a/g).la figure suivante illustre ça :

<table>
<tr><td rowspan="2">OSI Layer 2
Data Link Layer</td><td colspan="6">802.11 Logical Link Control (LLC)</td></tr>
<tr><td colspan="6">802.11 Medium Access Control (MAC)</td></tr>
<tr><td>OSI Layer 1
Physical Layer (PHY)</td><td>FHSS</td><td>DSSS</td><td>IR</td><td>Wi-Fi
802.11b</td><td>Wi-Fi
802.11g</td><td>Wi-Fi5
802.11a</td></tr>
</table>

Fig. 2.8 : Modèle en couches de l'IEEE 802.11 [18].

III.3.2 La couche liaison de données

La couche liaison de données en 802.11 est composée, à l'instar d'autres normes de la famille 802.*x*, des deux sous-couches LLC 802.2 et MAC 802.11. [13]

- **La couche LLC** (Logical Link Control) normalisée 802.2 permet de relier un WLAN 802.11 à tout autre réseau respectant l'une des normes de la famille 802.*x*. La sous couche LLC a été crée afin de permettre à une partie de la couche liaison de données de fonctionner indépendamment des technologies existantes. Cela assure la polyvalence des services fournis aux protocoles de couche réseau situés en amont de cette couche tout en communiquant avec les différentes technologies utilisés pour véhiculer les informations entre la source et la destination.

- **La couche MAC** : Cette couche met en place le protocole d'accès au canal, et propose deux modes d'accès :

1. **Le mode PCF (*Point Coordination Fonction*)** : un mode d'accès au canal sans contention dans lequel les stations de bases ont la charge de gestion de l'accès au canal dans leur zone de couverture pour les mobiles qui leur sont rattachés. Une station ne peut émettre que si elle est autorisée et elle ne peut recevoir que si elle est sélectionnée. Cette méthode est conçue pour les applications temps réel (vidéo, voix) nécessitant une gestion du délai lors des transmissions de données.

2. **Le mode DCF (*Distriduted Coordination Function*)**: propose un accès équitable au canal radio dont la gestion est réalisée de façon totalement distribuée entre les nœuds du réseau.

IV. DCF

DCF est un mode d'accès permet de donner la parole aux différents nœuds du réseau par la mise en place d'une technique d'accès distribuée proche des techniques d'accès à compétition que l'on peut trouver dans les communications filaires (Ethernet 802.3). DCF dispose de deux modes de fonctionnement : l'accès en mode station de base (CSMA/CA) et le mode RTS/CTS.
Le mode RTS/CTS a été conçu pour palier au problème des nœuds cachés survenant principalement en mode ad-hoc.
Le mode DCF est composé des éléments suivants :

- Le mécanisme CSMA/CA,
- Les durées IFS,
- Le tirage aléatoire de backoff.

IV.1 CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance)

Dans un réseau local Ethernet classique, la méthode d'accès utilisée par les machines est le ***CSMA/CD*** pour lequel chaque machine est libre de communiquer à n'importe quel moment. Chaque machine envoyant un message vérifie qu'aucun autre message n'a été envoyé en même temps par une autre machine.
Si c'est le cas, les deux machines patientent pendant un temps aléatoire avant de recommencer à émettre.

Dans un environnement sans fil, la méthode d'accès utilisée par les machines est le ***CSMA/CA***. Ce protocole utilise un mécanisme d'évitement de collision basé sur l'écoute du support afin de savoir s'il est libre ou pas avant l'émission, et sur le principe d'accusé de réceptions (ACK) réciproques entre l'émetteur et le récepteur pour pouvoir détecter s'il y a lieu une collision en non réception de cet ACK, Si l'émetteur ne reçoit pas l'accusé de réception, alors il retransmet le fragment jusqu'à ce qu'il l'obtienne ou abandonne au bout d'un certain nombre de retransmissions.

a) Déroulement du protocole CSMA/CA

Une station voulant émettre des données commence par déterminer si le médium est libre :

1) si le médium est libre après un temps spécifique (DIFS), elle peut envoyer ses données immédiatement. Si la station reçoit un acquittement (ACK) donc l'émission réussie, sinon échec d'émission : pour la retransmission aller à 2, si elle n'a pas atteint le nombre de retransmissions permis.

2) si le médium est occupé elle doit différer cet envoi de la manière suivante [13]:

i) attente jusqu'à ce que le médium redevienne libre,

ii) puis attente d'un temps DIFS + un temps aléatoire (*Backoff*) de CW slots (Contention Windows, fenêtre de contention) -sera détaillé dans la suite du chapitre-,

iii) si, à la suite de cette attente, le médium est toujours libre, la station a gagné l'accès au médium et peut ainsi émettre ses données ; Si la station reçoit un acquittement (ACK) donc l'émission réussie, sinon échec d'émission : pour la retransmission aller à 2, si elle n'a pas atteint le nombre de retransmissions permis,

iv) si, elle est dans la phase de fenêtre de contention, le médium est repris par une autre station, le processus est abandonné jusqu'`a la prochaine période (aller à i). Cependant, la station mémorise le temps qu'il lui restait à attendre, pour reprendre là où elle était restée, lors de la prochaine négociation d'accès au médium.

3) Quand une station cible reçoit une trame, elle doit envoyer un accusé de réception à l'émetteur après avoir attendre un temps SIFS.

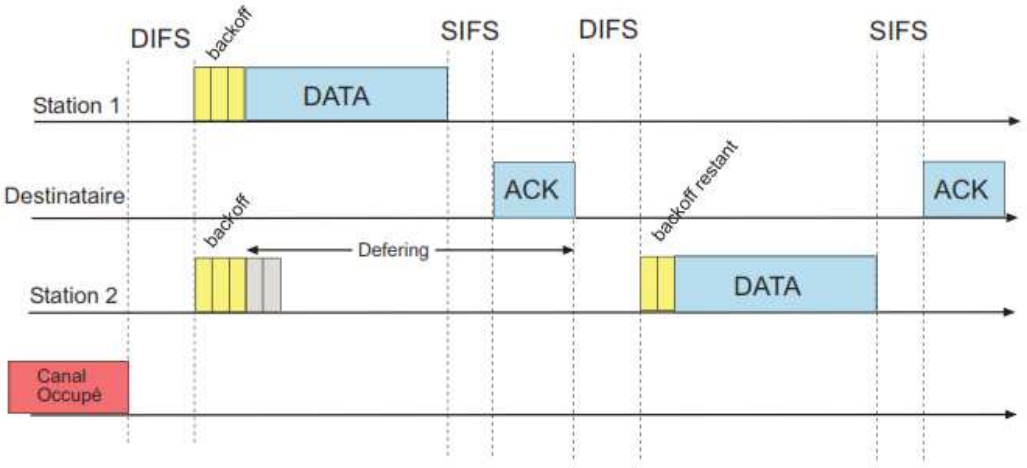


Fig. 2.9 : Accès au médium en mode CSMA/CA [13].

b) Organigramme de la procédure CSMA/CA :

La figure 2.10 résume le fonctionnement de la procédure CSMA/CA et de l'algorithme de Backoff.

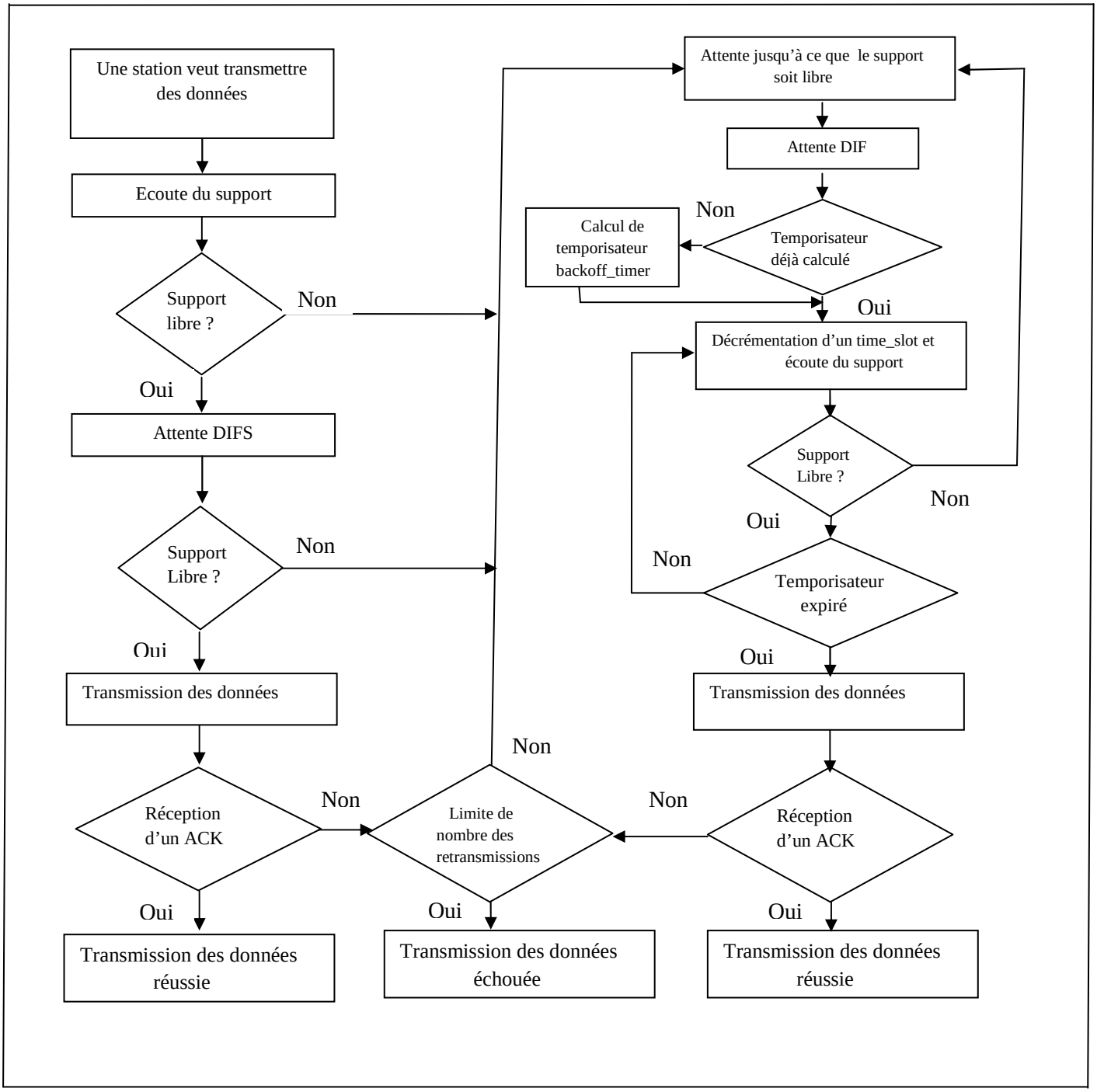


Fig. 2.10 : la Procédure CSMA/CA

c) Le mécanisme RTS/CTS

Les réseaux locaux sans fil sont victimes d'un phénomène appelé « *station cachée* » (*hidden station*). Considérons trois stations A, B et C : B peut communiquer avec A et C, mais A et C ne peuvent communiquer ensemble, par exemple parce qu'elles sont trop éloignées l'une de l'autre ou parce qu'un obstacle les sépare.

Pour contrer ce phénomène très fréquent en sans fil, en ajoutant le mécanisme RTS/CTS à la technique CSMA/CA [18] qui repose sur le principe suivant : avant de pouvoir envoyer ses données (trame DATA), une station devra d'abord envoyer une trame RTS (Request To Send) contenant l'adresse de la station de destination et la durée approximative de l'´emission (NAV, Network Allocation Vector), jusqu''a l'acquittement ACK ; la station de destination répond à ce message par un CTS (Clear To Send), en diffusant elle aussi l'information de durée contenue dans le RTS. Ainsi, toutes les stations à portée de la source comme de la destination sont informées qu'elles ne doivent pas utiliser le médium pendant le temps annoncé dans les trames RTS et CTS. La figure 2.11 illustre un réseau composé de trois stations A, B et C dont deux (A et C) ne se voient pas à cause d'un obstacle.

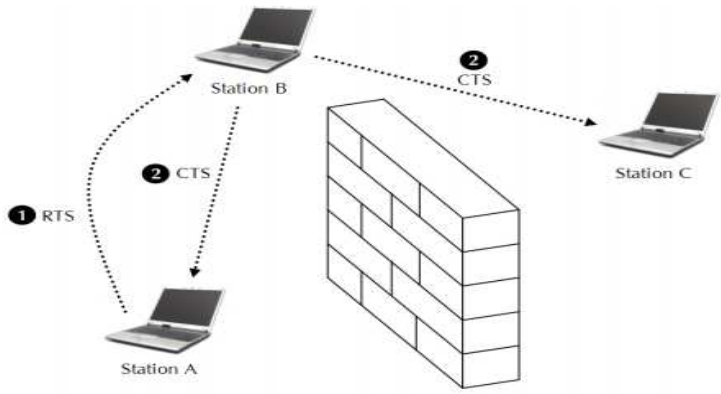


Fig. 2.11 : Topologie présentant deux terminaux mutuellement cachés

Au niveau de l'accès au médium, la figure 2.12 illustre le déroulement temporel de l'activité du médium.

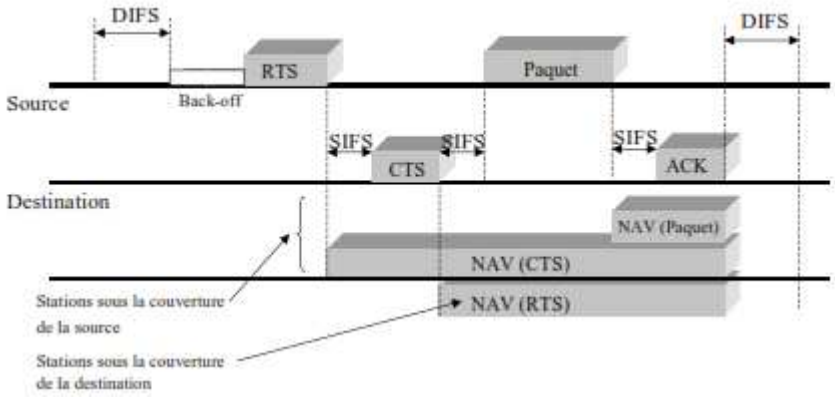


Fig. 2.12 : fonctionnement de RTS/CTS [18]

IV.2 Les durées IFS

Comme on peut le voir sur la figure 2.9, 802.11préconise l'usage de plusieurs temporisations inter-trame; il en existe quatre [22], de la plus courte à la plus longue :

1. **SIFS**, pour Short Inter-Frame Spacing, la plus courte, précède toutes les trames au sein d'un même échange, i.e. entre le RTS et le CTS, entre le CTS et les données puis enfin entre les données et l'accusé de réception : une fois l'accès au médium remporté, tout l'´echange se fait avec des temps inter-trame prioritaires pour conserver l'accès au médium. Les trames émises avec un temps inter-trame SIFS seront donc les plus prioritaires;

2. **PIFS**, pour Point Coordination Inter-Frame Spacing, précède les trames de polling émises par le point d'accès en mode PCF

3. **DIFS**, pour Distributed Inter-Frame Spacing, est le plus couramment utilisé (avec le SIFS). Il est utilisé en mode DCF comme temps minimal d'attente avant transmission.

4. **EIFS**, pour Extended IFS, est utilisé lorsqu'il y a détection de collision. Ce temps relativement long par rapport aux autres IFS est utilisé comme inhibiteur pour éviter des collisions en série.

Là encore, il est important de respecter ces quatre temps pour conserver un accès au médium équitable et d'éviter les collisions.

Comme nous l'avions énoncé en l'introduisant, la méthode DCF proposée par l'IEEE pour 802.11 ne propose qu'un service de type Best Effort c'est `a dire sans priorité et sans garantie. En effet, même si le dispositif RTS/CTS permet d'´eviter un grand nombre de collisions, certaines sont inévitables, en particulier quand les deux stations pensent en même temps que le médium est libre. Cette méthode d'accès ne présente donc aucune garantie, ni sur le débit, ni sur la latence. En cas de trafic élevé, les performances du réseau s'écroulent.

IV.3 L'algorithme de Backoff

La procédure de Backoff est un mécanisme simple, basé sur le calcul d'un temporisateur gérant les transmissions et les retransmissions. Il permet de réduire la probabilité de collision sur le canal en essayant de minimiser les chances d'avoir plusieurs stations qui accèdent au support en même temps [22].

a) *Déroulement* :

Une station S désirant envoyer des données attend pendant une période DIFS. Si après cette durée le canal est libre, la station accède directement au canal. Dans le cas contraire, la station déclenche le mécanisme de Backoff qui se déroule en 3 étapes :

1- La station calcule son temporisateur Backoff_Timer avec la formule suivante :

$$Backoff\_Timer = Random(\,) \times TS$$

Où

- *Random ( )* : nombre pseudo-aléatoire choisi ente 0 et CW-1 ; où CW est la taille de la fenêtre de contention.
- *TS* : durée d'un time -slot définie comme étant l'intervalle de temps nécessaire pour une station pour savoir si une autre a accédé au canal au début du time-slot précédant.

2- Quand le canal devient libre, et après un DIFS, la station commence à décrémenter son temporisateur time-slot par time-slot.
3- Lorsque la valeur de Backoff_Timer est égale à 0, la station peut alors envoyer. Si par contre au cours de la phase de décrémentation, une autre station S' termine de décrémenter son temporisateur, la station S bloque son temporisateur. Elle pourra continuer de le décrémenter une fois la transmission de la station S finie.

b) *Fenêtre de contention* :

La taille de la fenêtre de contention CW a pour valeur initiale CWmin. Deux cas de figures peuvent se présenter [19]:

1- Transmission réussie : dans ce cas, CW est réinitialisée à CWmin.
2- Transmission échouée : c'est-à-dire que la station émettrice ne reçoit pas d'acquittement au bout d'un certain temps. CW est alors incrémenté en fonction du nombre de retransmissions consécutives jusqu'à CWmax selon l'algorithme du Binary Exponential Backoff(BEB) [19] [22]. C'est à dire qu'à chaque retransmission consécutive il prend la valeur suivante dans la suite croissante d'entiers de la forme 2^k -1. La valeur de CW est donc doublée à chaque tentative de retransmission jusqu'à ce qu'elle atteigne la valeur CWmax.

$$CWnew = 2^k - 1$$

K : nombre de tentative de retransmission.

La station suppose dans ce cas qu'il y a eu collision lors de la transmission, et incrémente la taille de sa fenêtre de contention afin de diminuer les chances de collisions lors des prochaines retransmissions. Une valeur limite CWmax est cependant définie.

Si pour CW = CWmax la transmission échoue toujours, la valeur n'est plus incrémentée et est maintenue à CWmax.

La figure 2.13 montre un diagramme de variations de la taille de la fenêtre de contention en fonction du nombre de tentatives de transmissions.

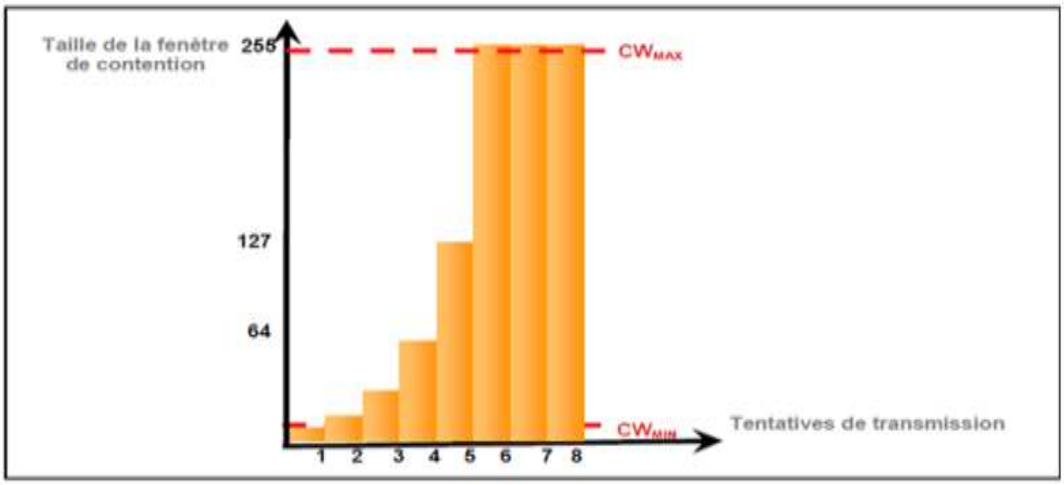


Figure 2.13: Exemple typique de la variation de la taille de la fenêtre de contention

V. Conclusion

Ce chapitre a donc introduit les connaissances de bases sur le standard IEEE 802.11, à savoir les deux couches basses du modèle OSI (MAC et Physique). La couche Mac propose deux mécanismes d'accès au médium différents : *(i)* un mécanisme distribué d'accès au canal (DCF), qui est exécuté indépendamment au niveau de chaque station sans fil; *(ii)* un mécanisme basé sur une station centrale responsable de gérer l'accès au canal (PCF). Pour détecter les collisions, les WLAN 802.11 utilisent une méthode proche de celle des réseaux Ethernet 802.3: l'accès partagé par l'écoute de la porteuse, ou CMSA/CA. Cela consiste, pour une station, à écouter le support pour détecter s'il y a un signal porteur et attendre, si c'est le cas, qu'il soit libre avant de transmettre. Sur un réseau Ethernet, lorsque deux stations émettent simultanément, le niveau du signal sur le câble augmente, leur indiquant qu'une collision à lieu. Les WLAN n'ayant évidemment pas cette capacité, le mécanisme d'accès est pensé pour éviter les collisions.

Chapitre 3

La Localisation

I. Introduction

Le développement des communications et des réseaux sans fil s'est considérablement accéléré ces dernières années. De plus, la possibilité de connaître le contexte et de s'en servir dans les processus de communication est désormais devenue un besoin crucial. La localisation physique est l'une des plus importantes composantes de la connaissance du contexte. De nombreuses méthodes de localisation dans les réseaux sans fil ont été développés et plusieurs systèmes sont déjà commercialisés et répandus.

La plupart des procédés employés pour déterminer une position sont basés sur des calculs géométriques comme la triangulation (en mesurant des angles par rapport à des points fixes ou des nœuds connaissant leur position) et la trilatération (en mesurant la distance entre les nœuds). Pour connaître la distance entre deux nœuds, plusieurs techniques peuvent être utilisées, comme la synchronisation, la puissance de signal reçu ainsi que les caractéristiques physiques de l'onde porteuse. D'autres approches, comme les caractéristiques du signal radio reçu et l'angle de l'arrivée peuvent être également appliquées pour le calcul de position.

Les techniques de localisation dans les réseaux de capteurs sans fil sont utilisées pour estimer l'emplacement des capteurs sans position connu auparavant dans le réseau en utilisant les informations de position de quelques capteurs spécifiques dans le réseau et leurs inter-mesures tels que : la distance, le décalage horaire d'arrivée, l'angle d'arrivée et la connectivité. Les capteurs avec les informations de localisation, apriori connus, sont appelés ancres ou références et leurs emplacements peut être obtenu en utilisant un système de positionnement global (GPS), ou bien en installant des points d'ancrage à des points avec des coordonnées connues. Dans les applications nécessitant un système de coordonnées global, ces ancres permettent de déterminer

l'emplacement du réseau de capteurs dans le système de coordonnées global. Dans les applications où un système de coordonnées local suffit, par exemple : dans les maisons intelligentes, les hôpitaux ou pour la gestion des stocks, dans la salle où la connaissance dans laquelle se trouve un capteur est suffisante, ces ancres définissent le système des coordonnées local auquel tous les autres capteurs sont visés. En raison des contraintes de coût, de taille des capteurs, de consommation d'énergie, d'environnement d'exécution (par exemple, dans certains milieux où le GPS n'est pas accessible) et de déploiement de capteurs (Peuvent être dispersés au hasard dans une région), la plupart des capteurs ne connaissent pas leurs propres endroits. Ces capteurs avec des informations de localisation inconnue sont appelés : les nœuds non ancre et leurs coordonnées doivent être estimées en utilisant un algorithme de localisation. Dans certaines autres applications, par exemple, pour le routage géographique dans un réseau de capteurs sans fil, où il n'y a pas de nœuds de référence et la connaissance de l'emplacement physique d'un capteur est inutile, les gens sont plus intéressés à savoir la position d'un capteur par rapport à d'autres capteurs. Dans ce cas, les algorithmes de localisation de capteur peuvent être utilisés pour estimer la position relative des capteurs à l'aide des mesures inter-capteurs.

Ce chapitre présente le contexte dans lequel s'inscrit la localisation dans les réseaux de capteurs sans fil. Pour cela nous commençons par la définition de la localisation, ensuite nous traitons les différentes techniques de localisation et de positionnement et finalement, la localisation dans les réseaux de capteurs statiques en illustrant ses méthodes de base et nous terminons par une conclusion.

I. Localisation

La localisation peut être déterminée comme la position d'un objet ou d'une personne dans un repère [29]. Un système de localisation doit avoir les propriétés suivantes :

- une technique d'estimation de position : trilatération ou triangulation, ou min-max qui seront définis dans la suite du chapitre ;
- un repère qui permet d'obtenir des positions et qui les organise de façon cohérente, et trois types de position sont observés :

    1) les positions absolues renseignent sur la position réelle de l'objet sur le globe terrestre : longitude et latitude.
    2) les positions relatives indiquent juste une direction par rapport à un voisinage donné (à droite au bout de la rue par exemple).
    3) les positions symboliques désignent par exemple une salle, un espace particulier.

- une précision de position : une position peut aller d'un point dans le cas d'une grande précision à une surface (ou volume) si la précision de position est moins importante.
- une architecture particulière : un système de positionnement en intérieur dans un bâtiment par exemple ne possède pas les mêmes contraintes qu'un système de localisation d'extérieur.

- un coût : matériel, infrastructure, ...

Pour estimer la position d'un objet, il faut des points dont la position est connue ou des points qui connaissent leur propre position ; points de référence ou ancres. Considérons qu'un nœud A désire estimer sa position. Son estimation de position dépendra du nombre et du type de position de nœuds de référence qu'il pourra trouver. La figure 3.1 détaille trois cas :

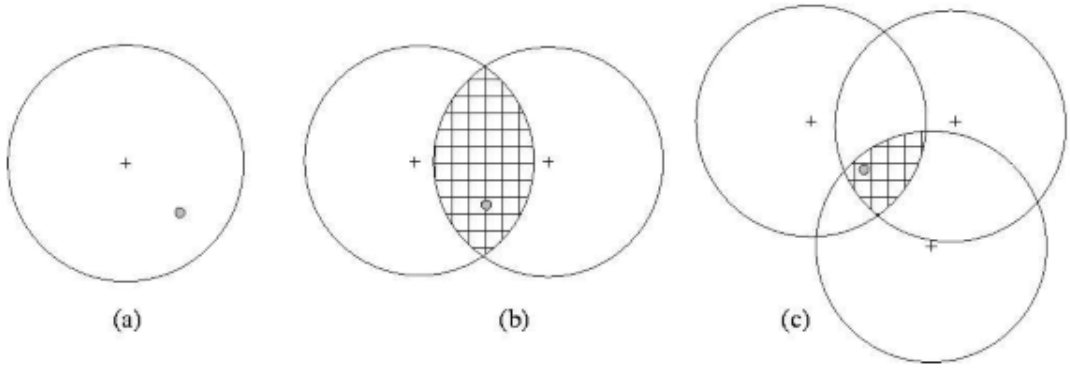


Fig. 3.1 : Position en fonction du nombre et du type de nœuds de référence [30].

Dans le cas 3.1(a), un seul nœud de référence est disponible. A aura une position de type symbolique -“ je suis dans telle pièce “- ou relative -“ je suis en bas à droite “- . La position de A est une position de proximité car seule une indication d'appartenance à une zone donnée peut être obtenue dans un tel contexte.

Si A utilise deux points de références -3.1(b)- alors son positionnement est plus précis qu'avec un seul nœud : il possède plus d'informations pour estimer sa position. Dans le cas où les nœuds de référence ont une très bonne précision de localisation, alors A est l'un des points d'intersection des deux cercles, cercle de couverture radio par exemple. Dans le cas contraire, A appartient à la *zone de recouvrement* des deux points de référence (zone commune aux deux nœuds de référence).
La position de A est d'autant mieux estimée que le nombre d'ancres augmente. Dans le cas 3.1(c), la position de A est soit un point - point d'intersection des trois cercles -, soit une aire de recouvrement limitée [30].

II. Système de localisation et de positionnement

La localisation c’est l’estimation des différentes distances séparant un objet des autres objets qu’il l’entour, par contre le positionnement donne les coordonnées relative à ce dernier.

III.1 Techniques de localisation

III.1.1 Timing

La distance entre les nœuds peut être évaluée à partir du temps de propagation d'un signal ou d'un paquet. Deux approches principales peuvent être définies pour ces méthodes : l'heure d'arrivée (TOA) et la différence de temps d'arrivé (TDOA).

a) Heure d'arrivée – Time of arrival (TOA)

La méthode de mesure du TOA estime le temps de vol (Time of Flight : TOF) du signal entre le nœud mobile (NM) et les nœuds de références (NR) pour estimer, par la suite, la distance qui sépare le NM des NR. Cependant, afin d'éviter toute ambiguïté sur le calcul du TOF, deux solutions peuvent être envisagées. La première, dont le principe général est illustré dans la figure 3.2(a), consiste en la synchronisation, du NM et des NR (les nœuds doivent avoir la même horloge). Dans cette situation, l'onde effectue un aller simple entre l'émetteur et le récepteur comme le montre la figure 3.2. (a) [31]. Dans ce cas, la distance, qui sépare le $i^{ème}$ NR du NM, s'écrit comme suit :

$$d_i = (t_i - T_0)c$$

Où t_i est l'instant d'arrivée de l'onde du $i^{ème}$ NR au niveau du NM, T_0 est l'instant d'émission du signal et c représente la vitesse de propagation.

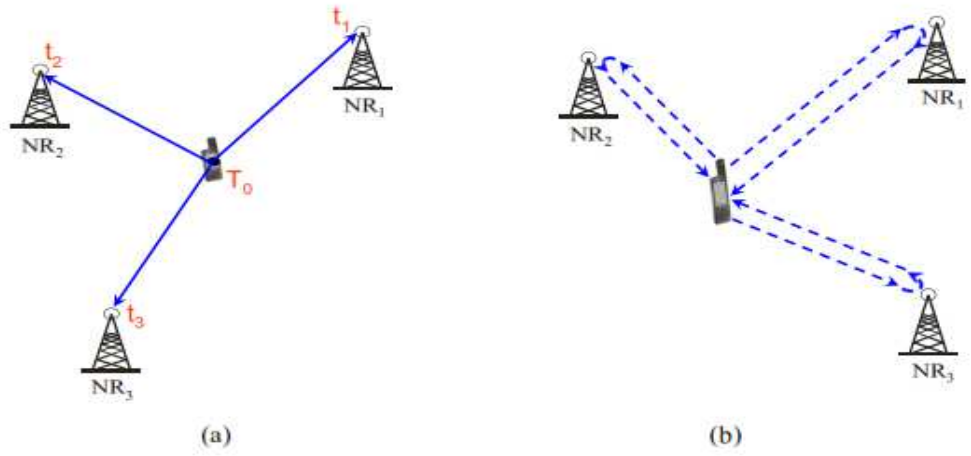


Fig. 3.2 : Mesure du TOA. (a) : Système TOA synchronisé, (b) : Système TOA non synchronisé.

La deuxième solution, qui peut être adoptée pour éviter le problème de l'ambiguïté, comme indiqué dans la figure 3.2(b) [32]. Cette dernière montre clairement que l'onde effectue un aller et un retour pour mesurer le temps de propagation entre le NM et le NR. Pin-Point's Local Positioning System (LPS) [33], les radars civils et militaires ou la proposition GPS-free [34] utilisent cette approche. Dans ce cas, les distances s'écrivent comme suit [32] :

$$d_i = \left(\frac{1}{2} (t_i - T_0) - t_{R,i} \right) c$$

Où $t_{R,\,i}$ est temps mis par le $i^{ème}$ NR pour répondre à la requête de l'émetteur.

b) Différence des temps d'arrivée – Time Difference of Arrival (TDOA)

Une deuxième approche consiste à évaluer la différence des temps d'arrivée de deux différents signaux. Ces signaux peuvent provenir de deux nœuds de référence distincts (Figure.3.3 (a)), ou peuvent être de natures différentes (Figure.3.3 (b)), comme les *ultrasons* et les signaux, *radio* qui peuvent être émis par une même source [30].

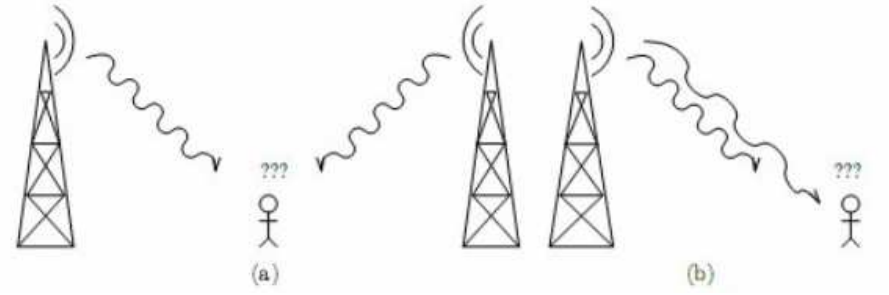


Fig. 3.3 : Temps différentiel d'arrivée [30].

Cette technologie s'applique dans les cas suivants :

- un récepteur reçoit des signaux d'une même nature d'au moins trois émetteurs ;
- un émetteur envoie un signal reçu par au moins trois récepteurs (dans ce dernier cas une vue globale des signaux sera connue).

Dans chacun des cas, les récepteurs mettent en corrélation leurs informations et en déduisent les distances qui les séparent des émetteurs. Il s'agit d'une simple résolution d'un système d'équations dont les distances sont les inconnues.

A partir des temps de propagation $t_{prop,}$ on détermine la distance d pour ensuite déterminer la position du capteur avec la relation suivante [35] :

$$t_{prop} = \frac{d}{c}$$

Où d est la distance d'un capteur i, c est la vitesse de propagation.

III.1.2 Direction – Angle d'arrivé (AOA)

Cette technique permet d'estimer la position d'une cible d'après les coordonnées des stations de base. Son fonctionnement est basé sur le calcul de l'angle du signal émis par un terminal lorsqu'il arrive à la station de base. Comme on peut le constater sur la figure 3.4, cette méthode nécessite deux stations de base au minimum afin de calculer la position du terminal mais pour plus de précision d'avantage de stations de base peuvent être utilisées. Pour le calcul de l'angle d'arrivée des signaux, chaque station de base doit être équipée d'un réseau d'antennes. Les angles α_1 et α_2 d'un signal émis par le terminal vers les stations de base sont calculés et la position de la cible peut être découverte par triangulation (voir la prochaine section) [36].

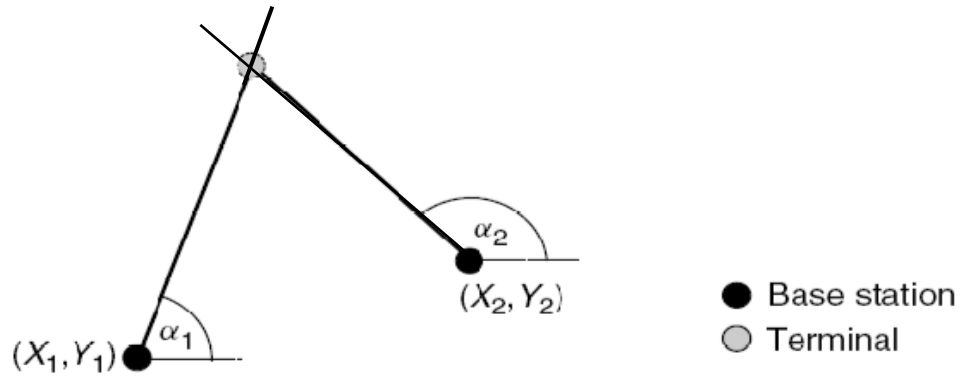


Fig. 3.4 : Angle of Arrival (AOA) [36].

III.1.3 Puissance du signal RSSI

Une autre manière d'estimer une position est de baser le calcul sur la caractéristique physique du canal radio [30]. La puissance d'émission et de réception d'un signal peut être également exploitée pour obtenir la distance entre deux capteurs. Trois capteurs de repère au minimum sont nécessaires pour déterminer la position en 2D. La technologie RSSI (Received Signal Strength Indicator) considère la perte de puissance d'un signal entre son émission et sa réception. Cette perte varie en fonction de la distance entre les deux capteurs : plus les capteurs sont éloignés (resp. proches), plus la perte est importante (resp. faible). Cette perte sera alors traduite en une distance [37].

La figure suivante illustre la détermination de la distance séparant un capteur de trois voisins en utilisant la méthode RSSI.

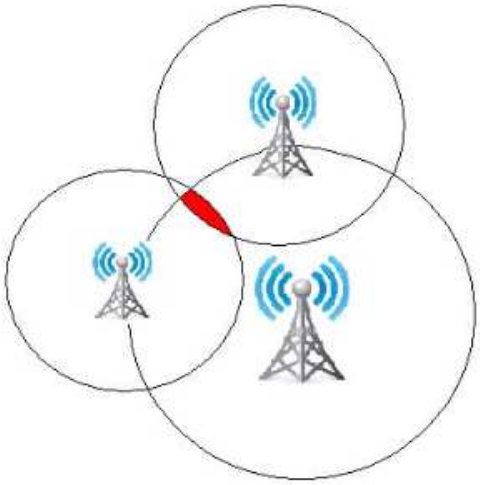


Fig. 3.5 : Principe de l'utilisation des RSSI [37].

III.2 Techniques de positionnement

Dans cette section, nous allons rappeler les trois techniques d'estimation de position utilisées par les systèmes de positionnement : triangulation, trilatération et Min-Max

III.2.1 Triangulation

La triangulation [38] est un algorithme utilisant les propriétés géométriques du triangle afin de pouvoir obtenir une position d'un nœud à partir de deux nœuds de référence. L'algorithme de triangulation nécessite pour estimer la position de l'objet mobile, les angles obtenus par la méthode Angle of Arrival (AOA) d'au moins deux sources. Pour deux stations de base Rx_1 et Rx_2, les angles d'incidence des trajets provenant de l'objet mobile, donnés respectivement par a et $ß$, sont représentés en 2D sur la Figure 3.6 :

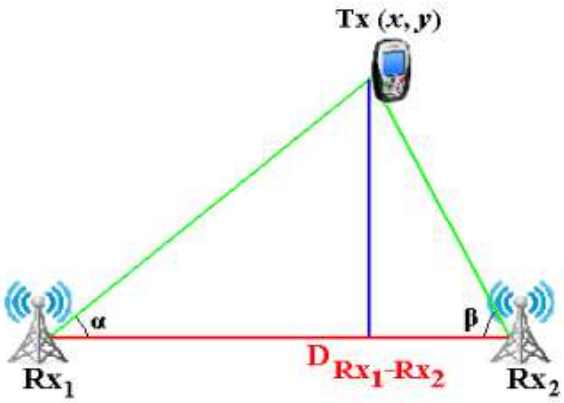


Fig. 3.6 : Principe de Triangulation.

Par construction géométrique, les coordonnées (x, y) de l'objet mobile Tx sont données par [39]:

$$\begin{cases} x = \dfrac{\tan(\beta)}{\tan(\alpha)+\tan(\beta)} D_{Rx_1\text{-}Rx_2} \\ y = \dfrac{\tan(\alpha)\tan(\beta)}{\tan(\alpha)+\tan(\beta)} D_{Rx_1\text{-}Rx_2} \end{cases}$$

III.2.2 Trilatération

La trilatération est une méthode permettant de déterminer une position relative du mobile en utilisant la géométrie des triangles d'une manière similaire à la triangulation [38]. Le procédé implique la connaissance de la distance de l'objet mobile par rapport à un ensemble de références dont les positions sont connues, cette distance peut être obtenue par deux méthodes : l'une à base de temps d'arrivé (TOA) et l'autre à base de mesure de puissance du signal (RSSI).
Les coordonnées (*x, y*) de l'objet mobile (*Tx)* exprimées en fonction des distances et des coordonnées connues des points de réception, et sont données par [39]:

$$\begin{cases} x = \dfrac{x_2^2 + d_1^2 - d_2^2}{2x_2} \\ y = \dfrac{x_3^2 + y_3^2 + d_1^2 - d_3^2 - 2xx_3}{2y_3} \end{cases}$$

Le procédé est illustré dans la Figure 3.7 ci-dessous :

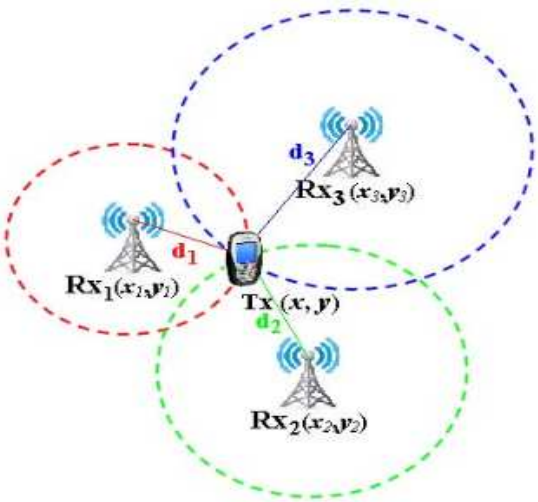


Fig. 3.7 : Principe de Trilatération.

III.2.3 Min-Max

Min-Max [40] est une méthode simplifiée pour déterminer la position et basée sur les distances connues et la localisation des autres nœuds, cette méthode ressemble beaucoup à la trilatération, mais au lieu de cercles, utilise des rectangles pour définir la distance d'un nœud.

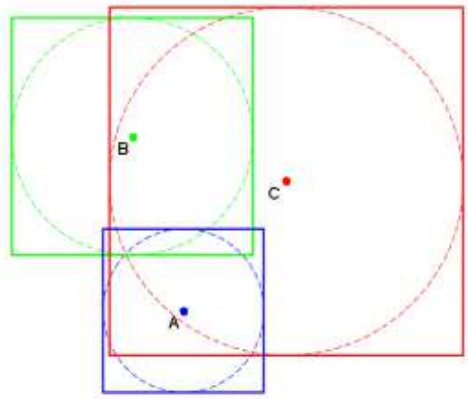


Fig. 3.8 : Exemple de la méthode Min-Max [40].

On essaie de trouver les coordonnées du nœud à positionner en utilisant la méthode min-max : d'abord, on trace un carré autour de chaque cercle, (voir la figure 3.8). Ensuite le chevauchement de tous les carrés est le domaine dans lequel se trouve le nœud à positionner. Parce que on ne sait pas où le nœud est placé dans la zone, on prend le milieu du rectangle formé par l'intersection des carrés comme une position finale, cette position est démontrée dans la figure 3.9.

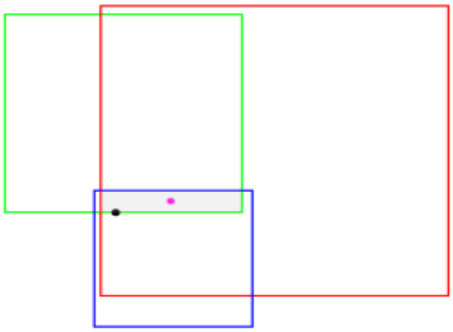


Fig. 3.9 : Résultat final de la méthode Min-Max [40].

L'avantage de min-max par rapport à la trilatération est qu'on peut estimer la position des nœuds en utilisant seulement deux nœuds. Un autre avantage est la simplicité de l'algorithme. Car nous avons besoin de calculer l'addition et la soustraction des rectangles.

Avec la trilatération, le nœud est toujours sur le cercle, mais avec la méthode min-max, le nœud n'est pas nécessairement sur le rectangle. Cependant, le nœud sera près du rectangle et non du centre [40].

IV. Localisation dans les RCSFs

La connaissance des positions des capteurs dans l'environnement est souvent souhaitable, afin de pouvoir déterminer l'origine des flux de mesures collectées.

Une méthode de localisation dans les réseaux de capteurs est composée de deux phases :

IV.1 Estimation des distances (offline)

Dans cette phase les nœuds communiquent entre eux et collectent différents indicateurs de qualité des communications radio. Le hardware radio peut rapporter diverses informations sur le signal radio entre deux nœuds.

En effet le simple fait qu'ils communiquent entre eux, nous indique qu'ils sont à portée radio l'un de l'autre. De plus le hardware de nos nœuds peut nous rapporter diverses caractéristiques à propos du signal entre les deux nœuds, à partir desquelles les distances séparant les nœuds peuvent être estimées [41] [42].

IV.2 Dérivation des positions (online)

Le but de cette phase est de trouver les positions des nœuds qui respectent au mieux les distances inter-nœuds estimées. Si nous connaissons la position de quelques nœuds du réseau dans un certain système de coordonnées, les positions des autres nœuds dans ce système de coordonnées peuvent être trouvées.

Notons que divers raffinements peuvent être appliqués, par exemple : les distances estimées entre les nœuds peuvent être raffinées en refaisant communiquer les nœuds. De la même façon, on peut raffiner la position des nœuds et appliquer diverses vérifications sur leurs positions [41].

IV.3 Techniques d'estimation de positions

IV.3.1 Technique d'empreinte (mapping)

La technique mapping utilise une base de données qui se compose des paramètres du signal estimé auparavant à des positions connues, pour estimer la position du nœud cible.

Généralement, la base de données est obtenue dans la phase calibrage (offline) avant le positionnement temps réel.

L'idée principale derrière l'estimation de position par les techniques mapping est de déterminer une fonction de régression sur un ensemble de données, puis d'estimer la position d'un nœud donné selon cette fonction :

$$T = \{(m_1, l_1), (m_2, l_2), .. \dots \dots \dots (m_{NT}, l_{NT})\}.$$

Où l_i est le vecteur des positions i, m_i représente le vecteur des paramètres estimés pour la position i et NT est le nombre total d'éléments dans l'ensemble de données (c-à-d la taille de la base de données).

Dans l'ensemble donné par la formule précédente, une technique mapping détermine la règle d'estimation de position, et estime alors la position l d'un nœud cible donné en se basant sur un vecteur de paramètres m lié a ce nœud. Un certain nombre des techniques mapping en estimation de position utilise l'estimation de k-proche voisin (k-NN), le vecteur de régression de soutien (SVR) et les réseaux de neurones [45] [46].

IV.3.2 Technique géométrique

Les technologies géométriques résolvent la position du nœud cible comme intersection des lignes de position obtenues à partir d'un ensemble de paramètres relatifs à un certains nombre de nœuds de référence. RSSI ou TOA définit des cercles pour la position du nœud cible par la trilatération. D'autre part, AOA définit une ligne droite passant par le nœud cible et le nœud de référence. Par conséquent, deux paramètres de références d'AOA sont suffisant pour localiser le nœud cible par la triangulation. Dans le cas du positionnement basé sur TDOA, chaque paramètre de TDOA détermine une hyperbole, pour la position du nœud cible. Pour trois nœuds de références, deux niveaux différents (obtenus à partir des paramètres de TDOA) définissant deux hyperbole, l'intersection rapporte la position du nœud cible (figure 3.10). Cependant, la position ne peut pas être déterminée toujours uniquement selon le traitement géométrique des nœuds [47] [28].

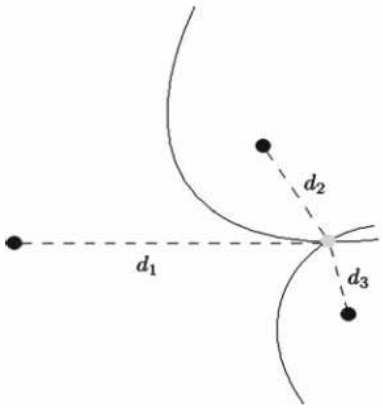


Fig. 3.10 : Positionnement par TDOA

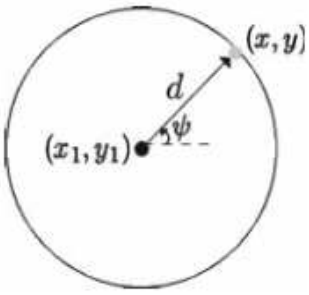


Fig. 3.11 : Positionnement par TOA/AOA

Les techniques géométriques peuvent également être appliquées aux systèmes hybrides, dans lesquels les types multiples de position sont liés aux paramètres, tels que TDOA/AOA [49] ou TOA/TDOA [50], sont utilisée en estimation de position.

Par exemple, pour un système de l'hybride AOA/TOA comme représenté sur la figure 3.11, le nœud de référence peut estimer l'AOA et le TOA à partir du signal du nœud cible, et la position peut être calculée comme suit :

$$x = x_1 + d \cos \psi$$
$$y = y1 + d \sin \psi$$

Où (x_1, y_1) est la position du nœud de référence, ψ est l'angle calculé par la méthode AOA, et d est le niveau obtenu par l'estimation TOA. En d'autres termes, le nombre minimum de nœuds qui sont exigés pour déterminer la position du nœud de cible peut changer selon les nœuds cible et/ou des nœuds de référence [51].

IV.3.2 Technique statistique

À la différence des techniques géométriques, l'approche statistique présente un cadre théorique pour l'estimation de position dans la présence de position multiple reliée au paramètre d'estimation avec ou sans le bruit [51].

V. Caractéristiques des méthodes

V.1 La nécessité de connaitre la position d'ancre

Si une méthode requiert l'encodage au préalable de la position d'un certain nombre d'ancres, cela signifie qu'il faudra qu'une personne intervienne avant un déploiement pour mesurer la position d'un certain nombre de nœuds. Cela est parfois difficile, voire impossible dans certaines situations. Et donc, le fait que la méthode de localisation requiert ou non de connaître la position d'un certain nombre d'ancres est une caractéristique importante de la méthode [52].

V.2 Forme d'implémentation

Il existe deux méthodes pour implémenter le processus de localisation :

V.2.1 Méthodes centralisées

Tous les nœuds communiquent avec leurs voisins et renvoient à l'ordinateur central soit des informations sur le signal, soit directement les distances. L'ordinateur central s'occupe si nécessaire d'estimer la distance à partir des informations sur le signal et ensuite de localiser les nœuds [53] [54].

V.2.2 Méthodes distribuées

Dans cette méthode, tous les nœuds communiquent avec leurs voisins pour estimer les distances et échangent leurs informations de voisinage. Ils dérivent ensuite de façon distribuée la position de tous les nœuds dans le réseau. C'est-à-dire qu'à la fin du processus de localisation, chaque nœud doit connaître sa position ainsi que celles de ses voisins et ce sans l'aide d'un ordinateur central qui effectuerait les calculs. Pour les grands réseaux, on considère qu'une méthode distribuée est nécessaire car les méthodes centralisées demanderaient trop de communication pour l'acheminement des informations vers l'unité centrale et consommeraient donc trop d'énergie [56] [57].

VI. Localisation dans les réseaux de capteurs statiques

VI.1 Méthodes libres de mesure

Dans cette famille de méthodes, les capteurs cherchant à déterminer leurs positions s'appuient uniquement sur les positions des ancres. Aucune mesure de distance ou d'angle n'est utilisée. Par conséquent, ces méthodes ne peuvent fournir que des positions estimées aux capteurs, Il existe deux techniques courantes dans ce type de méthodes :

a) La première consiste à définir des zones contenant les capteurs dont les centres de gravité correspondent à leurs positions estimées.

b) Dans la seconde technique, chaque capteur estime les distances qui le séparent des ancres et applique la trilatération pour calculer sa position estimée.

VI.1.1 HTRefine (Hot Top Refine)

Dans la méthode HTRefine, toutes les ancres diffusent leurs positions. Lorsqu'un capteur reçoit la position d'une ancre, il estime la distance qui le sépare d'elle. Pour ce faire, HTRefine utilise la technique d'estimation des distances DV-Hop (Distance Vector-Hop) [58].

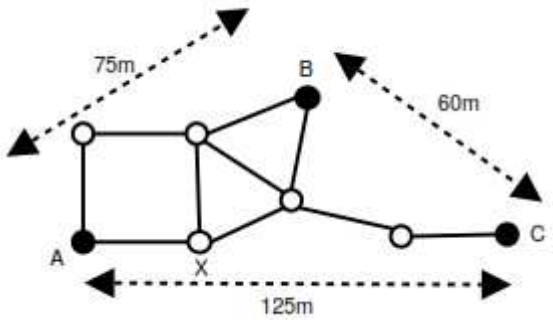


Fig. 3.12 : DV-Hop.

Dans cette technique, lors de l'inondation des positions des ancres, chaque capteur calcule le nombre de sauts minimum qui le sépare de chacune des ancres. Une deuxième vague

d'inondation fournit suffisamment d'informations au capteur pour qu'il puisse convertir ces nombres de sauts en estimations de distances.

La figure 3.12 est une illustration de DV-Hop où l'ancre A estime la distance moyenne d'un saut. Les nœuds noirs représentent les ancres et les nœuds blancs les capteurs non localisés. La distance moyenne d'un saut est donnée par la fraction : [(125+75)/(3+4)=28.57 mètre]. Le capteur X estimera les distances avec B et C comme suit : d_{XB} = 2*28.57 et d_{XC} = 3*28.57.
Pour obtenir leurs positions, les capteurs utilisent ensuite la multilatération. En reprenant l'exemple de la figure 3.12, X obtiendra sa position en résolvant le système suivant :

$$\begin{cases} d_{AX}^2=(x_X-x_A)^2+(y_X-y_A)^2 \\ d_{BX}^2=(x_X-x_B)^2+(y_X-y_B)^2 \\ d_{CX}^2=(x_X-x_C)^2+(y_X-y_C)^2 \end{cases}$$

Après avoir estimé leurs positions, les capteurs les diffusent à leurs voisins. En fonction de ces données et grâce aux relations de voisinage, les capteurs calculent à nouveau leurs positions qui se rapprochent de leurs positions réelles. Après un nombre d'itérations défini, les capteurs fixent leurs positions estimées [55].

VI.2 Méthodes basées mesure

Les méthodes basées mesures utilisent les technologies TOA, RSSI, AOA et autres, afin de mesurer les distances ou les angles entre deux capteurs voisins. Grâce à cette capacité de mesure, un capteur pourra, sous certaines conditions, obtenir sa position exacte. Autrement, une position estimée lui sera attribuée. Les méthodes basées mesures sont les plus répandues.

Les méthodes APS et APS_{AOA} présentées ci-après considèrent la première pour des capteurs ayant la capacité de mesurer des distances et la deuxième celle de mesurer des angles.

VI.2.1 APS

Dans APS et comme précédemment, les ancres commencent par diffuser leurs positions et les capteurs estiment les distances qui les séparent de ces ancres. La technique d'estimation des distances utilisée dans APS est nommée Euclidean [43] [58].

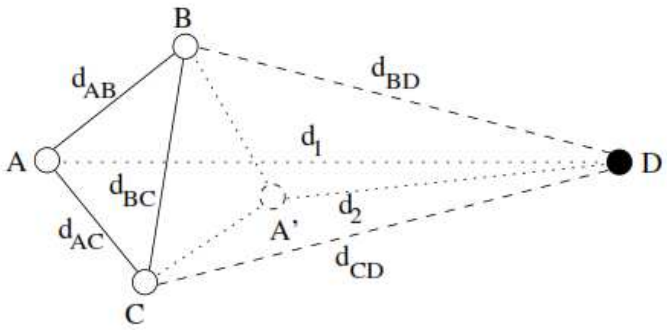


Fig. 3.13 : Euclidean [43].

Cette technique calcule, sous certaines conditions, une distance exacte lorsqu'un capteur connaît deux voisins ayant estimé leurs distances avec une ancre. Le principe de cette méthode est illustré en figure 3.13. Soient *A*, *B*, *C* étant des capteurs et *D* une ancre. *A* cherche à calculer sa distance avec *D*. *B* et *C* sont voisins de *A* et voisins entre eux. *A* et *B* ont calculé leurs distances avec *D* (d_{BD}, d_{CD}). *A* connaît alors les distances (d_{AB}, d_{AC}, d_{BC}, d_{BD}, d_{CD}). Par conséquent, il connaît tous les côtés et une diagonale du quadrilatère *ABCD*. La seconde diagonale correspond à la distance d_{AD}. Mais deux distances sont possibles (d_1 et d_2). Un processus de vote des voisins et un autre dit du voisin commun permet à *A* de déduire la distance qui le sépare de *D*. En fait, ces processus font intervenir un troisième capteur ayant lui aussi estimé sa distance avec *D*. Selon le cas où ce capteur est voisin de *B* ou *C*, ou bien voisin des deux, ces processus, par de simples relations géométriques, déduisent la distance d_{AD}.

Une fois les distances avec les ancres obtenues, les capteurs appliquent alors la multilatération décrite ci-avant. Dans APS, aucune phase de raffinement n'est proposée puisque les contraintes imposées par le voisinage et par les distances sont prises en compte lors de l'estimation des distances [43] [58].

VI.2.2 APS_{AOA}

APS_{AOA} est une extension de la méthode précédente qui considère des capteurs ayant, non pas la capacité de mesurer les distances avec leurs voisins, mais celle de mesurer les angles [44].

Au départ, les ancres diffusent leurs positions et les capteurs voisins cherchent à déduire les angles qu'ils forment avec chacune d'elles par rapport à leurs propres axes de référence. Ces capteurs font alors suivre leurs angles à leurs voisins pour qu'ils puissent déduire leurs angles à leurs tours et ainsi de suite.

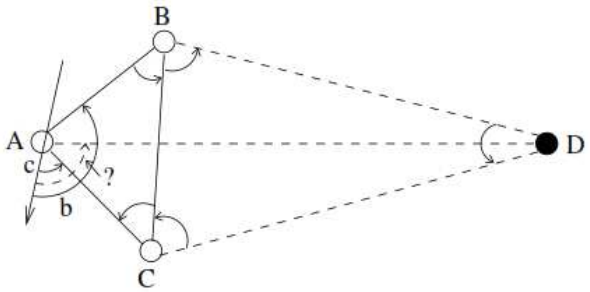


Fig. 3.14 : APS avec AOA [44].

La figure 3.14 présente comment un capteur déduit l'angle avec une ancre distante. Le capteur *A* cherche à déduire son angle avec l'ancre *D* par rapport à son axe de référence. Les capteurs *B* et *C*, voisins de *A* et voisins entre eux, connaissent les angles respectifs qu'ils forment avec *D*. Ainsi les angles des triangles *ABC* et *BCD* sont connus et l'angle *DAC* peut être calculé. L'angle par rapport à l'axe de *A* sera donc égal à c + *DAC*.

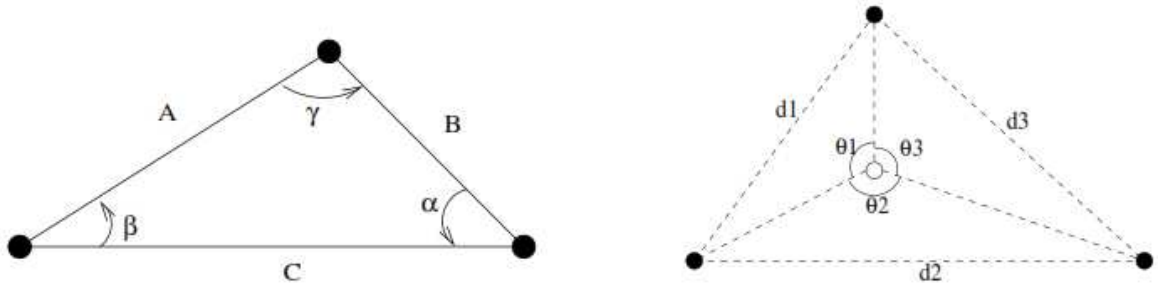


Fig. 3.15 : [44] a) La triangulation b) Calcul de la position

Dès lors qu'un capteur a obtenu les angles qu'il forme avec au moins trois ancres, il obtient sa position grâce à la triangulation qui, en considérant la figure 3.15(a), donne le système suivant [44] :

$$\begin{cases} A^2 = B^2 + C^2 - 2BC\cos\alpha \\ B^2 = A^2 + C^2 - 2AC\cos\beta \\ C^2 = A^2 + B^2 - 2AB\cos\gamma \end{cases}$$

Ainsi, la triangulation sera utilisée lorsqu'un capteur cherchant sa position possède au moins trois ancres dans son voisinage, comme illustré dans la figure 3.15(b), en connaissant les distances d_1, d_2, d_3 entre les ancres et les angles θ_1, θ_2, θ_3, le capteur peut calculer sa position en résolvant le système d'équations ci-dessus [16].

VII. Conclusion

La précision des mesures réalisées par ces technologies varie selon plusieurs paramètres liés à l'environnement du réseau : obstacles, degré d'humidité, vitesse du vent, ... Par exemple, les technologies RSSI ou AOA sont fortement déconseillées dans des milieux fermés. Des études ont analysé l'impact de ces paramètres sur les mesures de distances et d'angles et ont montré que les mesures de distances étaient moins perturbées que celles des angles. Si certaines méthodes de localisation s'interdisent d'utiliser ces technologies à cause de leurs erreurs de mesure, il est indispensable, pour celles qui les utilisent, de considérer ces erreurs lors du calcul des positions des capteurs.

Dans le chapitre suivant nous allons utiliser la technique RSSI pour la localisation d'un capteur mobile, et la technique trilatération pour le positionnement de ce capteur.

Chapitre 4

Simulation et implémentation

I. Introduction

Dans les environnements d'intelligence ambiante les éléments de la vie quotidienne sont équipés de petits dispositifs interconnectés permettant de rendre différents services allant de la prévention (incendies, accidents) à l'assistance (guidage, contrôle à distance) en passant par le confort. Pour réaliser des applications fournissant ces services, on doit prendre en compte plusieurs paramètres tels que la qualité de service et souvent le temps réel.

Parmi les applications les plus répondues et fondamentales dans les environnements d'intelligence ambiante et à laquelle on s'intéresse dans notre travail est la localisation indoor dans les réseaux de capteurs sans fils. En effet la résolution du problème de la localisation de mobile en intérieur nécessite le déploiement d'un ensemble de capteurs fournissant chacun une part d'information, l'agrégation de ces informations permet l'estimation de la position du mobile en temps réel.

Dans le but de traiter ce problème de localisation, dans notre travail, on se base sur la technique RSSI pour le calcul des distances et ensuite sur la technique géométrique multi-latération afin d'estimer la position du mobile. Sachant que la multi-latération donne plus de précision d'autant qu'on a plus de cercles, c'est-à-dire plus de distances ce qui sous-entend de recevoir le maximum de signaux possibles qui proviennent des différents capteurs dans un laps de temps Δt. Connaissant le fonctionnement des protocoles des réseaux de capteurs sans fils, concernant l'envoi et la réception de messages tel que CSMA/CA qui gère aléatoirement la (re)transmission des messages, ce compromis s'avère être difficile à atteindre.

Dans ce chapitre il est question de présenter notre proposition qui consiste à rendre le protocole d'accès au support CSMA/CA sous Contiki plus adapté pour une application de localisation en temps réel. Dans un premier temps, nous allons donner l'objectif de notre travail et détailler la problématique à l'origine de ce dernier. Dans un second temps, nous allons revenir sur le système d'exploitation Contiki - introduit au chapitre 1 - plus précisément sur la pile Rime et la couche MAC afin de présenter les modules qui sont exploités par l'application. Nous décrivons également le protocole CSMA/CA tel qu'il est implémenté sous Contiki 2.6. Nous détaillons ensuite un scénario du fonctionnement de l'application à réaliser qui peut être décomposée en deux partie : 1) la première partie concerne la programmation des différents capteurs contribuant aux échanges de messages permettant le calcul des distances, 2) la deuxième partie, qui se déroule non pas sur les capteurs mais plutôt sur le PC qui exploite les résultats de la première pour estimer la position du mobile. En outre, nous présentons trois propositions pour remédier aux problèmes soulignés dans la problématique. Tout au long de chapitre nous effectueront diverses simulations, sous le simulateur Cooja, portant sur chacune des propositions en vue de choisir la solution la plus efficace qui sera implémentée sur des capteurs réels. Nous allons conclure ce chapitre par une démonstration de l'application de localisation indoor.

II. Objectif de notre de travail

Notre travail consiste à réaliser une application de localisation indoor dans les réseaux de capteur sans fils. Pour cela on déploie un ensemble de capteurs dans un environnement où se déplace un mobile qui désire connaître sa position à chaque instant *t*. Nous allons employer la technique RSSI qui ne nécessite pas d'autres matériels autres que les capteurs de localisation pour calculer les distances et la technique géométrique multi-latération pour estimer la position de mobile dans cet environnement.

Pour avoir les coordonnées de la position à chaque instant *t* et avec une précision adéquate (puisque elle dépend du domaine d'application), notre application doit satisfaire plusieurs contraintes relatives aux systèmes temps réels.

II.1 Quelques contraintes des systèmes temps réel

Un système est dit temps réel lorsque ce système est capable de contrôler (ou piloter) un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé. Ce système se différencie des autres systèmes informatiques par la prise en compte de contraintes temporelles dont le respect est aussi important que l'exactitude du résultat, autrement dit le système ne doit pas simplement délivrer des résultats exacts, il doit les délivrer dans des délais imposés.

Comme conséquence de cette définition, les applications temps réels :

1. Sont des systèmes en interaction avec l'environnement physique ;
2. Doivent satisfaire les contraintes de temps : la date de livraison d'un résultat ;
3. Sont des systèmes dédiés (non généralistes) : matériel (capteur, actionneur, …) ou logiciel ;
4. Respectent des contraintes industrielles : poids, taille, consommation de l'énergie, coût, …, par capteurs miniaturisés enfouis dans des objets électroménagers à l'intérieur de la maison.

II.2 Comment prendre en compte ces contraintes dans l'application

Les contraintes 1, 3 et 4 sont satisfaites vu que notre application dépend de l'environnement et les objets qui se trouvent dedans, elle est dédiée à la localisation et nous utilisons des capteurs qu'on peut placer n'importe où. Par contre, il nous reste la contrainte temporelle et comme pour tous les autres systèmes la précision.

- Le mobile voudra avoir sa position à chaque intervalle Δt de temps, c'est-à-dire, le temps de la réception des signaux RSSI est borné dans le temps et ne peut dépasser Δt. Cette contrainte (temporelle) doit être prise en compte dans la programmation des capteurs.
- La technique multi-latération, décrite dans le chapitre 3, donne souvent plus de précision lorsque le nombre de nœuds de références est plus important (et on reçoit des signaux de ces nœuds).

Nous avons choisi le système d'exploitation Contiki, pour les raisons invoquées dans le chapitre 1, pour les réseaux de capteurs sans fils afin de programmer et de faire communiquer les capteurs. Avant de se lancer dans la manipulation des capteurs réels, nous avons procédé à un ensemble de simulation sous Cooja. Pourquoi ? Pour deux raisons principales : à force de manipuler les capteurs et de répéter cette manœuvre ces derniers se détériorent, en plus de cela pour éviter le gaspillage de l'énergie ce qui est connue dans les WSN.

Les résultats préliminaires obtenus des simulations étaient loin de répondre aux contraintes des applications en temps réel en général et au domaine de la localisation en particulier. Nous allons analyser ces résultats et expliquer en quoi ne sont pas adéquat à notre application dans la partie problématique.

III. Problématique

Afin de mieux cerner les éléments de la problématique, nous introduisons la figure 4.1 qui représente une séquence de messages reçus par le mobile lors d'un test.

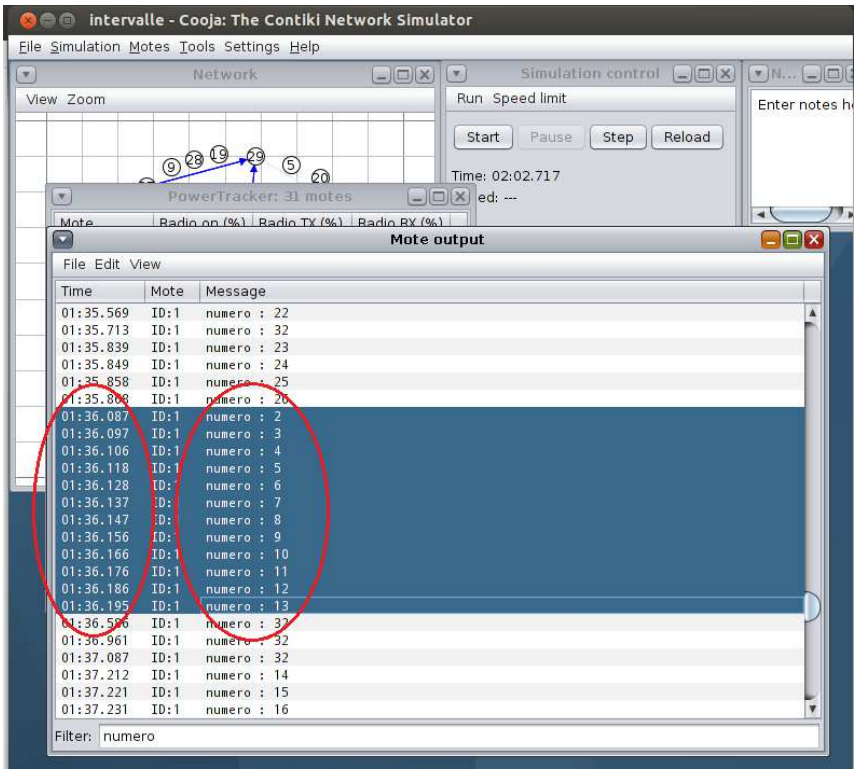


Fig. 4.1 : La séquence de messages reçus par le mobile

Nous voyons à partir de la figure qu'à un instant avancé de la simulation (après 1mn 36 s), le mobile a reçu une suite de messages (numéro : 2, 3, 4,…, 13) qu'il aurait dû recevoir bien avant. Ces messages retardataires engendrent plusieurs anomalies pour les systèmes temps réel dont fait partie la localisation indoor :

1. L'information qu'ils contiennent n'est plus appropriée à l'instant présent (obsolète) ;
2. Souvent le mobile reçoit successivement une liste de messages en retard de même capteur ancre, par conséquent, tous les RSSI reçus ont la même valeur et conduisent à estimer une même distance par rapport à un seul capteur ancre ce qui est un inconvénient pour la précision de la localisation (manque de précision);
3. Le capteur procède à la transmission de tous les paquets en retard dans sa file d'attente, il monopolisera le canal empêchant ainsi les autres capteurs de transmettre vers le mobile les informations utiles;

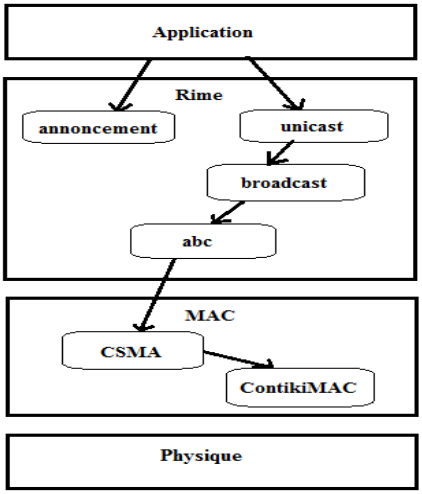
4. La transmission d'un message est toujours consommatrice d'énergie, alors transmettre des messages inutiles est un gaspillage d'énergie.

Dans le but de comprendre les origines de ces anomalies, nous allons d'abord étudier le système Contiki en se focalisant sur la couche réseau afin de retracer les différentes étapes sur les quelles passe un paquet depuis sa génération jusqu'à sa transmission effective, une de ces étapes importantes est l'algorithme d'accès au support CSMA qui aussi décortiqué.

IV. Contiki un peu plus en détails

Le système d'exploitation Contiki bénéficie de deux piles de protocoles pour communiquer sur le réseau, la pile uIP offre un ensemble plus que suffisant des protocoles de la suite TCP/IP pour qu'un capteur qui exécute Contiki soit capable d'obtenir une configuration IP et d'échanger des paquets de données sur internet. Mais la couche qui nous intéresse dans ce travail est la couche Rime, Rime est une pile légère de transmission conçue pour les radios de basse puissance. Rime est le résultat de la superposition de plusieurs primitives, chacune d'entre elles offre un service à la couche supérieur jusqu'à arriver à la couche application. Parmi ces primitives, celles qui nous intéressent sont:

- **Anonymous best-effort local area broadcast ou abc** : Le module *abc* envoie des paquets anonymement (sans ajouter l'adresse de l'émetteur) à tous les voisins locaux ;
- **Best-effort local area broadcast**: le module *broadcast* ajoute au paquet d'adresse Rime de l'émetteur et fait appel ensuite aux services du module abc ;
- **Single-hop unicast** : le module unicast ajoute au paquet l'adresse Rime du récepteur et fait appel ensuite aux services du module *broadcast* ;
- **Announcements** : le module d'annonce est utilisé par les capteurs pour envoyer des annonces aux voisins afin de signaler leur présence. Les annonces consistent en de petites valeurs sur 16 bits.



La pile Rime utilise des canaux représentés par un nombre entier de 16 bits qui doit être le même chez l'émetteur et le récepteur pour ouvrir une connexion. Lors de l'ouverture des connexions (car entre deux capteurs on peut ouvrir plusieurs connexion) le système Contiki fait appel aux mécanismes de *callback* (appel de retour), en fait, les primitives supérieurs passent aux modules inférieurs des pointeurs vers des fonctions qui seront appelées en retour par ces dernier, par exemple lors de la réception d'un paquet, cela permet de remonter aux couches supérieurs. Enfin, Un capteur souhaitant utiliser la couche Rime doit impérativement posséder une adresse Rime qui consiste en un simple nombre.

Une fois le paquet arrivé au module abc, ce dernier veille à le passer à la couche MAC. Dans la couche MAC le paquet va être traité par deux primitives, dans un premier temps il doit passer par l'algorithme CSMA puis par ContikiMAC, nous expliquons tout de suite ContikiMAC, CSMA sera décrit en détails dans la section suivante.

La couche MAC se situe juste en dessous de la couche Rime (voir la Figure 4.2), elle consiste en un ensemble de protocoles de conservation d'énergie comme ContikiMAC et d'accès au support comme CSMA. Les appareils sans fil de basse puissance doivent garder leurs émetteurs récepteurs par radio hors fonction autant que possible pour atteindre une faible consommation d'énergie, mais ils doivent se réveiller assez souvent pour pouvoir recevoir la transmission de leurs voisins. ContikiMAC est un protocole qui permet aux nœuds de participer à la communication réseau tout en maintenant leurs radios arrêtées pour approximativement 99% du temps [63]. Cela est possible car ContikiMAC permet au capteur de ce réveiller périodiquement et d'écouter le support, si une transmission est détecter le capteur est garder éveillé pour recevoir le prochain paquet. Quand le paquet est reçu avec succès, le récepteur envoie un accusé de réception. L'émetteur ne cesse d'envoyer le même paquet jusqu'à ce qu'il reçoit un acquittement (et c'est le rôle de CSMA). Les paquets qui sont envoyés en broadcast ne réclament pas d'être acquittés, au lieu de cela, l'émetteur envoie à plusieurs reprises le paquet pendant la durée de son éveille, pour s'assurer que tous les voisins l'ont reçu.

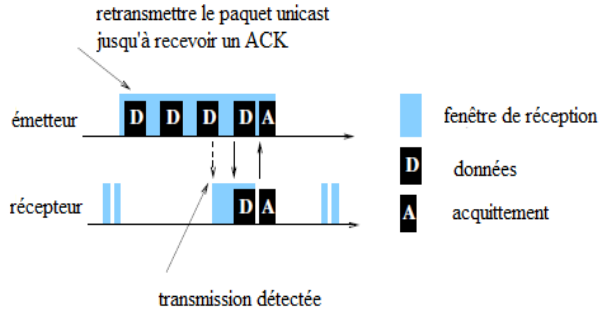


Fig. 4.3 : Le fonctionnement de contikiMAC lors d'un unicast [63]

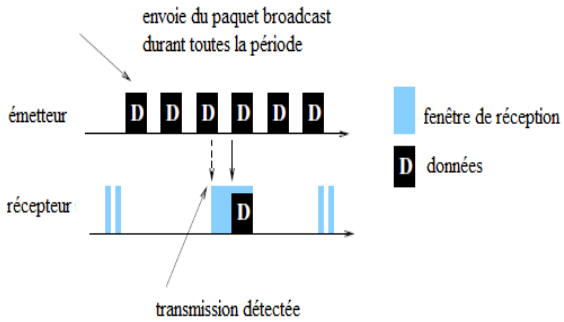


Fig. 4.4 : Le fonctionnement de contikiMAC lors d'un broadcast [63]

V. La description du protocole CSMA tel qu'il est implémenté dans Contiki

Dans CSMA il existe principalement deux structures de données importantes, la première est la liste des voisins et la deuxième est la liste de paquets jointe à un voisin dont il est le destinataire, le nombre de voisins et de paquets varie selon la plateforme et la pile réseau utilisée (uIP ou Rime).

À chaque arrivée d'un paquet à la couche MAC, CSMA vérifie si il s'agit d'un broadcast ou d'un unicast, un paquet broadcast est expédié immédiatement sans rien assigner ou allouer. Dans le cas d'un unicast, CSMA récupère l'adresse de destinataire, si ce destinataire n'existe pas dans la liste des voisins alors un espace mémoire est alloué pour l'ajouter ainsi que la liste des paquets à envoyer vers ce voisin, mais si ce voisin existe déjà alors un espace mémoire est alloué juste pour le nouveau paquet. À cette étape aussi CSMA initialise les métadonnées de ce nouveau paquet la plus importante d'entre elle est le nombre maximum de transmissions, cette valeur est vérifiée pour la première fois dans un champ de ce nouveau paquet, en cas où elle n'est pas définie alors c'est la valeur définie par la plateforme matériel ou dans le fichier CSMA lui-même qui est prise.

Si un paquet de données nouvellement généré est le premier dans la liste vers un voisin donné, CSMA tente de l'envoyer immédiatement sans trop attendre, mais s'il ne l'est pas il sera ajouté à la fin de la liste (la liste de paquets est gérée en FIFO sauf les acquittements qui sont placés à l'en-tête de la file).

Si la transmission d'un paquet réussie, le paquet et ces métadonnées sont aussi tôt supprimés en réinitialisant les champs de la structure de ce voisin à 0 pour le prochain paquet dans la liste, si ce paquet est le dernier, dans ce cas, si tout le voisin qui est supprimé de la liste des voisins.

L'envoi d'un paquet est considéré comme échoué si :

- Le capteur a réussie à le transmette mais il n'a reçu aucun acquittement, dans ce cas, le nombre de transmissions est incrémenté ;
- Une collision s'est produite en cours de transmission, dans ce cas, le nombre de collisions est incrémenté.

Si le nombre de transmission de ce paquet n'a pas dépassé le nombre maximum de transmissions fixé avant, le programme va réessayer de le retransmettre dans une durée calculée aléatoirement qui dépend de plusieurs paramètres :

$$T = time + (random\_rand\ ()\ \%\ (backoff\_transmissions * time))$$

- *time* : l'intervalle de vérification du canal définie par ContikiMAC;
- La fonction *random_rand ()* : tire un nombre entier aléatoire entre 0 et 65535 ;
- *backoff_transmissions* : reçoit le nombre de transmission + 1, le backoff est borné par la valeur 3 afin de ne pas attendre une long durée pour tenter une prochaine retransmission.

La formule assure un temps d'attente *T* tel que : *time <= T < 4 * time*.

Dans le cas où le nombre de transmissions dépasse 3 tentatives non acquittées, le paquet en question sera supprimé de la même manière qu'il est supprimé lors d'une transmission réussie.

Un autre aspect important de CSMA dans Contiki est qu'un premier paquet peut ne pas être encore envoyé que d'autres paquets arrivent (si la longueur de la liste des paquets le permet), et lorsque le capteur obtient enfin l'accès au canal il le monopolisera jusqu'à ce qu'il envoie tous ses paquets ce qui risque de prendre un peu de temps selon le nombre de paquet dans la liste.

CSMA tel qu'il est implémenté dans Contiki 2.6 est entièrement pensé pour ne pas perdre des paquets de données même s'ils subissent des collisions ou que leurs transmissions ne sont pas acquittées un certain nombre de fois, cette approche soulève quelques problématiques pour une application temps réel, voici les facteurs au niveau de CSMA qui conduisent à l'apparition des messages retardataires abordés dans la problématique :

1. La non limitation du temps de transmission

CSMA ne fait aucune hypothèse sur l'intervalle de temps que le paquet doit rester dans la file d'attente à cause des transmissions échouées avant d'être supprimé. La durée peut être longue, et si jamais le capteur réussie à transmettre le message, ce paquet sera sûrement un retardataire.

2. La non limitation du nombre de collisions

Dans le code source de CSMA à chaque fois qu'un paquet subit une collision, il existe une variable qui sera incrémentée mais elle n'est jamais comparée à une valeur maximale qu'elle ne doit pas dépasser, faute de quoi, le paquet sera supprimé. Si par exemple le paquet ne subit que des collisions alors il sera indéfiniment dans la liste des paquets occasionnant ainsi un très grand retard aux autres paquets.

3. La taille et la gestion de la fille d'attente des paquets

CSMA gère une file d'attente de paquets en FIFO (*First In First Out*), cette fille d'attente est d'une taille qui varie entre 8 et 16 selon la pile réseau utilisée, cela permet donc l'accumulation de plusieurs paquets dont les premiers de la file sont probablement au retard. Ajouter à cela, que les paquets nouvellement créés qui ne trouvent pas de place dans la file d'attente sont supprimés et ceux dans la liste sont toujours gardés.

Ces trois éléments font que les messages qui sont générés par les capteurs et qui ne sont toujours pas transmis vers le mobile à cause des collisions, soient gardés dans la file d'attente du capteur longtemps avant d'être supprimés ou transmis au retard.

NB. Le CSMA/CA tel qu'il est sous Contiki, on le nomme par csma/ca-Contiki dans ce qui suit.

VI. Le fonctionnement de l'application de localisation indoor

Nous avons vue dans le chapitre précédent qu'il existe plusieurs techniques afin de réaliser une localisation, dans ce travail il est question d'utiliser un réseau de capteurs afin de réaliser une application de localisation qui exploite la technique de la puissance du signal reçu RSSI pour le calcul des distances et la technique de Multi-latération pour le calcul de position en 2D du mobile (les coordonnées (x, y)).

L'application comptes trois entités distinctes, le mobile, les ancres ou immobiles, et une station de base. Le mobile est un capteur sans fil dont on veut connaitre sa position, les ancres sont aussi des capteurs dont la position est connue et la station de base c'est un ordinateur. Voici le fonctionnement étape par étape de l'application :

1. Le mobile et les capteurs ancres ouvrent deux connexions simultanées, une connexion broadcast en utilisant la primitive *Best-effort local area broadcast* et une connexion unicast en utilisant la primitive *Single-hop unicast*. Les deux primitives *broadcast* et *unicast* utilisent le module *abc* qui envoie des paquets juste aux voisins locaux (pas de routage), par conséquent, les ancres doivent tous être dans la portée du mobile.

2. Le mobile envoie un message broadcast qui peut être interprété comme une demande de localisation ;

3. Tous les capteurs ancres qui ont reçu la demande de localisation, extrairont l'adresse Rime du mobile est répondront par un message unicast.

4. Le nœud mobile recevra plusieurs messages unicast des différents capteurs ancres, dans ces paquets reçus il existe un champ dont le mobile à besoin précisément, c'est le champ RSSI, car on verra plus tard que grâce à ce dernier le mobile peut estimer la distance qui le sépare de capteur ancre à l'origine de message. Connaissant la distance qui le sépare par rapport à certains nombre de capteurs ancres, le mobile peut déduire sa position par une multi-latération.

5. Pour réduire la quantité de calcul que le mobile doit effectuer il transmit les RSSI reçus à la station de base et c'est à ce niveau que le système d'équation de la multi-latération est résolu et la position affichée à l'écran.

Les 4 étapes précédentes nous donnerons la position du mobile à un instant donné, or que nous, nous voulons avoir ces positions successives dans le temps. Pour se faire, les capteurs ancres doivent retransmettre périodiquement des messages unicast, dans ce qui suit nous appellerons cette période *le temps de régénération*. Les messages unicast sont marqués par un numéro afin de les identifier. A chaque fois qu'un capteur ancre envoie un message unicast son identifiant est incrémenté de 1. De cette façon, si un message arrive parmi des paquets ayant un identifiant supérieur au sien nous saurons qu'il s'agit d'un retardataire.

Maintenant que nous avons expliqué comment l'application fonctionne voici un exemple qui illustre le phénomène de messages retardataires, nous disposons de 3 capteurs ancres et un mobile, à l'instant *t0*, ils envoient tous le message numéro 1, le capteur mobile reçoit seulement 2 messages dont le numéro est 1 car le troisième capteur n'a pas pu envoyer son message. À l'instant *t1* les capteurs qui ont réussi à envoyer le message 1 se préparent à envoyer le message numéro 2, seulement, le capteur qui a échoué se prépare à retransmettre le message numéro 1, le mobile risque cette fois de recevoir 2 paquets numéro 2 et un paquet numéro 1, ce dernier est considéré comme au retard. Ce phénomène risque de se reproduire plus souvent au fur et à mesure que la localisation avance, nous entrainons de plus en plus dans des faux calculs.

Une autre subtilité est le choix de temps de régénération qui est délicat à faire, un temps réduit est synonyme d'une fluidité confortable mais la précision et réduite car de plus en plus de capteurs ancres n'arrivent pas à répondre dans le temps imposé par l'application. Tandis qu'un grand temps rend l'application moins fluide mais d'une grande précision vu le grand nombre de capteurs qui arrivent à répondre.

Le but de notre travail est de trouver dans un premier temps le moyen pour supprimer les messages retardataires et dans un second temps déduire une correspondance entre le temps de régénération et le nombre de capteurs ancres que nous voulons exploiter. Mais avant cela, nous allons réaliser des tests sans rien toucher à CSMA pour essayer d'étudier son fonctionnement.

VII. Tester le fonctionnement de CSMA/CA-Contiki

Nous avons codé la première partie de l'application qui est suffisante pour procéder à des tests et voir comment le protocole d'accès au support CSMA se comporte dans une application temps réel.

VII.1. Tests

Les tests ont été effectués sur deux topologies physiques différentes (étoile et grille), pour chaque topologie nous avons simulé 4 réseaux, ils contiennent respectivement 5, 10, 20 et 30 capteurs ancres. Pour chaque réseau, trois tests ont était effectués en variant le temps de régénération. Les tests ont était effectués dans le simulateur de réseaux de capteurs Cooja. Nous faisons durer la simulation environ 2 minutes et les résultats qui nous intéressent sont les suivant :

- **Le nombre de collisions** : c'est le nombre total de collisions survenant dans le réseau durant le temps de la simulation

- **La moyenne des messages arrivés en séquence**: si les capteurs ancres envoient un message chaque 2s par exemple, en 2 minutes(le temps de la simulation) chaque capteur va envoyer environ 60 messages. Ce qui fait que le capteur mobile peut recevoir jusqu'à 60 groupes de massages, les messages de chaque groupe ont le même identifiant qui sera incrémenté lors de prochain groupe. Les messages arrivés en séquence sont justement les messages portant le même identifiant reçu avant un premier message d'un identifiant supérieur. Nous additionnons le nombre de messages en séquence de tous les groupes puis nous divisons ça sur le nombre de groupe pour avoir la moyenne.
- **La moyenne des messages en retard** : un message est considéré en retard, si tout simplement il n'arrive pas dans son groupe, nous les identifions par leurs numéro qui est inférieur à celui du groupe dans lequel ils sont arrivés (comme dans la figure 4.1). La somme de nombre de messages en retard de tous les groupes sur le nombre de groupe nous donne la moyenne.

Exemple : reprenons l'exemple de 3 capteurs ancres et un capteur mobile, les capteurs ancres envoient chaque 2 second un message unicast, la suite des massages reçus par le mobile est représenté par la figure 4.5:

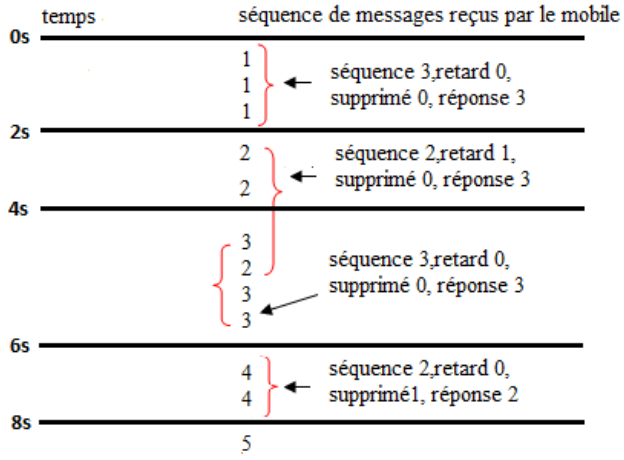


Fig. 4.5 : La suite des massages reçus par le mobile

D'après la figure, durant les 8s, le mobile a reçu 4 groupes de messages (séparés par des lignes dans la figure), un nouveau groupe arrive chaque 2 second. Le dernier message numéro 2 est un retardataire car il est arrivé parmi les messages numéro 3, mais la séquence des messages numéro 3 ne compte aucun retardataire même s'ils sont pas successifs car c'est le dernier 2 qui s'est glissé parmi eux. Le groupe de message numéro 4 ne compte aucun retardataire même si le mobile n'a reçu que 2 messages, car le dernier n'est pas arrivé au mobile. Les calculs se font de la manière suivante :

- La moyenne des messages en séquence = (3 + 2 + 3 +2) / 4 = 2.5 ;
- La moyenne des messages en retard = (0 + 1 + 0 + 0) / 4 = 0.25 ;

- **Le temps de réponse** : un autre facteur important à remarquer est le temps de réponse, par le temps de réponse nous entendons le temps qui s'est écoulé depuis la réception par le mobile du premier message d'un groupe donné jusqu'au dernier de ce même groupe, c'est donc le temps qu'il aurait fallu pour recevoir tous les messages de ce groupe.

- **Consommation d'énergie** : nous obtenons le pourcentage de la consommation d'énergie grâce à un plug-in que fournit le simulateur Cooja, cette valeur est un pourcentage qui représente la moyenne de la consommation d'énergie de tous les capteurs dans le réseau.

En plus de ces 5 valeurs, nous ajoutons les taux de séquence et de retard qui se calculent par rapport au nombre total de messages durant le temps de la simulation.

En réalité le mobile reçoit un nombre important de messages durant les 2 minutes de la simulation et il est impossible de faire ces calculs manuellement, pour cette raison nous avons écrit un programme en langage C qui effectuent ces calculs et imprime ensuite les résultats.

Les résultats des tests sont enregistrés dans les deux tableaux suivants :

Topologie	Nombre de capteurs	Temps de régénération	Nombre de collisions	Moy. des messages en séquence	Moy. des messages en retard	Temps de réponse moyen	Taux		Consommation d'énergie
							Séquence	Retard	
Etoile	1+5	0.5s	1029	3.26	1.49	2.74s	65.23%	29.79%	4.75%
		1s	611	4.69	0.13	0.66s	93.83%	2.67%	3.25%
		1.5s	336	5.00	0.00	0.55s	100.00%	0.00%	2.17%
	1+10	1s	2691	4.61	3.47	9.45s	46.10%	34.75%	4.27%
		2s	1638	9.00	0.15	1.74s	90%	1.53%	2.89%
		2.5s	1316	9.38	0.00	1.56s	93.83%	0.00%	2.38%
	1+20	2s	5861	8.22	5.00	35.81s	41.10%	25.00%	3.44%
		3s	4774	13.13	1.92	7.80s	65.64%	9.62%	3.54%
		4s	3945	15.93	0.00	3.25s	79.67%	0.00%	2.71%
	1+30	3s	9136	9.58	5.00	40.30s	32.83%	16.67%	3.64%
		5s	6879	17.75	0.42	6.28s	59.17%	1.39%	3.02%
		6s	5476	16.65	0.00	4.50s	55.50%	0.00%	2.69%

Tableau 4.1 : Les résultats des tests avec csma/ca-Contiki (topologie en étoile)

Topologie	Nombre de capteurs	Temps de régénération	Nombre de collisions	Moy. des messages en séquence	Moy. des messages en retard	Temps de réponse moyen	Taux		Consommation d'énergie
							Séquence	Retard	
Grille	1+5	0.5s	969	3.16	1.62	4.78	2.72s	63.29%	4.78%
		1s	604	4.76	0.15	4.92	0.73s	95.29%	3.72%
		1.5s	520	4.89	0.00	4.89	0.74s	97.72%	2.57%
	1+10	1s	2644	4.74	3.66	8.39	7.58s	47.35%	3.97%
		2s	1653	9.12	0.07	9.19	1.59s	91.19%	3.07%
		2.5s	1358	9.40	0.00	9.40	1.55s	94.04%	2.42%
	1+20	2s	5880	7.25	4.54	11.80	27.35s	36.27%	3.36%
		3s	4803	12.31	1.72	14.03	9.04s	61.54%	3.42%
		4s	3124	13.43	0.00	13.43	2.86s	67.17%	2.40%
	1+30	3s	9163	11.00	5.30	16.30	19.40s	36.67%	3.36%
		5s	6702	20.08	0.83	20.92	6.69s	66.94%	3.08%
		6s	4810	18.90	0.00	18.90	4.49s	63.00%	2.23%

Tableau 4.2 : Les résultats des tests avec csma/ca-Contiki (Topologie en grille)

VII.2. Analyse des résultats

Dans un premier temps, il convient de remarquer qu'il y'a pas une différence remarquable entre les résultats des deux topologies physiques en étoile et en grille. Les résultats des tests pour les deux topologies se rapprochent beaucoup car un simulateur ne permet pas de reproduire les vraies conditions de propagation des signaux hertziens. Pour cette raison nous allons nous contenter d'analyser la topologie en étoile car tout ce qui peut être dit sur CSMA ici s'applique à la topologie en grille.

- **Les collisions** : Dans les 4 réseaux en étoile, lorsque le temps de régénération était trop petit (*0.5s* pour 5 ancres, *1s* pour 10 ancres, *2s* pour 20 ancres et 3 pour 30 ancres), il se produit un grand nombre de collisions dans le réseau et plus le temps de la régénération augmente plus le nombre collisions diminue;

- **Les moyennes des messages en séquence et en retard** : C'est lors des temps de régénération les plus petits que nous avons enregistré le moins de séquence et le plus de retard, mais en augmentant le temps de régénération, la moyenne des messages en séquence augmente jusqu'à ce qu'elle s'approche de nombre total de capteurs sauf pour les réseaux avec 20 et 30 capteurs. Parallèlement, la moyenne des messages en retard diminue jusqu'à s'annuler.

- **Le temps de réponse** : Quand le temps de la régénération est petit le temps de réponse est le plus élevé, ce dernier dépasse même de loin le temps de la régénération (par exemple pour 10 capteur quand le temps de la régénération est de 1s, le temps de réponse est de 9.45s). Au ferai et à mesure que le temps de la génération augmente le temps de réponse diminue pour finir par être inférieur au temps de régénération.

- **Les Taux** : Les taux sont une autre manière de représenter les moyennes des messages en séquence et en retard, cette fois nous représentons ces deux notions en pourcentage par rapport au nombre total de messages. Ces taux évoluent de la même manière que les moyennes par rapport au temps de la régénération.

- **Consommation d'énergie** : quand le temps de la régénération est petit les capteurs consomment plus d'énergie, plus le temps de la régénération augmente, plus la consommation d'énergie est moins importante.

VII.3. Interprétation des résultats

- **Les collisions** : les capteurs ancres essayent tous de répondre au mobile à l'instant où ils recevront le broadcast ce qui augmente considérablement le risque de collision, de plus, si le temps de la régénération est petit les capteurs créent un nombre conséquent de messages à transmettre. En essayant d'accéder au canal dans un intervalle très petit, les instants d'envoie sont suffisamment proches pour qu'il y ait des collisions. Ce phénomène va se reproduire très souvent car il y a un nombre important de messages à transmettre mais aussi à retransmettre en cas de collision ou de transmission non acquittée. En augmentant le temps de la régénération moins de messages sont engendrés ce qui réduit le nombre de tentatives d'accès au support et donc de collisions.

- **Les moyennes des messages en séquence et en retard**: Avec un temps de régénération petit on a enregistré le moins de séquence et le plus de retard car les paquets subissent beaucoup de collisions ou des transmissions non acquittées. Un paquet dont la transmission a échoué par exemple à l'instant t0, CSMA va le retransmettre à un instant t1 = t0 + T (T est donné dans la description de CSMA), si la transmission échoue plusieurs fois aux instants t1, t2 et t3 CSMA permet toujours de reporter la retransmission à t4 = t3+T sauf que l'instant t4 risque de dépasser le temps de la régénération, très petit ici, et le paquet va donc arriver au retard. Un capteur peut aussi ne pas avoir accès au canal pendant longtemps à cause de trafic important dans le réseau et CSMA garde ce paquet toujours dans sa file d'attente et lorsqu'il a enfin accès au canal, ce message arrive au mobile très tardivement. Plus le temps de la régénération grandit plus les capteurs disposent d'un plus grand intervalle de temps pour retransmettre les paquets qui ont

subi des collisions ou des transmissions non acquittées ou d'accéder tout simplement au canal avant que les paquets de prochain groupe ne soient générés et par conséquence le nombre de messages retardataires se trouve réduit à 0 et le nombre de messages en séquence s'approche de nombre de capteurs total.

CSMA supprime un paquet qui a attient trois transmission non acquittées, les paquets atteignent facilement ce nombre dans un temps de régénération très petit à cause des collisions très fréquentes et ils se trouvent donc supprimés par CSMA d'où la moyenne de séquence un peu faible. Plus le temps de régénération grandit moins il y a de collisions et moins CSMA supprime de paquet d'où la moyenne de séquence qui s'améliore. Dans les réseaux qui contiennent 20 (resp. 30) capteurs ancres, même avec un temps de régénération de 4s (resp. 6s) nous n'arrivons pas à exploiter tous les capteurs alors qu'il y a aucun retardataire, la raison à cela est que le nombre de collision reste assez élevé pour que CSMA supprime des paquets à leur troisième transmission non acquittée mais le temps de régénération est suffisamment grand pour que tous les paquets d'un groupe soient envoyés ou supprimées avant que les prochains messages arrivent.

- **Le temps de réponse** : le temps de réponse est plus élevé dans un temps de régénération faible car il y a beaucoup de messages retardataires, et plus le temps de la régénération augmente moins il y a de retardataires et plus le temps de réponse diminue.

- **Consommation d'énergie** : Un temps de régénération trop petit conduit à la transmission d'un grand nombre de messages, en plus des retransmissions des paquets dont les envois précédents ont échoué, d'où la grande consommation de l'énergie. Plus le temps de régénération augmente plus le nombre d'erreurs dans le réseau et de messages générés par capteur diminue, ainsi, l'activité du module radio se trouve réduite et le capteur consomme moins d'énergie.

VII.4. Evaluation

Pour résumer les résultats des tests, nous pouvons dire qu'un temps de régénération petit par rapport au nombre de capteurs ancres dans le réseau conduit à des collisions et puisque CSMA persiste à retransmettre ces paquets cela cause des messages retardataires qui augmentent à leur tour le temps de réponse. Il est clair que plus le nombre de capteurs est grand, plus il faut un temps de régénération plus important pour atteindre un bon résultat :

- 0 message en retard et donc un temps de réponse inférieur à celui de la régénération;

- Plus de message qui arrive en séquence.

Mais même avec un grand temps nous n'arrivons toujours pas à exploiter la totalité des capteurs dans le cas où le nombre de ces derniers est conséquent. En réduisant ce temps pour actualiser la position du mobile plus fréquemment nous sommes confrontés aux problèmes de collisions, malheureusement ces dernières sont pas sans conséquence car CSMA retente de transmettre ces paquets pendant une long durée consommant ainsi beaucoup d'énergie, mais en plus, ces messages ne serviront à rien car ils seront très en retard.

Il faut donc trouver une solution pour réduire le nombre de messages retardataires et augmenter par la même occasion le nombre de messages en séquence même avec un grand nombre de capteurs ancres. Pour résoudre ce problème nous introduisons dans ce qui suit trois propositions.

VIII. Propositions

Afin de remédier aux anomalies cités dans la problématique, et d'atteindre notre but qui est de recevoir le plus possible de messages en séquence que le temps de la régénération le permet sans retardataire, nous allons présenter trois propositions. Pour la première proposition nous avons introduit une nouvelle entité dans le réseau que nous avons appelé *Coordinateur*. La seconde proposition consiste en une légère modification de CSMA, tandis que la troisième est un ensemble de changements au niveau de CSMA qui vont le rendre plus compact en temps réel. Pour chaque proposition nous adoptons la même structure à suivre :

1. Le fonctionnement ;
2. Les changements apportés à CSMA ;
3. Résultat de test ;
4. Analyse et interprétation ;
5. Evaluation.

VIII.1. Le Coordinateur

Le Coordinateur est un nouveau capteur sans fil que nous ajoutons dans le réseau à la portée des autres capteurs, comme son nom l'indique, le rôle de cette entité est la coordination dans le réseau.

VIII.1.1. Fonctionnement

L'introduction du coordinateur change significativement le fonctionnement de l'application de localisation, le mobile s'adresse maintenant au coordinateur pour demander sa localisation. Voici son fonctionnement détaillé :

1. Les capteurs ancres envoient au coordinateur des annonces grâce au module *Announcements* de la couche Rime;
2. Lorsque le coordinateur reçoit les annonces il ajoute les capteurs correspondants à une liste, si le capteur à l'origine de l'annonce existe déjà dans la liste, le coordinateur renouvelle son timeout, si le timeout d'un capteur expire avant que le coordinateur ne reçoit une nouvelle annonce de ce capteur, il le supprime de la liste (considéré comme éteint ou en panne);
3. Le mobile envoie un broadcast au coordinateur et il inclut dans le message une période *P*. La période *P* correspond au temps de la régénération vue précédemment.
4. Le coordinateur récupère l'adresse du mobile et l'intervalle *P*, il divise en suite *P* sur le nombre de capteurs ancres dans la liste pour obtenir un intervalle *i* alloué pour chaque capteur ancres;
5. Le coordinateur parcourt la liste des capteurs ancres un par un pour leur donner l'autorisation d'envoyer un message au mobile. L'autorisation consiste en un message unicast envoyé du coordinateur au capteur ancre, ce message inclut l'adresse du mobile et l'intervalle *i*, le coordinateur attend à chaque fois la durée *i* avant de donner l'autorisation au prochain capteur dans la liste;
6. Si un capteur ancres réussie à envoyer le message au mobile avant que le temps qui lui est alloué (*i*) ne soit terminé, il informe le coordinateur en lui envoyant un message pour passer au prochain capteur dans la liste ;
7. Le mobile reçoit enfin les messages unicast l'un après l'autre durant *P*.

VIII.1.2. Le changement apporté à CSMA

L'idée derrière le coordinateur et d'essayer de pallier la problématique tout en restant au niveau application par conséquent nous n'avons pas modifié CSMA ici sauf pour la liste des voisins que nous avons allongé car le coordinateur envoie des messages unicast à plusieurs capteurs ancres.

VIII.1.3. Résultats des tests

Afin d'évaluer le fonctionnement du coordinateur, Nous avons refait les tests exactement de la même manière qu'avec le CSMA tel qu'il est sous contiki, mais uniquement pour la topologie en étoile en gardant les mêmes temps de régénération, les résultats sont enregistrés dans le tableau suivant :

Topologie	Nombre de capteurs	Temps de régénération	Collisions	Moy. des messages en séquence	Moy. des messages en retard	Temps de réponse moyen	Taux		Consommation d'énergie
							Séquence	Retard	
Etoile	1+5	0.5s	1668	1.32	3.25	14.65s	26.40 %	65.08 %	7.01%
		1s	1064	3.08	1.82	2.91s	61.69 %	36.36 %	6.12%
		1.5s	825	4.44	0.53	1.16s	88.83 %	10.63 %	5.22%
	1+10	1s	2670	2.84	5.95	12.59s	28.43 %	59.48 %	5.72%
		2s	1617	5.33	4.13	9.65s	53.33 %	41.33 %	4.95%
		2.5s	870	6.71	2.82	12.29s	67.09 %	28.18 %	4.07%
	1+20	2s	3113	5.17	9.48	47.93s	25.83 %	47.40 %	4.04%
		3s	3131	7.08	8.53	43.20s	35.39 %	42.63 %	3.84%
		4s	2594	9.00	8.11	20.36s	45.00 %	40.54 %	4.23%
	1+30	3s	4556	5.59	10.90	45.82s	18.62%	36.32%	3.83%
		5s	2215	12.00	14.27	40.93s	40.00%	47.58%	3.15%
		6s	2843	12.26	9.05	38.81s	40.88%	30.18%	3.13%

Tableau 4.3 **:** Les résultats des tests avec le coordinateur

VII.1.4. Analyse et interprétation des résultats

Par rapport au tableau 4.1, nous remarquons une diminution du nombre de collisions, ainsi un nombre de messages en séquence plus réduit et un nombre de messages retardataires plus important (voir les figures 4.6 et 4.7). Les temps de réponse dépassent aussi largement le temps alloué.

NB. Toutes les courbes de comparaison reposent sur le cas du réseau de 20 capteurs.

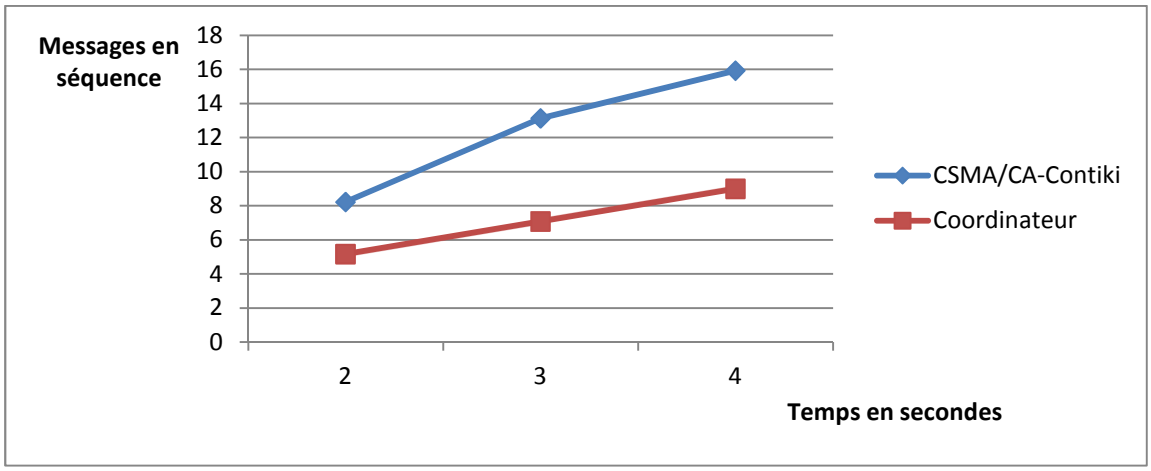


Fig. 4.6 : La comparaison entre csma/ca-Contiki et le coordinateur (séquence)

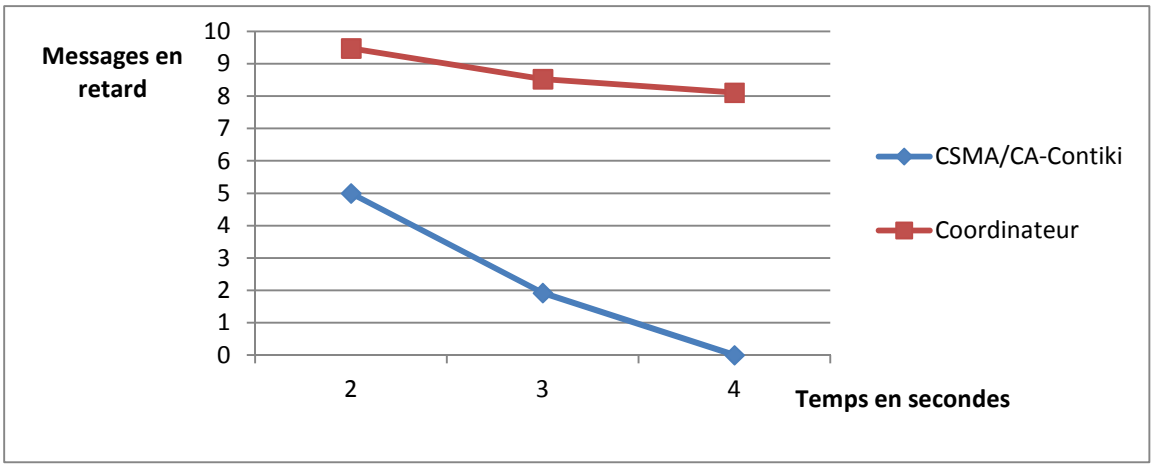


Fig.4.7 : La comparaison entre csma/ca-Contiki et le coordinateur (retard)

En ce qui concerne la consommation d'énergie nous constatons qu'elle est plus importante (voir la figure 4.8).

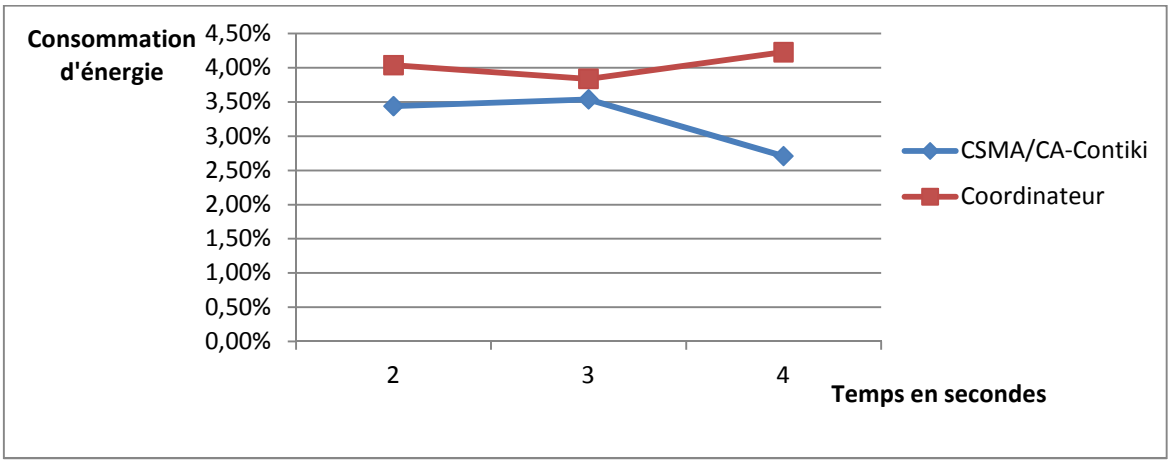


Fig. 4.8 : La comparaison entre csma/ca-Contiki et le coordinateur (énergie)

Le coordinateur a donné des résultats très médiocres. Avec le coordinateur nous avons essayé de réduire le risque de collisions à 0, logiquement le coordinateur permet à un seul capteur ancre d'envoyer un message au mobile à un instant donné parce qu'il leur donne l'autorisation à des instants différents et il n'y a donc aucun risque de collision. En essayant d'ordonner les envois de paquets des ancres vers le mobile nous avons créé un trafic important dans le réseau. En effet, les nœuds ancres envoient des annonces pour s'enregistrer, le coordinateur envoie aux ancres des messages unicast pour les autoriser et les ancres envoient aussi des messages unicast au coordinateur pour confirmer qu'ils ont réussi d'envoyer vers le mobile. Cela fait trois nouveaux messages supplémentaires.

Puisque les autorisations sont des messages unicast, elles subissent aussi des collisions, par conséquence, certaines d'entre elles sont supprimées les autres arrivent aux capteurs ancres au retard. Celles qui sont supprimées ont contribué à diminuer le nombre de message générés dans le réseau, en effet, si les ancres ne reçoivent pas les autorisations ils ne vont pas envoyer vers le mobile, cela explique la diminution de nombre de collision par rapport à csma/ca-Contiki. Et si l'autorisation arrive au retard il est clair que les actions déclenchées par cette dernière seront aussi au retard cela explique la diminution des messages en séquence et l'augmentation des retardataires.

Par ailleurs, le coordinateur n'a aucun contrôle sur le moment dont le capteur ancre va transmettre son message après lui avoir donné l'autorisation, même en admettant que les autorisations arrivent dans un temps approprié aux capteurs ancres, les messages envoyés par ces derniers vers le mobile risquent aussi de subir des collisions avec des annonces, le capteur ancre va essayer de retransmettre le paquet ultérieurement, si le temps de la régénération est assez petit le paquet en question arrivera très au retard.

Le coordinateur a conduit à une plus grande consommation d'énergie parce qu'il y a plusieurs type de messages à gérés et un nombre important de retransmissions à cause de nombre d'erreurs élevé dans le réseau.

VII.1.5. Evaluation

A cause des messages supplémentaires que nous avons injectés dans le réseau nous avons amplifié le problème en créant un message qui ne tolère pas les collisions (message d'autorisation), cela provoque un retard d'envoi et le retard augmente le temps de réponse. Il est important de remarqué aussi que le coordinateur ne résiste pas aux pannes, en effet, ce dernier gère trois type de messages : les autorisations, les annonces et les confirmations des capteurs ancres, tous ces messages conduisent à une plus grande consommation d'énergie, le coordinateur risque donc d'épuiser sa source d'énergie et de s'éteindre rapidement et sans coordinateur la localisation ne peut plus fonctionner.

Nous avons échoué de résoudre le problème en restant à la couche application, la prochaine étape consiste à essayer de le résoudre directement au niveau MAC.

VII.2. CSMA avec un nombre maximum de transmission réduit

CSMA supprime déjà les paquets qui atteignent leurs troisième transmission non acquittée ce qui explique les taux de réponse (séquence + retard) qui n'atteint pas 100% lors des temps de régénération les plus petits dans le tableau 4.1, mais le nombre de messages aux retard reste relativement élevés, on en déduit que CSMA donne plus de chance au paquet afin qu'il soit transmis tard . En supprimant les paquets qui subissent des erreurs d'envoi plus rapidement de la liste des paquets, ils ne vont pas être transmis en retard et ils devront laisser la place aux nouveaux paque²ts, ainsi le phénomène de messages retardataires devrait régresser.

VII.2.1. Fonctionnement

Lorsqu' un capteur ne reçoit pas un acquittement pour un paquet transmis en unicast sans collision, il incrémente la variable qui trace le nombre de transmissions puis la compare avec une constante qui se trouve dans le code source CSMA/CA, si la variable dépasse cette constante, CSMA va supprimer ce paquet. Nous allons réduire la valeur de cette constante, afin que CSMA supprime plus rapidement les paquets qui seront probablement des retardataires.

VII.2.2. Le changement apporté à CSMA

Nous allons diminuer le nombre maximum de transmissions tolérées par CSMA qui était de 3 à 1, en éditant la valeur de la constante de préprocesseur qui se trouve au niveau de fichier source CSMA (*#define CSMA_MAX_MAC_TRANSMISSIONS 1*).

VII.2.3. Résultats des tests

Nous avons refait les tests pour la topologie en étoile, les résultats sont enregistrés dans le tableau suivant :

Topologie	Nombre de capteurs	Temps de régénération	Collisions	Moy. des messages en séquence	Moy. des messages en retard	Temps de réponse moyen	Taux		Consommation d'énergie
							Séquence	Retard	
Etoile	1+5	0.5s	1166	2.01	0.44	1.08s	40.21%	8.70%	4.72%
		1s	515	3.34	0.00	0.41s	66.89%	0.00%	2.63%
	1+10	1s	1692	3.52	0.23	0.80s	35.22%	2.27%	3.41%
		2s	905	3.15	0.00	0.97s	31.48%	0.00%	2.00%
	1+20	2s	2509	5.17	0.02	0.91s	25.83%	0.08%	2.25%
		3s	2145	6.20	0.00	1.14s	31.00%	0.00%	2.00%
	1+30	3s	4103	7.53	0.07	1.62s	25.08%	0.25%	2.04%
		4s	3675	7.73	0.00	2.87s	25.78%	0.00%	2.27%

Tableau 4.4 **:** Les résultats des tests avec max de transmission égale à 1

VII.2.4. Analyse et interprétation des résultats

En comparant avec les résultats obtenus avec le CSMA/CA-Contiki (tableau 4.1), nous distinguons que le nombre de collisions a diminué. Le nombre de message qui arrivent en séquence est nettement plus bas (voir la figure 4.9).

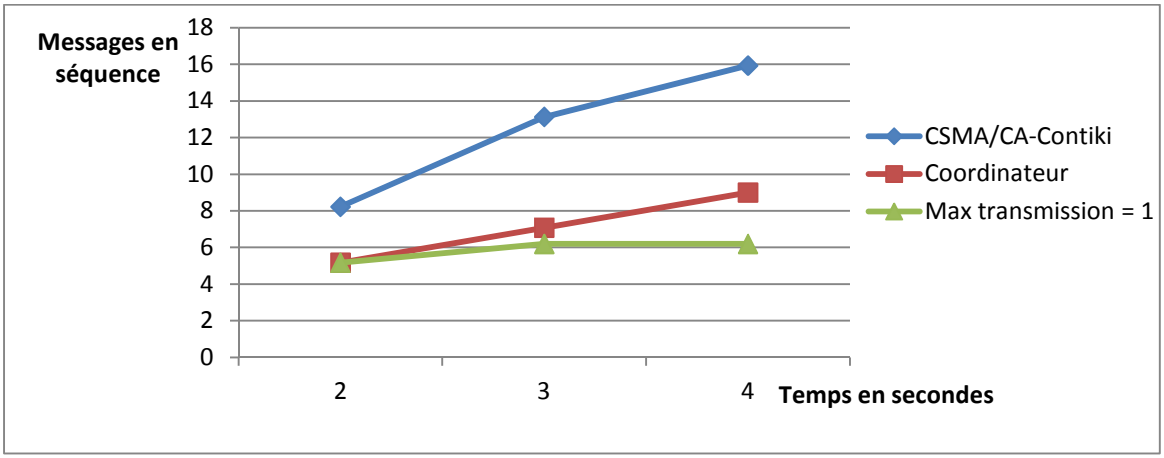


Fig. 4.9 : La comparaison csma/ca-Contiki/Coordinateur/Max tran = 1 (séquence)

Les messages retardataires apparaissent avec de faible quantité seulement dans les temps de régénération les plus petit sinon pour un grand temps de régénération ils disparaissent complètement (voir la figure 4.10).

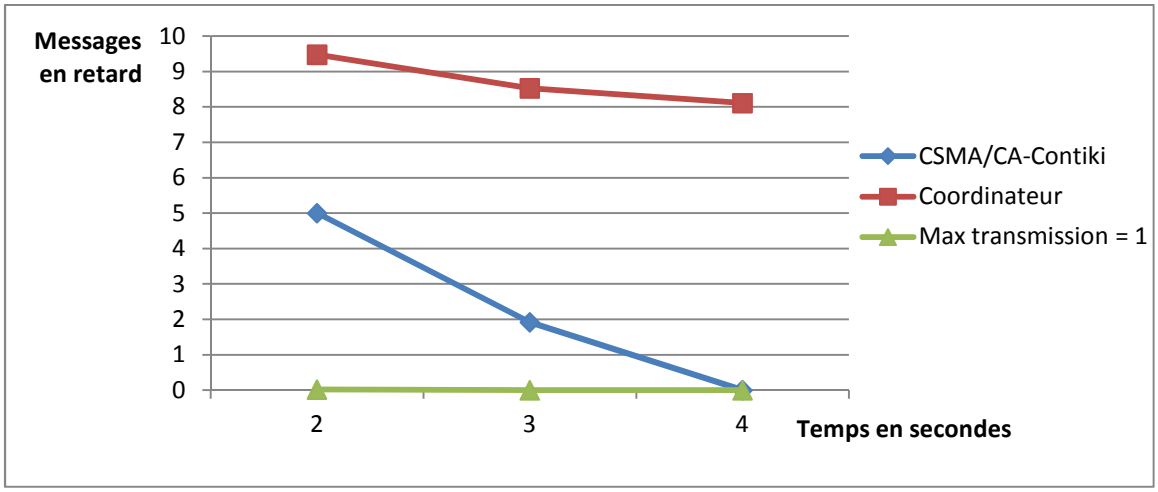


Fig. 4.10 : La comparaison csma/ca-Contiki/Coordinateur/Max tran = 1 (retard)

Le temps de réponse est toujours inférieur strictement au temps imposé (temps de la localisation). Une diminution peut être notée en ce qui concerne la consommation d'énergie (voir la figure 4.11).

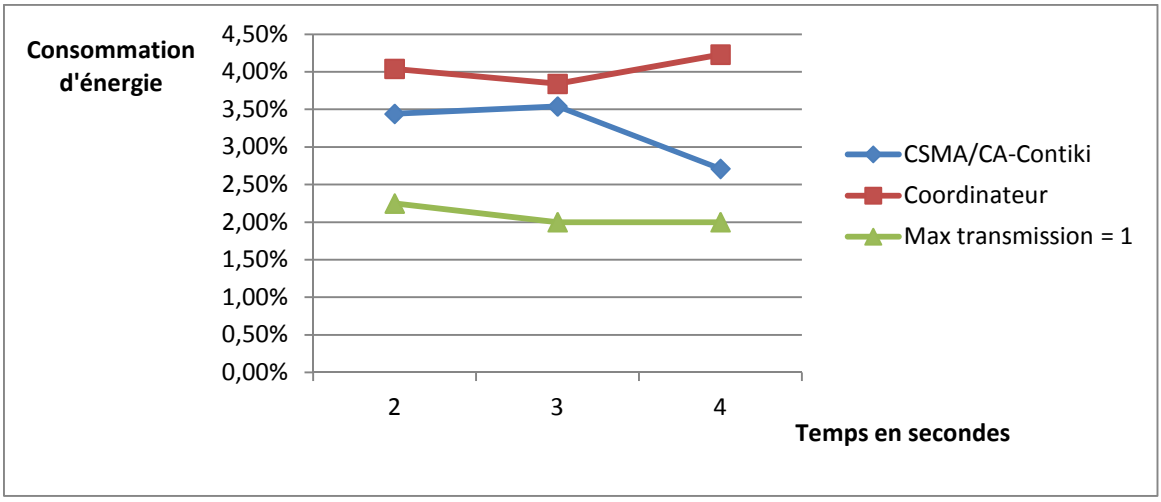


Fig. 4.11 : La comparaison csma/ca-Contiki/Coordinateur/Max tran = 1 (énergie)

Nous expliquons la diminution sur tous les niveaux : collision, séquence, retard et temps de réponse, etc. par le changement de nombre maximum de transmission que nous avons réduit de 3 à 1, CSMA supprime maintenant un paquet dès sa première transmission non acquittée contrairement à avant où il tolérait jusqu'à 3 transmissions non acquittées. Un nombre important de paquets sont donc supprimés et cela a pour effet de :

1. Réduire le nombre de collision car un paquet ne va pas être retransmit plusieurs fois au risque de subir d'autres collisions ;
2. Réduire le nombre de messages qui arrivent au mobile en séquence ;
3. La majorité des paquets sont supprimés avant que les prochain messages ne soient générées ce qui élimine les retardataire ;
4. Puisque le nombre de messages qui arrivent au mobile est très réduit c'est logique que le temps de réponse le soit aussi ;
5. Un capteur dépense moins d'énergie sur un paquet car il tente de l'envoyer qu'une seule fois.

VII.2.5. Evaluation

Réduire le nombre maximum de transmission possède plusieurs avantages et un inconvénient majeur. Les avantages sont que le mobile ne reçoit plus de messages retardataires et le temps de réponse ainsi que la consommation d'énergie sont réduits. L'inconvénient est que moins de messages arrivent au mobile même en augmentant le temps de la régénération, avec 10 capteurs il y a seulement 3 messages, avec 20 capteurs il y a 6 messages et pour 30 capteurs le mobile ne reçoit pas plus de 8 messages, nous arrivons donc pas à exploiter la totalité des capteurs même avec un grand délai.

CSMA supprime très vite les messages sans exploiter la totalité de temps qui lui est alloué (le temps de la régénération), en effet à leur premier échec d'envoie CSMA possède encore le temps pour les retransmettre sans qu'ils soient des retardataire. Dans la prochaine proposition nous allons essayer d'utiliser ce temps au mieux afin d'obtenir de meilleurs résultats.

VII.3. Limitation du temps de transmission d'un paquet

Lorsque nous avons diminué le nombre maximum de transmission les résultats se sont un petit peu améliorés mais le mobile recevait très peu de messages en séquence et nous avons expliqué ça par le fait que CSMA supprime les paquets qui échouent mais nous avons également dit que CSMA supprime très vite les paquets sans exploiter la totalité de temps de la régénération. Nous allons continuer dans le même ordre d'idée mais en permettant à CSMA d'utiliser au mieux cette durée.

VII.3.1. Fonctionnement

Le mobile envoie un broadcast aux capteurs ancres pour initier la localisation. Il inclut dans ce broadcast une période de temps qui sera récupérée par les ancres (*le temps de la régénération*), les ancres attendent à chaque fois cette période avant d'envoyer un nouveau message vers le mobile.

Par ailleurs, dans cette solution, un capteur gère ce temps comme le montre la figure 4.12 :

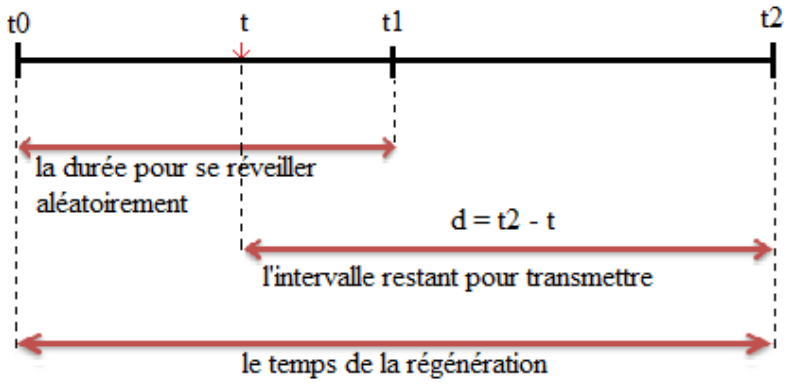


Fig. 4.12 : La durée *d* pour envoyer un message

- diviser la période en deux parties, la première entre *t0* et *t1* et le deuxième continue de *t1* à *t2* ;
- se réveiller aléatoirement durant la première partie à instant *t*;
- essayer d'envoyer le paquet durant *d* ;

L'idée est que à partir du moment où le capteur se réveille à l'instant *t* pour envoyer un paquet, CSMA alloue à ce paquet une durée *d*, durant laquelle il doit le transmettre, si à cause des collisions ou la non réception d'un acquittement le capteur n'a pas réussie à transmettre ce paquet pendant *d* il doit le supprimer. Un tel mécanisme devrait éliminer complètement les messages en retard, mais pour la sécurité nous avons également réduit la taille de la file d'attente des paquets à 1 et la génération d'un nouveau paquet supprime l'ancien, s'il existe.

VII.3.2. Le changement apporté à CSMA

Dans un premier temps nous avons ajouté un attribue au paquet que nous avons nommé *PACKETBUF_ATTR_MAX_MAC_INTERVAL*, il consiste en l'intervalle de temps alloué pour chaque paquet. Cet attribue peut parfaitement être assigné directement dans la couche application par le capteur ancre en utilisant la fonction *packetbuf_set_attr ()*, mais si on omet de l'assigné à ce niveau, CSMA va prendre la valeur d'une constante *CSMA_MAX_MAC_INTERVAL* que nous avons définie directement dans le ficher *csma.c* comme valeur par défaut. Dans les deux cas CSMA doit donc récupérer cette valeur afin qu'il puisse initialiser le champ *max_interval* que nous avons également ajouté à la structure qui représente les métadonnées d'un paquet, *qbuf_metadata*.

Dans une autre structure *neighbor_queue* toujours au niveau de CSMA, nous avons ajouté le champ *interval*. Ce champ a pour rôle de tracer la durée écoulé depuis la première tentative de transmission échouée, bien entendu cette valeur est remise à 0 pour chaque nouveau paquet.

À chaque fois que la transmission d'un paquet subit un échec le paquet sera retransmis dans une durée calculée aléatoirement (la durée *T* donnée dans la description de CSMA) en même temps le champ *interval* sera incrémenté de cette durée. Si le champ *interval* dépasse l'intervalle *max_interval* de ce même paquet, ce dernier sera supprimé. Logiquement deux paquets ne peuvent pas coexistés dans la file d'attente car avant la génération du deuxième, le première est envoyé ou supprimé, mais pour être sûre, nous avons réduit la taille de la fille d'attente à 1 et nous avons fait en sorte que la génération d'un nouveau paquet supprime l'ancien s'il est toujours dans la file.

VII.3.3. Résultats des tests

Pour ces tests nous avons ajouté 2 nouveaux réseaux qui comptent respectivement 40 et 50 capteurs afin d'essayer de déduire une relation entre le temps de régénération et le nombre maximal de capteurs que ce temps permet d'exploiter. Le les résultats sont représentés dans le tableau suivant :

Topologie	Nombre de capteurs	Temps de régénération	Collisions	Moy. des messages en séquence	Moy. des messages en retard	Temps de réponse moyen	Taux		Consommation d'énergie
							Séquence	Retard	
Etoile	1+5	0.5s	1199	3.07	0.02	0.42s	61.36 %	0.42 %	4.10%
		1s	500	4.80	0.00	0.63s	95.97 %	0.00 %	2.94%
		1.5s	287	5.00	0.00	0.69s	100 %	0.00 %	2.22%
	1+10	1s	2282	6.02	0.12	0.83s	60.17%	1.19 %	3.25%
		2s	906	9.63	0.03	1.54s	96.33 %	0.33 %	2.02%
		2.5s	330	9.94	0.04	1.69s	99.38 %	0.42 %	1.78%
	1+20	2s	4603	9.75	0.35	1.82s	48.75 %	1.75 %	2.96%
		3s	2641	15.43	0.38	2.96s	77.12 %	1.88 %	2.54%
		4s	1059	19.73	0.07	3.41s	98.67 %	0.33 %	1.60%
	1+30	3s	7206	12.32	0.28	2.75s	41.08%	0.92%	3.06%
		5s	3672	22.67	0.75	5s	75.56%	2.50%	2.29%
		6s	1808	27.55	0.25	5.38s	91.83%	0.83%	1.72%
	1+40	8s	2260	38.33	0.27	7.37s	95.83%	0.67%	1.90%
	1+50	12.5s	1424	49.80	0.20	9.57s	99.60%	0.40%	1.36%

Tableau 4.5 : Les résultats des tests avec csma/ca-Adapté

VII.3.4. Analyse et interprétation des résultats

Nous constatons par rapport au csma/ca-Contiki, une réduction des collisions, l'augmentation du nombre de messages en séquence (voir la figure 4.13), l'élimination des messages retardataires (voir la figure 4.14). En ce qui concerne le temps de réponse, nous remarquons une grande amélioration surtout dans les temps de la localisation très petit. L'énergie est aussi mieux économisée dans cette solution (voir la figure 4.15).

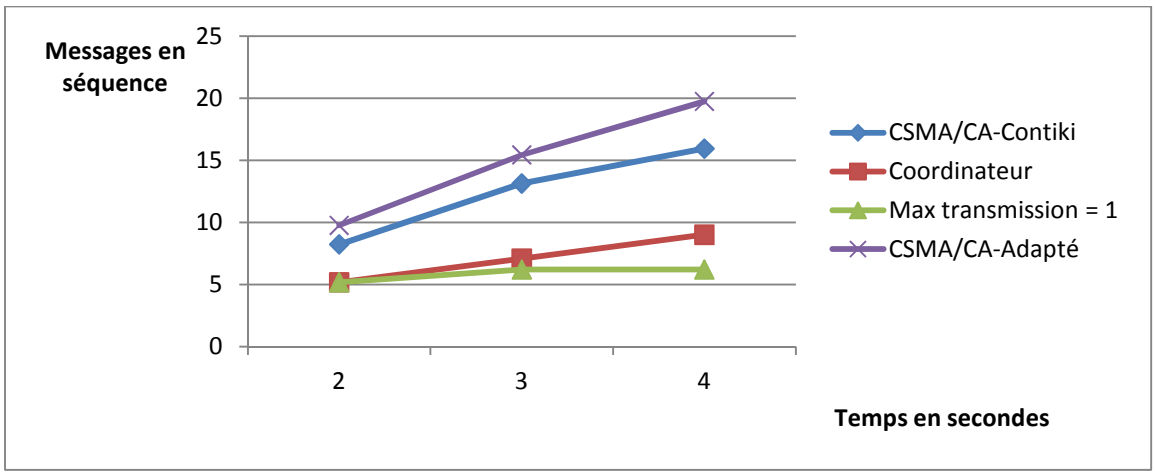


Fig. 4.13 : La comparaison csma/ca-Contiki/Coordinateur/Max trans=1/ csma/ca-Adapté (séquence)

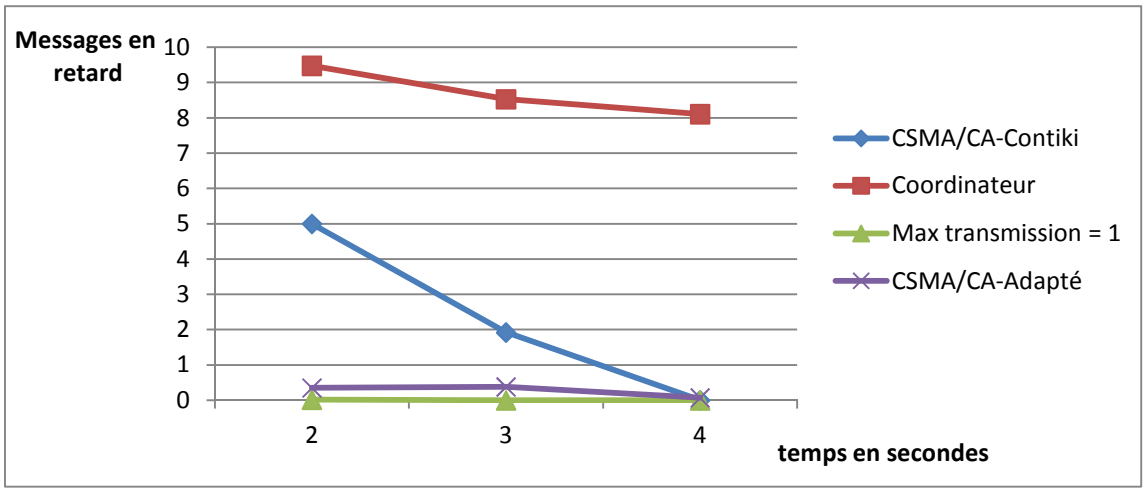


Fig. 4.14 : La comparaison csma/ca-Contiki/Coordinateur/Max trans = 1/csma/ca-Adapté (retard)

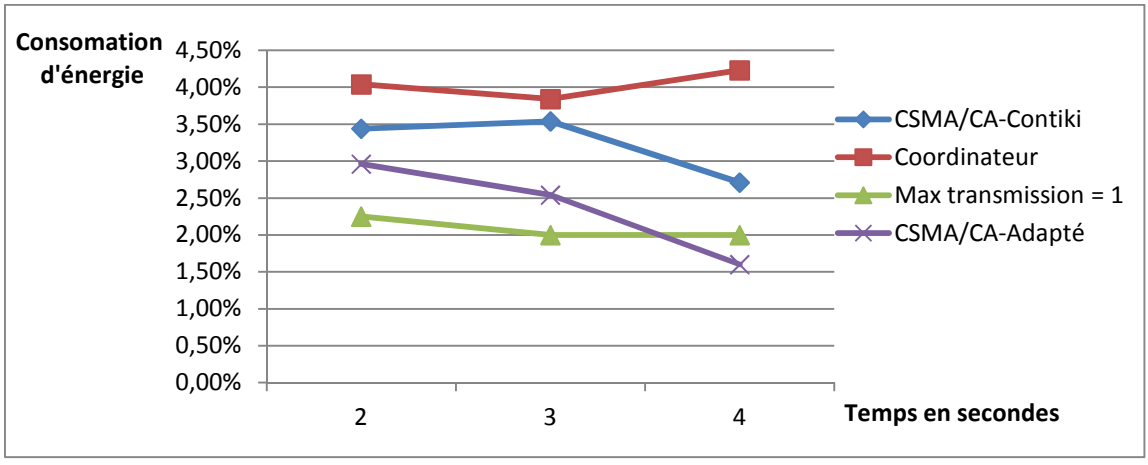


Fig. 4.15: La comparaison csma/ca-Contiki/Coordinateur/Max tran = 1/csma/ca-Adapté (énergie)

On doit la réduction du nombre de collisions à une meilleure gestion du temps de régénération, en effet, les capteurs se réveillent aléatoirement dans une période bien déterminée, ce qui réduit les collisions.

Réduire le nombre de collisions ainsi que donner le plus grand intervalle de temps possible *d* pour envoyer un paquet a contribué à l'augmentation du nombre de messages en séquences.

CSMA supprime les paquets dont il n'a pas réussi la transmission après l'écoulement de temps alloué *d*, en plus, la taille de la file des paquets est réduite à 1 et la génération d'un nouveau paquet supprime le retardataire, cela a éliminé les retardataires, mais nous remarquons l'apparition de quelques retardataires dans les cas où nous avons un grand nombre de capteurs (cas 1+20) cela est dû à la non synchronisation des capteurs ancres, quoique, leurs nombre reste toujours une quantité négligeable.

Le fait de ne pas avoir des messages retardataires a nettement amélioré les temps de réponse qui restent toujours inférieurs au temps de régénération même dans une forte contrainte temporelle.

La consommation d'énergie est relative au nombre de retransmissions, dans csma/ca-Contiki, un paquet peut être retransmis un grand nombre de fois avant d'être supprimé consommant ainsi une grande quantité d'énergie, mais avec cette solution le nombre de retransmissions est limité par l'intervalle de temps, moins de retransmission implique moins d'énergie consommée.

VII.3.5. Evaluation

Les résultats montrent que cette solution offre plusieurs avantages par rapport à csma/ca-Contiki et les deux solutions précédentes, car elle assure :

1. l'élimination complète des messages retardataires même pour les temps de régénération les plus petits avec un grand nombre de capteurs ;
2. garantir un temps de réponse inférieur au délai de la localisation quel que soit ce dernier;
3. une meilleure conservation de l'énergie prolongeant la durée de vie du réseau ;
4. une grande exploitation des capteurs si le temps de la régénération le permet, en effet, le seul critère qui influence la suppression des messages et donc l'exploitation des capteurs est le temps de la régénération (voir la figure 4.16).

NB. Cette solution on va la nommée csma/ca-adapté.

VIII.4 Evaluation Finale

Après avoir étudié chacune des trois propositions précédentes et csma/ca-Contiki, nous pouvons conclure que la dernière d'entre elles (csma/ca-adapté) est une solution adéquate et plus appropriée que les autres pour les systèmes temps réel, car elle satisfit la contrainte temporelle (élimination des messages retardataires) et garantit une meilleure exploitation des capteurs et l'économie de l'énergie n'est pas laissée en reste, ce que csma/ca-Contiki ne peut pas les garantir. Les deux figures suivantes représentent et affirment l'amélioration apportée par csma/ca-Adapté par rapport à csma/ca-Contiki dont la première décrit la relation entre le temps et le nombre de capteurs qu'on peut exploiter, la seconde décrit la consommation de l'énergie par capteur :

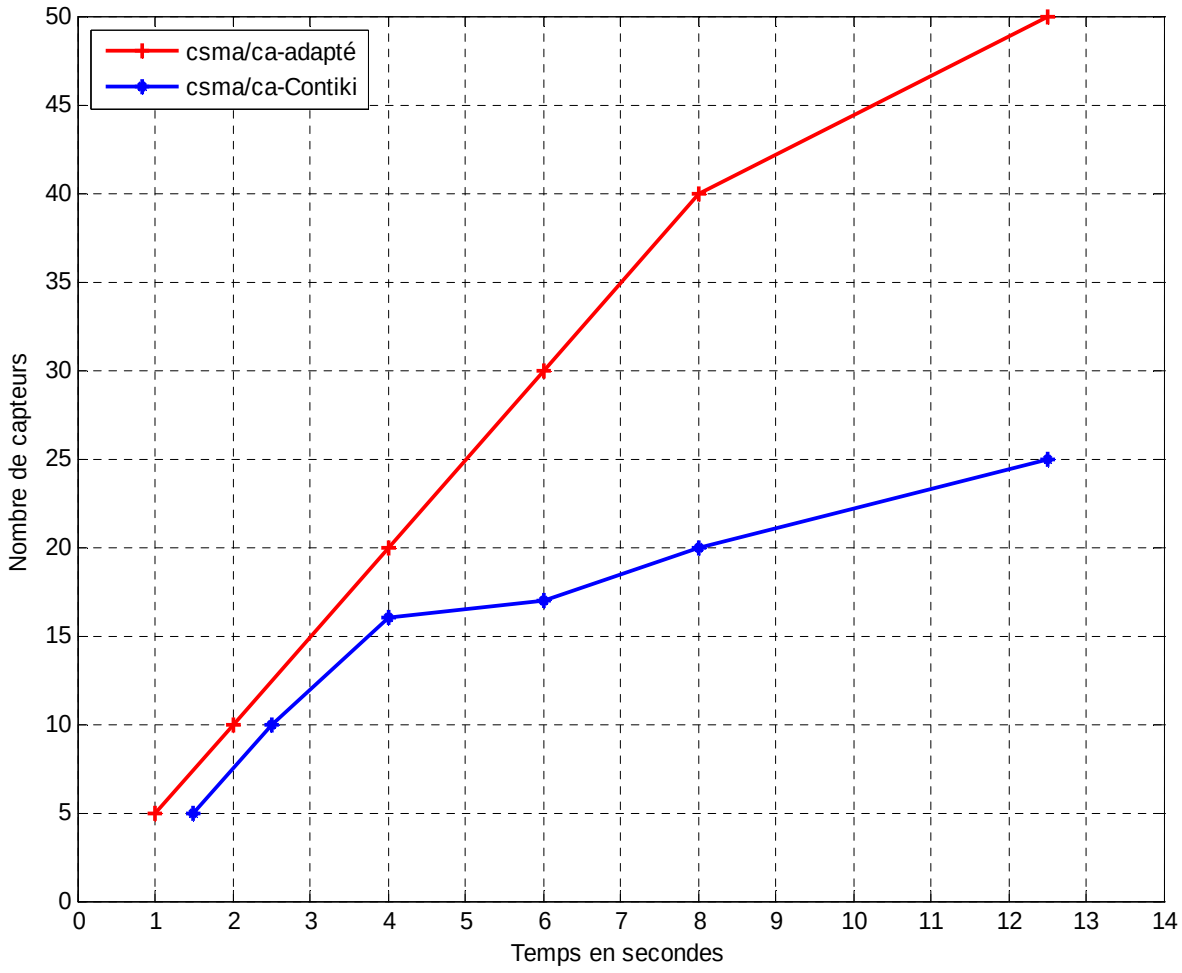


Fig. 4.16: La relation entre le temps et le nombre de capteurs qu'on peut exploiter

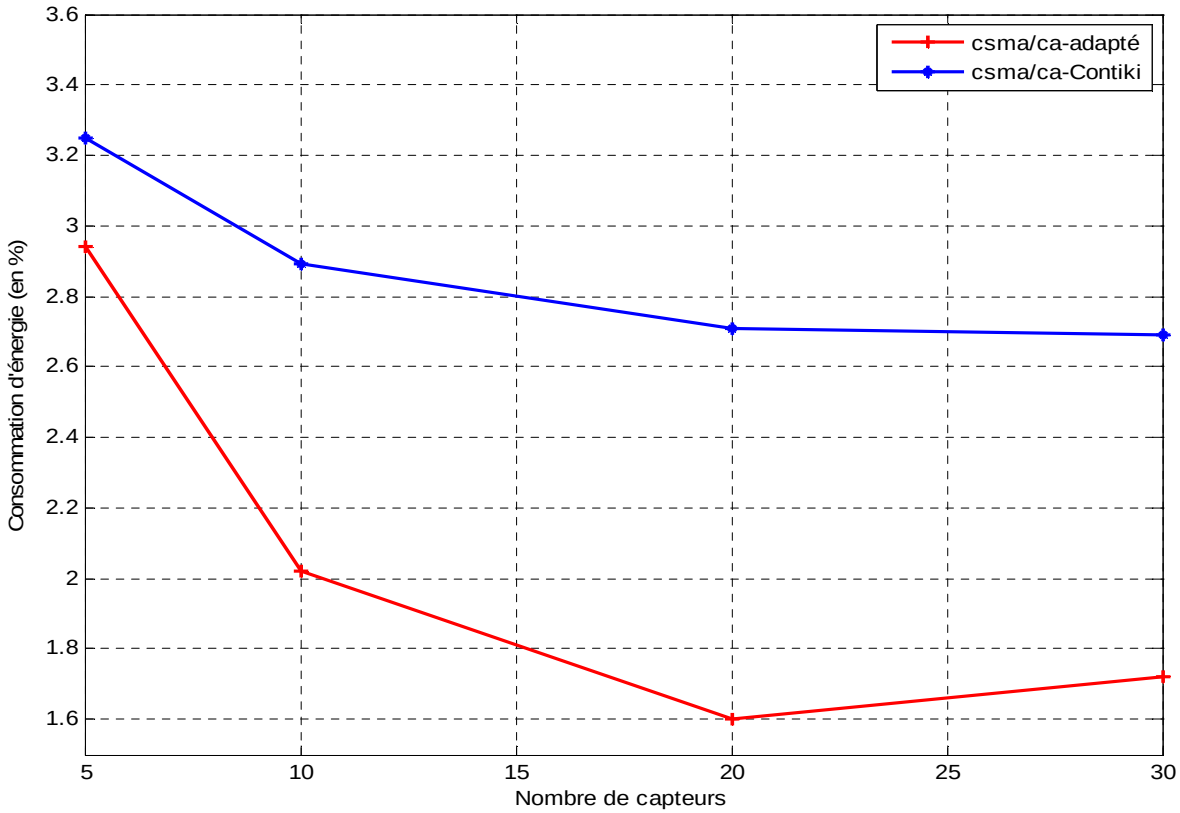


Fig. 4.17: Le niveau de consommation d'énergie par capteur

IX. Déploiement de CSMA/CA-Adapté sur des capteurs réels

Afin d'évaluer les performances de csma/ca-Adapté dans une application temps réel, nous avons déployé la solution sur les capteurs qui seront utilisés pour la localisation indoor. Les capteurs utilisés sont du modèle TelosB déjà introduis dans le premier chapitre.

IX.1 La puissance du signal RSSI

Tout au long des chapitres précédents nous avons parlé de la technique RSSI sans vraiment détailler son fonctionnement car pour comprendre le principe du RSSI il faut d'abord définir quelques concepts importants dans la communication par radiofréquence :

- Une antenne *isotrope* est un modèle théorique, elle rayonne uniformément dans toutes les directions, le diagramme de rayonnement de l'antenne isotrope est une sphère [70].

- Le gain d'une antenne dans une direction donnée est le rapport entre la puissance rayonnée dans cette direction et la puissance rayonnée par une antenne de référence (isotrope) [71]. L'unité de mesure du gain est le *dBi.*
- Si la ligne entre l'émetteur et le récepteur est droite sans aucune rupture alors on dit qu'il y a une ligne de vue ou en anglais Ligne-of-Sight (LOS) entre eux [72].
- La diffraction est l'obstruction de la ligne de vue par des obstacles [72].
- La réfraction est la déviation d'une onde lorsque sa vitesse change entre deux milieux différents par exemple un faisceau de lumière dans l'eau [72].
- Le multipath est lorsque le signal offert au récepteur contient non seulement le signal de la ligne de vue, mais également d'un grand nombre d'ondes radio réfléchies [71].

La technique RSSI se base sur la puissance du signal reçu pour estimer la distance entre les deux points d'émission et de réception, car selon Friis [70] la formule générale pour calculer la puissance du signal reçu dans un espace libre est :

$$\frac{Pr}{Pt} = Gt\ Gr \left(\frac{\lambda}{4\ \pi\ R}\right)^2$$

Où :

- Pr : la puissance du signal reçu en Watt ;
- Pt : la puissance du signal émis en Watt ;
- Gr : gain de l'antenne réceptrice ;
- Gt : gain de l'antenne émettrice ;
- λ : la longueur d'onde, égale à la vitesse de propagation divisée par la fréquence ;
- R : la distance entre l'émetteur et le récepteur en mètres.

Les caractéristiques de l'antenne interne dont sont équipés les capteurs TelosB sont :

- Puissance d'émission 0.0 dBm ;
- Les gains de l'antenne 3.1 dBi ;
- λ = 0.12491 m.

Connaissant *Pt, Pr, Gt, Gr* et λ, on peut estimer les distances qui séparent le mobile des différents capteurs ancres par la relation suivante:

$$R = \frac{\lambda}{4\pi} * \sqrt{\frac{(Pt * Gt * Gr)}{Pr}}$$

IX.2 La multi-latération

Dans cette seconde phase on détermine la position du mobile en se basant sur les distances estimées avec un certain nombre d'ancres. La méthode la plus commune pour calculer une position est la latération qui est une forme de triangulation [73]. Depuis les distances estimées (d_i) et les positions connues (x_i , y_i) des ancres, [73] dérive le système d'équation suivant :

$$(x_1 - x)^2 + (y_1 - y)^2 = {d_1}^2$$

$$(x_2 - x)^2 + (y_2 - y)^2 = {d_2}^2$$

$$\vdots$$

$$(x_n - x)^2 + (y_n - y)^2 = {d_n}^2$$

Où la position inconnue du mobile est donnée par (x, y). Le système peut être linéarisé par la soustraction de la dernière équation des $n - 1$ premières équations :

$${x_1}^2 - {x_n}^2 - 2(x_1 - x_n)\,x + {y_1}^2 - {y_n}^2 - 2(y_1 - y_n)\,y = {d_1}^2 - {d_n}^2$$

$$\vdots$$

$${x_{n-1}}^2 - {x_n}^2 - 2(x_{n-1} - x_n)\,x + {y_{n-1}}^2 - {y_n}^2 - 2(y_{n-1} - y_n)\,y = {d_{n-1}}^2 - {d_n}^2$$

Le réarrangement des termes donne un système approprié des équations linéaires sous la forme A x = b, où :

$$A = \begin{bmatrix} 2(x_1 - x_n) & & 2(y_1 - y_n) \\ & \vdots & \\ 2(x_{n-1} - x_n) & & 2(y_{n-1} - y_n) \end{bmatrix}$$

$$b = \begin{bmatrix} {x_1}^2 - {x_n}^2 + {y_1}^2 - {y_n}^2 + {d_n}^2 - {d_1}^2 \\ \vdots \\ {x_{n-1}}^2 - {x_n}^2 + {y_{n-1}}^2 - {y_n}^2 + {d_n}^2 - {d_{n-1}}^2 \end{bmatrix}$$

Le système est résolu en utilisant la méthode des moindres carrés : x = $(A^T\,A)^{-1}\,A^T\,b$.

IX.3 Application de localisation

Nous avons développé un programme à l'aide de langage de programmation JAVA qui s'exécute sur l'ordinateur et dont le rôle est de récupérer les RSSI reçues par le mobile et de trouver sa position, tout cela en temps réel bien entendu. Le programme offre une interface graphique ou l'environnement de la localisation est représenté par une simple zone de dessin (voir la figure 4.18). Les ancres y sont représentées par les points bleus et le mobile par le point vert.

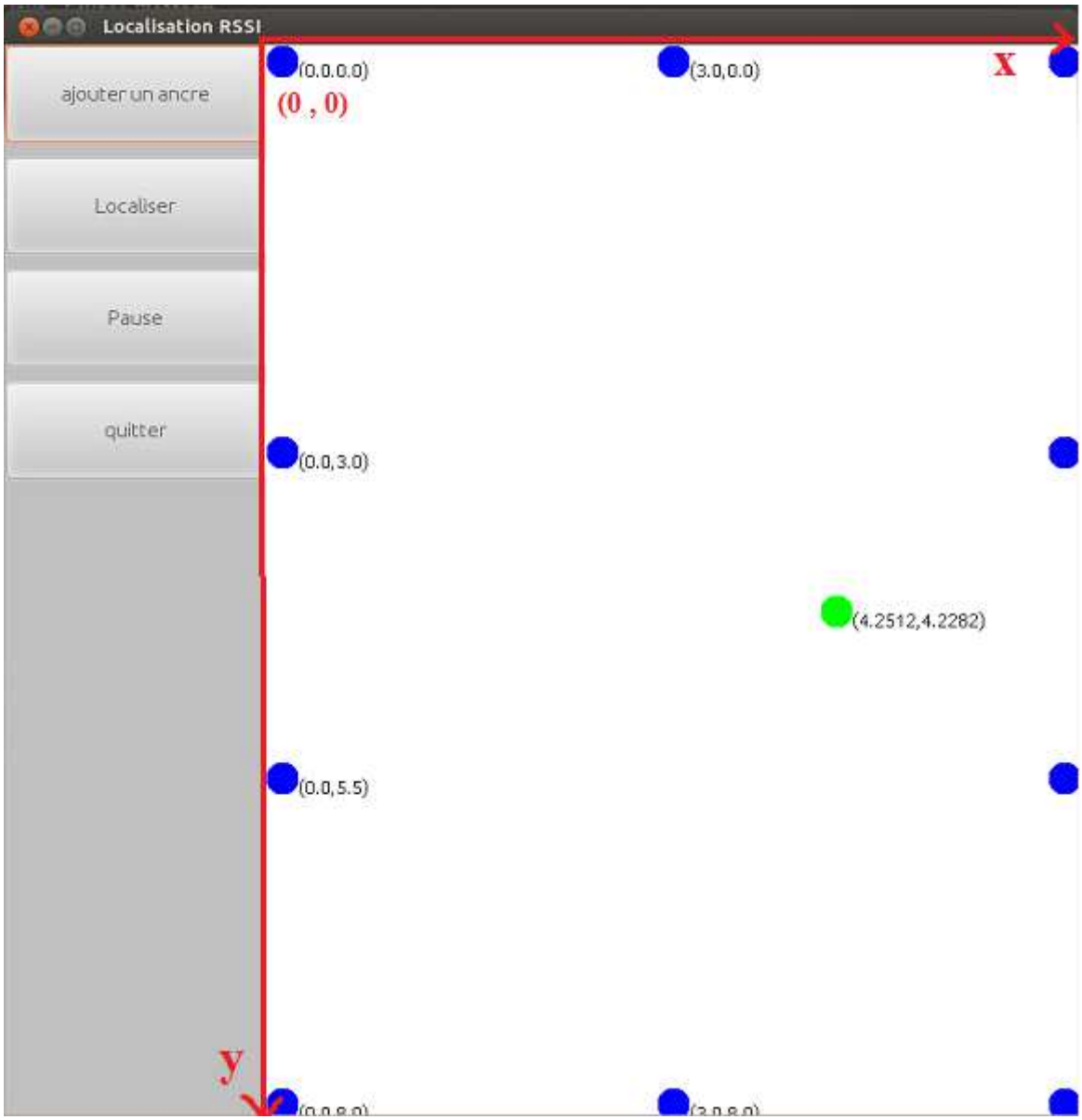


Fig. 4.18: Application de localisation

L'application est également dotée de capacité de stockage, c'est-à-dire, que les positions calculées durant l'exécution sont archivées, grâce à cette fonctionnalité, nous pouvons plus tard consulter ces positions et les comparer avec les positions réelles afin de déduire une marge d'erreur et c'est d'ailleurs ce que nous allons faire tout de suite.

IX.4 Tester CSMA/CA-Adapté dans une application de localisation

Pour donner un sens à ce travail, nous avons voulu tester csma/ca-Adapté en dehors du simulateur, dans l'application de localisation. Pour ce dernier test nous avons utilisé 10 capteurs ancres en étoile autour d'un mobile embarqué sur l'ordinateur (voir la figure 4.18). Nous avons également refait ce même test pour csma/ca-Contiki pour avoir un point de comparaison.

Le test en lui-même consiste en le placement du mobile dans des positions bien connues puis nous faisons calculer sa position à l'application de localisation. Nous déduisant la marge d'erreur par le calcul de la distance euclidienne comme suit :

$$m = \sqrt{(x - x')^2 + (y - y')^2}$$

Où (x, y) est la position réelle et (x', y') est la position calculée. Les marges d'erreur données par csma/ca-Adapté et csma/ca-Contiki sont dans le tableau 4.6 :

Position réelle (en mètre)	Position calculée par csma/ca-Adapté	Marge d'erreur pour csma/ca-Adapté	Position calculée par csma/ca-Contiki	Marge d'erreur pour csma/ca-Contiki
(4.6 , 6.4)	(4.10 , 4.19)	2.26	(2.92 , 4.78)	2.33
(4.6 , 4.0)	(3.86 , 4.52)	0.90	(3.12 , 3.89)	1.48
(4.0 , 1.4)	(3.66 , 3.27)	1.90	(3.25 , 3.86)	2.57
(1.5 , 1.4)	(2.35 , 3.13)	1.93	(3.15 , 4.41)	3.43
(1.5 , 4.0)	(1.59 , 2.95)	1.05	(3.11 , 4.92)	1.85
(1.5 , 6.4)	(3.28 , 6.70)	1.80	(2.57, 4.16)	2.48

Tableau 4.6 **:** Les marges d'erreur dans les positions calculées

Dans la pratique les trois phénomènes que sont la diffraction, la réfraction et le multipath font que la puissance du signal reçu ne reflète pas vraiment les distances réelles qui séparent le mobile des ancres, pour cette raison il est extrêmement difficile d'avoir une bonne précision en

se basant uniquement sur le RSSI d'où les positions dans le tableau 4.6 qui s'éloignes par fois de la vraie position. Néanmoins, nous pouvons voir que csma/ca-Adapté possède une marge d'erreur inférieure à celle de csma/ca-Contiki et cela pour tous les points, afin de visualiser cette différence nous présentons la figure 4.19 qui représente le parcours réel et ceux donnés par les deux csma/ca :

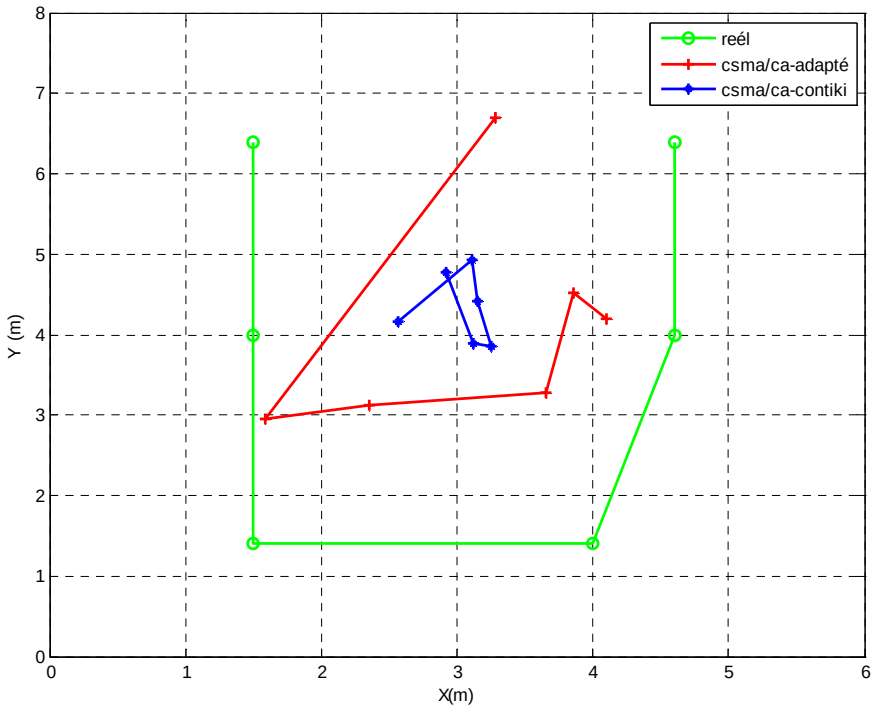


Fig. 4.19: Comparaison des trois parcours

La figure 4.19 montre que lorsque nous avons effectué la localisation avec csma/ca adapté nous nous sommes rapprochés d'avantage du parcours réel que lorsque la localisation était effectuée avec csma/ca-Contiki.

Nous devons cette légère amélioration dans la précision à la réception de plus de RSSI avec csma/ca-Adapté (voir le tableau 4.7) ce qui a donné la possibilité à la multi-latération de calculer la position du mobile avec plus de points de référence.

Par ailleurs, le tableau 4.7 démontre que csma/ca-Adapté est aussi efficace dans le cas réel que dans un simulateur car la solution a su garder des résultats aussi satisfaisants que ceux obtenus dans un simulateur.

CSMA/CA	Nombre de capteurs	Temps de régénération	Moy. des messages en séquence	Moy. des messages en retard	Taux	
					Séquence	Retard
csma/ca-Contiki	1+10	1s	4.32	2.74	43.18%	27.41%
		2s	6.66	2.45	66.60%	24.50%
		2.5	6.51	0.20	65.11%	2.00%
csma/ca-Adapté	1+10	1s	5.64	0.61	56.43 %	6.05 %
		2s	9.05	0.28	90.52%	2.78%
		2.5s	9.34	0.20	93.37%	2.00%

Tableau 4.7 **:** Les résultats des tests dans le cas réel

X. Conclusion

Dans ce chapitre nous avons mis le point sur la problématique que pose le protocole d'accès au support CSMA/CA pour le temps réel à travers une application de localisation. Nous avons souligné le fait que CSMA transmit les paquets même après avoir dépassé le temps alloué par l'application. Pour pallier cette problématique nous avons présenté trois solutions : le coordinateur, réduire le nombre de transmission et l'intervalle de temps. Le Coordinateur n'a pas été à la hauteur de nos attentes, pour cette raison il fût rapidement abandonné. Réduire le nombre de transmission été une solution radicale dans le sens où les paquets sont supprimés très vite, n'exploitant pas la totalité de temps alloué par l'application. La solution de l'intervalle de temps alloué au paquet été la seule à pouvoir résoudre le problème car elle a réussi à éliminer les retardataires donnant ainsi des temps de réponse raisonnables et permet d'exploiter plus de capteurs.

Nous avons déployé la solution de l'intervalle pour une application de localisation indoor, mais elle peut aussi être utilisée pour toute autre application temps réel pour les réseaux de capteurs qui nécessite que les paquets envoyés soit reçus dans un délai bien déterminé ou supprimés, ce délai sera donné à CSMA directement depuis la couche application via la fonction *packetbuf_set_attr()* que offre le système d'exploitation Contiki.

Conclusion générale

Les réseaux de capteurs sans fil (RCSFs) constituent des sujets de recherche innovants pour diverses disciplines des sciences et techniques de l'information et de la communication, mais avec toutefois des contraintes spécifiques s'érigeant en défis. Parmi les problèmes posés dans ce type de réseaux est le problème de localisation.

Certaines applications de localisation imposent des fortes contraintes, comme la certitude de positionnement et ainsi un temps de réponse limité. csma/ca-Contiki ne peut pas satisfaire ces contrainte, car il est adapter pour envoyer un maximum d'informations par capteurs et peu importe le temps occupé. En résultat de ce comportement, il se produit une mauvaise exploitation des capteurs, surcharge du réseau avec des paquets inutiles (informations obsolètes), mauvaise estimation de la position et le gaspillage de l'énergie.

Nous avons proposé dans ce mémoire une adaptation du protocole d'accès au support CSMA/CA pour une application temps réel. Notre travail fait l'objet d'assurer une meilleure exploitation des capteurs en terme de quantité des informations (plus de précision), la validité des messages (éliminer les paquets retardataire) et garantir un temps de réponse souhaité. Pour cela nous avons procédé en trois phases, dont nous avons commencé par la description de la problématique du csma/ca-Contiki, ensuite nous avons développée trois solutions :

- la première : est la conception d'un coordinateur qui gère les réponses des ancres où nous avons enregistré des mauvaises résultats, c'est-à-dire avec ce coordinateur, on ne peut pas satisfaire les contraintes d'un système à temps réel, cela à cause des messages additionnel de dialogue entre le coordinateur et les autres capteurs, ainsi les annonces périodique des ancres qui provoquent des collisions et en plus nous avons créé un paquet (l'autorisation envoyée par le coordinateur aux ancres afin de transmettre) sensible aux collisions, en effet les messages arrivent souvent en retard et le temps imposé n'est pas respecté et pour cela nous avons procédé à d'autres solutions;

- La deuxième solution : nous avons pensé à limiter le nombre de transmission à 1 afin d'éliminer les messages qui seront probablement retardés en supprimant ces derniers dès leurs premier échec de transmission (non réception de l'ACK). En résultat, nous avons éliminé complètement le phénomène des retardataires mais avec un nombre de messages en séquence très réduit à cause du nombre de collisions qui donne une grande probabilité pour que le paquet subisse un échec de transmission, effectivement CSMA supprime ce dernier très tôt, ce qui diminue la précision de positionnement dans le réseau et la mauvaise exploitation des capteurs;

- La troisième solution : consiste à intégrer un contrôle de la validité des paquets par un interval de temps bien déterminé dans le corps du CSMA, c'est-à-dire que CSMA en quelque sorte lance un temporisateur pour chaque paquet prêt à être transmit. Si le capteur a réussi l'envoie de ce paquet avant l'écoulement de ce temporisateur, donc le message arrivera au mobile à temps, sinon le paquet sera supprimé pour qu'il ne soit pas retardé. Pour perfectionner plus la solution, nous avons réduit la taille de la file d'attente des paquets à 1 afin d'éviter complètement les retardataires (les nouveaux messages suppriment les anciens et prennent leur place). Ce qui assure l'élimination des retardataires, une garantie du temps de réponse souhaitable (le temps de réponse est toujours proportionnel au nombre de capteurs), une meilleurs exploitation des capteurs, réduction du nombre de collision en gérant la compétition des capteurs (le temps alloué est divisé en deux partie, la première est réservée pour le tirage aléatoire de l'éveille des capteurs) et ainsi une économie d'énergie.

Enfin, après avoir eu des bons résultats avec la troisième solution sous le simulateur, nous avons terminé par le déploiement de cette solution sur des capteurs réels (10 capteurs) en réalisant une application de localisation où nous avons constaté une différence remarquable entre les résultats de la simulation et de cas réel avec csma/ca-Contiki (9.38 capteurs ont pu rependre en 2.5s dans la simulation contre 6.51 dans le cas réel), mais à notre plus grande surprise, csma/ca-Adapté a donné des résultats convaincants très proches de ceux de la simulation(9.94 capteurs ont pu rependre en 2.5s dans la simulation contre 9.34 dans le cas réel).

Ce travail nous a permis d'acquérir plusieurs connaissances de bases sur le fonctionnement des réseaux de capteurs sans fil en générale et sur le design du système d'exploitation Contiki en particulier. Nous avons également appris à simuler des réseaux de capteurs sous le simulateur *Cooja* et de faire exécuter aux capteurs les applications de notre choix et par la suite le déploiement de ces dernières sur des capteurs réels. Pour avancer dans ce travail nous avons créé des programmes en langage C et JAVA, sans oublier les innombrables notions que nous avons appris sur le monde GNU/Linux.

Une suite logique à ce travail sera de combiner notre solution avec d'autres techniques pour avoir une meilleure précision. Et pourquoi ne pas déduire une relation fiable entre le temps et le nombre de capteurs exploités.

Bibliographie

Référence	Description
[1]	G. Sousa : «*Etude en vue de la réalisation de logiciels bas niveau dédiés aux réseaux de capteurs sans fil : microsystème de fichiers*», Thèse de doctorat, Université Blaise Pascal – Clermont II, 2008.
[2]	G. Asch : «*Acquisition de données du capteur à l'ordinateur* », ISBN : 2 10 006310 3, pages : 315-332, 2003.
[3]	Crossbow Technology, Inc. TPR2420CA IEEE 802.15.4 TelosB Mote with Sensor Suite
[4]	Y. Zatout : «*Conception et évaluation de performances d'un réseau de capteurs sans fil hétérogène pour une application domotique*», Thèse de doctorat, Université Toulouse 2, le Mirail (UT2 Le Mirail), 2011.
[5]	A. Dunkels, B. Gronvall and T. Voigt: «*Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors*», Swedish Institute of Computer Science (SICS), 2004.
[6]	A. Cazes et J. Delacroix : « *Architecture des machines et des systèmes informatiques* », ISBN 978-2-10-053945-1, pages : 266-288, 2008.
[7]	P. Ficheux : «*Linux embarqué* », ISBN : 2-212-11674-8, pages : 18-34, 2005.
[8]	S. Bhatti & J. Carlson & H. Dai & J. Deng & J. Rose & A. Sheth & B. Shucker & C. Gruenwald & A. Torgerson and R. Han : «*MANTIS OS: An Embedded Multithreaded Operating System for Wireless MicroSensor Platforms*», University of Colorado at Boulder, 2005.
[9]	P. Levis & S. Madden & J. Polastre & R. Szewczyk & K.Whitehouse, A. Woo & D. Gay & J. Hill & M. Welsh & E. Brewer and D. Culler: «*TinyOS: An Operating System for Sensor Networks*», University of California Berkeley, 2004.
[10]	L. Iannone & F. Benbadis & M. Dias de Amorim and S. Fdida : «*Some applications of Wireless Sensor Network* », LIP6 /CNRS Université Pierre et Marie Curie, 2004.
[11]	P. Honeine & C. Richard & H. Snoussi and M. Essoloh : «*Auto-localisation dans les réseaux de capteurs sans fil par régression de matrices de Gram*», Laboratoire LM2S, Institut Charles Delaunay (FRE CNRS 2848) Université de technologie de Troyes, 2009.
[12]	I. Stojmenovic : «*Handbook of Sensor Networks: Algorithms and Architectures*», Wiley Series on Parallel and Distributed computing, 2005.
[13]	D. Dhoutaut: « *Etude du standard IEEE802.11 dans le cadre des réseaux ad hoc : de la simulation à l'expérimentation* », Thèse de Doctorat, Institut National des Sciences Appliquées de Lyon, 2003.

[14] M. Toumi «*Evaluation des performances du réseau IEEE802.11 mode ad hoc avec fragmentation*», Thèse de magistère, Département d'informatique, Université de Bejaia, 2006.

[15] «*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Technical report* », IEEE Std 802.11-1997.

[16] «*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-speed physical layer extension in the 2.4GHz band*». Technical report, IEEE Std 802.11b-1999.

[17] «*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-speed physical layer extension in the 5GHz ban* ». Technical report, IEEE Std 802.11a. 2001.

[18] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 2002.

[19] M. Duchataut : «*Analyse et simulation du déploiement d'un réseau sans fil à l'ULB*», Mémoire de fin d'études, Université Libre de Bruxelles, 2005.

[20] CERT. «*Wireless Fidelity*», république tunisienne, ministère des technologies de la communication, Centre d études et de recherche des télécommunications. 2005.

[21] N. Tibi : «*Etude des mécanismes de différentiation de service niveau IP et Mac*», Rapport de projet de fin d'étude, Ecole supérieures des communications de Tunisie. 2004.

[22] A. Ksentini : «*qualité de service dans les réseaux locaux sans fil basés sur la technologie IEEE 802.11*», Thèse de Doctorat ,Université de CERGY-PONTOISE, France. 2005.

[23] A. VAN DEN BOSSCHE : «*Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles* », Thèse de Doctorat, Université de Toulouse II, Ecole Doctorale Systèmes, France. 2007.

[24] M. Terré : «*Le Standard 802.11, Couche physique et couche MAC*»; Mars 2007 en ligne : http://stic.cnam.fr/elau/publi/terre/images/WiFi.pdf.

[25] http://fr.wikipedia.org/wiki/IEEE_802 ; consulté le 04/2013.

[26] http://fr.wikipedia.org/wiki/Contrôle_de_la_liaison_logique ; consulté le 04/2013.

[27] http://fr.wikipedia.org/wiki/Carrier_Sense_Multiple_Access_with_Collision_Detection ; consulté le 04/2013.

[28] http://fr.wikipedia.org/wiki/IEEE_802.12 ; consulté le 04/2013.

[29] K. heurtefeux & F. Valois : «*protocoles d'auto-organisation : une étude qualitative au cours de la vie des réseaux de capteur* », ARES INRIA/CITI, INSA-Lyon, F-69621, France, 2006.

[30] E. Ermel : «*Localisation et Routage géographique dans les réseaux sans fil hétérogènes*», Thèse de Doctorat, Université Paris VI Piere et marie CURIE juin 2004.

[31] Projet OCELOT. «*Objets communicants-expérimentations technologiques et spécifications* », Technical report, ICP-INPG, Grenoble, April 2006.

[33] J. Werb & C. Lanzl : «*A positioning system for finding things indoors* », IEEE Spectrum, vol. 35, n°9, pages: 71-78, 1998.

[34] S. Capkun & M. Hamdi and J.P UBAUX : «*GPS-free Positioning in Mobile Ad-Hoc Networks*», Cluster Computing, vol. 5, n° 2, 2002.

[35] N. Douma & M. El Hammouti : «*La localisation dans un réseau de capteurs* », Université d'Avignon, France, 2008.

[36] A. Küpper : «*Location-based services: Fundamentals and Operation*», Chichester: John Wiley & Sons, 2005.

[37] P. Bahl & V. N. Padmanabhan : «*Radar : An inbuilding rf-based user location and tracking system*», INFOCOM, l'Université Charles de Gaulle, Lille 3, France, 2000.

[38] A. Srinivasan & J. Wu : «*A Survey on Secure Localization in Wireless Sensor Networks*», Encyclopedia of Wireless and Mobile Comm., CRC Press, Taylor and Francis Group, 2007.

[39] D. Bucur : «*Location Sensing in Ubiquitous Computing*» , presentation for the Activity Based Computing group at DAIMI, Department of Computer Science, DAIMI Faculty of Science, University of Aarhus April, 2006.

[40] A. de Keijzer : «*Localization in Ad Hoc Sensor Network*», EYES Project, 14th April 2003.

[41] Y. Darcilon : «*Localisation des nœuds dans les réseaux de capteurs sans fil*», Rapport de projet master 2, Université d'Avignon, France 2007/2008.

[42] K. heutefeux & F. Valois : «*Localisation collaborative pour réseaux de capteurs*» ARES INRIA / CITI,INSA- Lyon, France, 2007.

[43] D. Niculescu & B. Nath : «*Ad hoc positioning system (aps)*»,in Proceedings of GLOBECOM, San Antonio,2001.

[44] D. Niculescu & B. Nath : «*Ad hoc positioning system (aps) using aoa*», IEEE INFOCOM, San Francisco, CA.2003.

[45] M. McGuire & K. N. Plataniotis and A. N. Venetsanapoulos : «*Location of mobile terminal using time measurements and servey points*». IEEE Trans. Vehicular Technology, vol. 52, no. 4, pages: 999-1011, July 2003.

[46] A. H. Sayed & A. Taroghat and N. Khajehnouri : «*Network-based wireless location*». IEEE signal Processing Magazine, vol. 22, no. 4, pages : 24-40, July 2005.

[47] L. Cong & W. Zhuang : «*Non-line-of-sight error mitigation in mobile location*», IEEE Transactions on Wireless Communications, vol. 4, pages: 560-573, March 2005.

[48] J. J. Caffery : «*Wireless location in CDMA cellular radio systems*». Boston: Kluwer Academic Publishers, 2000.

[49] L. Cong & W. Zhuang : «*Hybrid TDOA/AOA mobile user location for wide-band cdma cellular systems*». IEEE Transactions on Wireless Communications, vol. 1, pages:.439 -447, July 2005.

[50] R. I. Reza : «*Data fusion for improved TOA/TDOA position determination in wireless systems*». Ph.D. Dissertation, Virginia Tech, 2000.

[51] S. Gezici : «*A Survey on Wireless Position Estimation*», Université Bilkent, Ankara, Turquie, 2007.

[52] V. Mathieu «*Réseaux de senseurs sans fil : Problème de localisation*», Université Libre de Bruxelles. France. 2007.

[53] N. Bulusu : «*Self- con_guring Location Systems*» PhD thesis, University of California, 2002.

[54] N. Bulusu & J. heidemann and D. Estrin : «*GPS-less-low-cost outdoor localization for very small devices* ». Personal Communication, IEEE, pages 28-34, 2000.

[55] C. Saad «*Quelques contributions dans les réseaux de capteurs sans fil : Localisation et Routage*», Thèse de Doctorat, Université d'Avignon et des pays Vaucluse, Juillet 2008.

[56] K. langendoen & N. Reijers : «*Distributed localization in wireless sensor networks: a quantitative comparison*». Computer Networks, pages : 499_518, 2003.

[57] S. Simic & S. Sastry : «*A distributed algorithm for localization in random wireless networks*» Discrete Applied mathematics, 2003.

[58] D. Nicolescu & B. Nath : «*Ad hoc Positioning Systems (APS)*». Proceedings of the GLOBECOM, IEEE International Global Telecommunications Conference, San Antonio, TX, USA., pages : 2926-2931, 2001.

[59] IEEE Computer Society : « *Token Bus Access Method and Physical Layer Specifications*», IEEE Std 802.4, 1985.

[60] IEEE Computer Society : «*Token Ring Access Method and Physical Layer Specification*», IEEE Std 802.5, 1985.

[61] IEEE Computer Society : «*Distributed Queue Dual Bus (DQDB) Subnetwork of Metropolitan Area Network*», IEEE Std 802.6, 1990.

[62] P. Muhlethaler «*802.11 et les réseaux sans fil* », Eyrolles, 2002.

[63] A. Dunkels : «*The ContikiMAC Radio Duty Cycling Protocol*», Swedish Institute of Computer Sciences, 2011.

[64] W. DU : «*Modélisation et Simulation de Réseaux de Capteurs sans* Fil », Thèse de doctorat, Institut des Nanotechnologies de Lyon, 2011.

[65] CCNA Exploration 4.0 : «*Communication de réseau local et réseau local sans fil* », Chapitre 7 : concepts et configuration de base d'un réseau sans fil, 2008.

[66] X. Yong & A. Andres & G. Andres et B. Mickaël : « *Agrégation de données dans les réseaux de capteurs* », Rapport final, Université de Technologie Compiègne, 2010.

[67] N. Merrani et N. Khimoum : « *Simulation et Evaluation de protocoles de gestion de clés dans les réseaux de capteurs* », Mémoire de fin de cycle, Université A. Mira de Bejaïa, 2009.

[68] A. Mallikarjuna & D. Janakiram and G. Kumar : «*Operating Systems for Wireless Sensor Networks: A Survey Technical Report*», Indian Institute of Technology Madras, 2007.

[69] http://wiki.contiki-os.org/doku.php?id=an_introduction_to_cooja ; consulté le : 03/2013.

[70] http://fr.wikipedia.org/wiki/Antenne-isotrope ; consulté le 06/2013.

[71] http://people.seas.harvard.edu/~jones/es151/prop_models/propagation.html ; consulté le : 06/2013

[72] M. Naili : «*Gestion de contexte dans l'habitat communicant : Localisation indoor et fourniture de services sensibles au contexte par ontologie* », Mémoire de magistère, Université Abderrahmane Mira, Bejaia, 2012.

[73] K. Langendoen and N. Reijers : «*Distributed Localization Algorithms*», Delft University of Technology , 2004.

Annexe A

La première fois que nous avons entendu notre encadreur nous dire la phrase : « installer Contiki sous Linux », un petit silence a régné sur la discussion en se demandant comment allons-nous installer un système d'exploitation sur un autre, s'agit-il une cohabitation de 2 systèmes comme nous avons l'habitude de faire avec Windows? Mais Contiki est bien un système d'exploitation pour les capteurs sans fil et l'installer en cohabitation avec Linux, sous-entend que nous allons utiliser notre ordinateur comme un capteur! Cette confusion a duré une petite semaine, et à force de chercher nous avons enfin compris la subtilité.

Bien que nous ayons vu sur internet des personnes qui s'amusent à installer Contiki sur des anciens ordinateurs comme APPELE II afin de l'exécuter directement sur les ressources de ce dernier, pour programmer des capteurs qui exécutent Contiki ce n'est pas la chose à faire. En fait ce n'est pas une installation au sens propre du terme mais il s'agit plutôt de préparer un environnement Contiki sous linux. Cette annexe montre comment obtenir l'environnement en question étape par étape sous *unbuntu* GNU/Linux 12.04 LTS. Notons tout de même qu'il existe un environnement Contiki préinstallé appelé *InstantContiki* prêt à l'emploi, il suffit de l'exécuter à travers un logiciel de virtualisation (permet de créer des machines virtuelles) comme VMware par exemple, le seul inconvénient de cette approche est la lenteur de la machine virtuelle. Dans notre cas, pour exploiter la pleine puissance de nos CPU, nous préférons équiper Linux des outils nécessaires pour manipuler Contiki que de passer par la virtualisation.

Etape 1 : installer les compilateurs et les bibliothèques

Pour les capteurs on trouve principalement deux grandes familles de microcontrôleurs :

1. *msp430* de Texas Instrements Inc. ;
2. *avr* d'Atmel.

Nous allons installer les boites à outils de ces deux processeurs, il s'agit principalement des bibliothèques et de leur version du compilateur *gcc*, pour se faire, tapons les commandes suivantes dans le terminal :

```
sudo apt-get install binutils-msp430 gcc-msp430 msp430-libc
```

```
sudo apt-get install gcc-avr binutils-avr gdb-avr avr-libc avrdude
```

Etape 2 : installer les outils java

Les distributions récentes de *ubuntu* n'incluent ni *OpenJDK* (l'implémentation de Java Development Kit) ni la *OpenJRE* (la machine virtuelle) or que le simulateur *Cooja* qui est écrit en java a besoin de ces deux derniers pour fonctionner, il faut donc les installer avec la commande suivante :

```
sudo apt-get install openjdk-7-jdk openjdk-7-jre
```

Après l'installation de java nous devons éditer la variable d'environnement *path*, une manière de faire sous *ubuntu* est d'éditer le fichier *.bashrc* qui se trouve dans le répertoire personnel et lui ajouter les lignes suivantes à la fin:

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export PATH=${PATH}:${JAVA_HOME}/bin
```

Etape 2 : installer le programme *ant*

ant est un programme qui ressemble à *make* pour compiler du code java, dans notre cas il sera utilisé pour compiler le simulateur *Cooja* en ligne de commande. L'installation de *ant* est simple avec la commande suivante :

```
sudo apt-get install ant
```

Etape 3 : télécharger le code source de Contiki

A cette étape, nous avons installé suffisamment d'outils pour exécuter Contiki, mais on doit d'abord le télécharger en allant sur le site officiel de Contiki, *http://www.contiki-os.org.*

Etape 4 : tester le fonctionnement de Contiki

Nous décompressons le fichier téléchargé dans le répertoire personnel et nous nous rendons au sous-dossier *hello-world* dans l'arborescence avec la commande *cd* puis nous compilons Contiki avec *make* comme suit :

```
user@machine : ~$ cd ./contki-2.6/examples/hello-world

user@machine : ~/contki-2.6/examples/hello-world $make TARGET=native hello-world
```

La première compilation dure quelques dizaines de second car c'est tout le système Contiki qui va être compilé. Si la compilation s'est déroulé sans erreurs ; un fichier exécutable portant le nom *hello-world.natvie* est créé, nous exécutons ce fichier à partir du terminal :

```
user@machine : ~/contki-2.6/examples/hello-world $./hellow-world.native
```

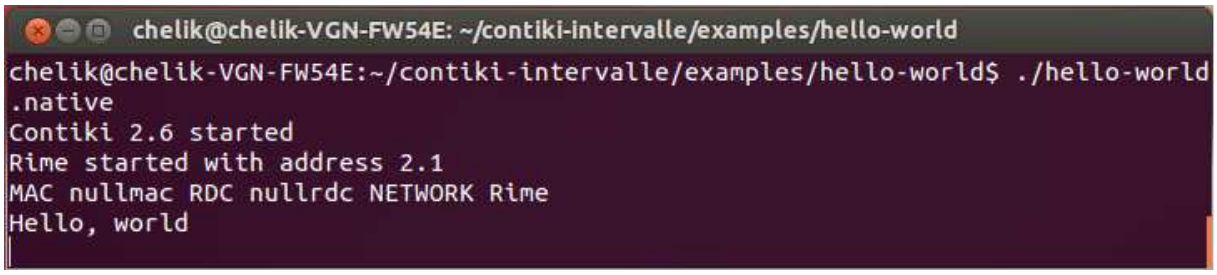


Fig a.1 : l'exécution de Contiki

Pour quitter, tapant <Ctrl> c.

Etape 5: lancer le simulateur Cooja

Pour exécuter Cooja il faut se déplacer dans le sous répertoire *cooja* dans l'arborescence de Contiki et exécuter la commande *ant run* :

```
user@machine : ~$ cd contki-2.6/tools/cooja

user@machine : ~/contki-2.6/tools/cooja $ant run
```

Enfin, nous sommes prêts à simuler des réseaux de capteurs qui exécutent Contiki et bénéficier de tous les outils offerts par *Cooja*.

Résumé

Les réseaux de capteurs sans fil (RCSFs) sont une nouvelle technologie, qui a surgis après les grand progrès technologiques concernant le développement des nœuds capteurs, des processeurs puissants et des protocoles de communication sans fil. La localisation est devenue un concept intéressant et important dans plusieurs applications, dont certaines de ces applications imposent une forte contrainte temporelle que doit assurer le réseau de capteurs en plus de l'exactitude des résultats. Ce mémoire présente une adaptation du protocole d'accès au support CSMA/CA, utilisé par le système d'exploitation des capteurs sans fil Contiki 2.6, dans l'objectif de développer une application de localisations en temps réel, qui exploite le maximum possible de capteurs et permet de gagner en consommation de l'énergie.

Mots clés: Localisation en temps réel, Multi-latération, Réseaux de capteurs sans fil, CSMA/CA, Technique RSSI, Consommation d'énergie.

Abstract

The wireless sensor networks (WSN) are a new technology that is emerged after a major technological advance on the development of sensors, powerful processors and wireless communication protocols. The location has become an interesting and important concept in many applications; some of these applications require a high temporal constraint that the sensor network must provide in addition to the accuracy of the results. This paper presents an adaptation of the medium access protocol CSMA/CA used by the operating system of wireless sensors Contiki 2.6, with the aim to develop an application of location real time witch exploits the maximum possible sensors and saving in energy consumption.

Keywords: Real-time location, Multilateration, Wireless sensor networks, CSMA/CA, RSSI technical, Energy consumption.