Année 09-10

OPTIMISATION MAC DANS UN RÉSEAU BAN MULTI-SERVICES

Rapport de Projet de Fin d'Études

Juin 2010

Javier CUADRADO LÓPEZ

Encadrant: Claire GOURSAUD
Relecteur: Omar GAOUAR
Président: Antoine FRABOULET

Laboratoire CITI – Département Télécommunications, Services & Usages
INSA Lyon

Remerciements

Avant de commencer je voudrais remercier à toutes les personnes qui ont permit que ce PFE soit une réalité.

Ma famille qui a été toujours en train de me soutenir dans toutes mes études et le PFE. En particulier, je voudrais consacrer ce projet à mon grand-père Leonardo qui est décédé il y a un an et il s'est toujours intéressé à mes études et il a été en train de m'encourager à les finir.

Dans le laboratoire CITI je remercie tout particulièrement :

- Claire GOURSAUD pour m'avoir offert ce PFE et avoir partagé toutes ses connaissances. Je la remercie aussi pour avoir suivi mon travaille en tant que tuteure de PFE.

- Cédric LEVY-BENCHETON et Anya APAVATJRUT pour m'avoir aidé sur le problème de l'énergie des capteurs.

- Paul FERRAND pour m'avoir aidé sur les problèmes de propagation et surtout le shadowing et le fading.

Plus généralement, je remercie tous les collègues avec qui j'ai travaillé au sein du laboratoire CITI. Citer les noms serait trop exhaustif mais je remercie vraiment toutes ces personnes pour avoir été ouvertes à mes questions.

Et finalement, je ne peux pas m'oublier de tous les amis que j'ai connus pendant cette année incroyable d'Erasmus. Je ne vais pas vous oublier et maintenant je dois vous dire que c'est seulement un « à bientôt ».

1. Introduction **4**

2. Contexte du projet **5**

2.1. Body Area Networks (BAN) **5**

2.2. Couche MAC **6**

2.2.1. Définition et Structure 6

2.2.2. Trames beaconnés et non-beaconnés 8

2.2.3. Relai 9

2.2.4. Retransmission 11

2.2.5. Implémentation machine à états 12

2.3. WSNet **13**

3. Performances de WSNet **16**

4. Simulations **18**

4.1. Scénario 1. Trois nœuds (Emetteur-Relai-Recepteur) **18**

4.1.1. Analyse de la PER 18

4.1.2. Analyse de l'énergie dépensée 20

4.2. Scénario 2. Trois nœuds (Deux émetteurs/relais-Recepteur) **22**

4.2.1. Analyse de la PER 22

4.2.2. Analyse de l'énergie dépensée 23

4.3. Scénario 3. Neuf nœuds (Huit émetteurs/relais-Recepteur) **24**

4.3.1. Analyse de la PER 24

4.3.2. Analyse de l'énergie dépensée 28

5. Shadowing et Fading **29**

6. Conclusion **31**

Bibliographie **32**

Annexe I **33**

Annexe II **39**

1. Introduction

J'ai réalisé mon Projet de Fin d'Études au sein du laboratoire CITI au département Télécommunications, Services & Usages à l'INSA de Lyon et sous l'encadrement de Mme Claire GOURSAUD.
L'objectif de mon projet était l'optimisation de la couche MAC dans un réseau de capteurs BAN (Body Area Network) en termes de probabilité d'erreur et énergie consommée.

Avant sa mise en place, le déploiement d'un réseau de capteurs a besoin d'une phase de simulations afin de s'assurer du bon fonctionnement de tous les dispositifs. Cela c'est nécessaire à cause du coût important du déploiement d'un tel réseau. Dans cette phase là il y a beaucoup de chercheurs qui interviennent. Ils simulent les différents protocoles qui vont être utilisés par les capteurs, les différents modèles de propagation, différents scénarios… Tout ça, comme j'ai déjà dit, pour s'assurer du bon fonctionnement de tous les dispositifs avant de passer à la phase de tests où tous ces protocoles vont être implémentés sur les capteurs.

Dans mon cas particulier, mon PFE avait comme cible l'optimisation de la couche MAC dans un réseau BAN. Ce que j'ai fait c'est développer dans WSNet les différents algorithmes de relayage envisagés afin de pouvoir les simuler et obtenir des résultats en termes de PER (probabilité d'erreur) et énergie dépensée. Comme on verra à travers ce rapport on a simulé trois scénarios différents.
WSNet a déjà des modules de couches MAC inclus, toutefois j'ai dû en développer des nouveaux afin d'ajouter les nouvelles caractéristiques des protocoles qu'on utilise.

À partir de maintenant je vais présenter un peu le contexte dans lequel j'ai travaillé, je m'attacherai à décrire les divers aspects du travail effectué lors de ce PFE et je donnerai une conclusion finale.

2. Contexte du projet

2.1. Body Area Networks (BAN)

Body Area Networks (BAN) c'est aujourd'hui un acronyme très connu qui fait référence aux réseaux de capteurs sans fils intégrés dans le corps humain (soit implantés dans le corps, soit portés par la personne).

Les BANs sont composés d'un certain nombre de capteurs mobiles et compacts inters communiqués qui surveillent les paramètres vitaux du corps ainsi que mouvements. Ces capteurs capturent information en terme de données numériques et les transmettent jusqu'à une station de base ou nœud coordinateur qui sera le capteur qui traitera, prendra décisions, alertera, enregistrera ou enverra cette information en temps réel jusqu'à un hôpital, une clinique, un docteur…

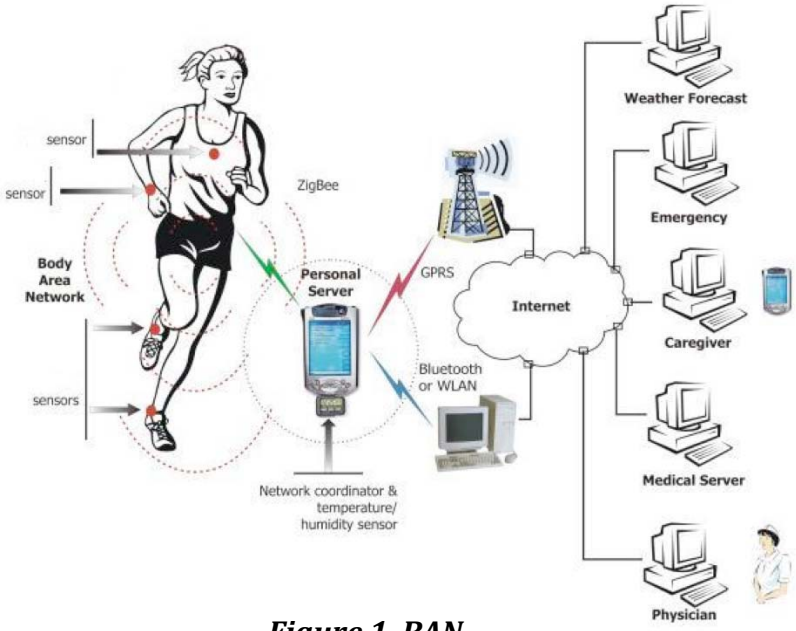

Figure 1. BAN

Favoriser les communications efficaces dans le corps humain ainsi qu'entre eux est une vieille idée. On espère que les Body Area Networks seront déployés dans beaucoup d'applications afin d'instrumenter et d'aider les humains dans beaucoup d'aspects de leur vie.

Il y a beaucoup d'applications qui utilisent les BANs comme par exemple dans le sport, hôpitaux ou systèmes de loisirs qui font définitivement de la définition d'un protocole standard un défi très stimulant.
Néanmoins, c'est dans le domaine médical que nous avons les prévisions les plus grandes sur ces types de réseaux. La raison c'est parce qu'ils sont très utiles pour surveiller et noter depuis n'importe où les paramètres vitaux des patients qui souffrent des maladies chroniques comme l'asthme, le diabète, les crises cardiaques…

Ce type de dispositifs et réseaux aura un rôle de plus en plus important dans la société et est de plus en plus rigoureux en termes de qualité de service (QoS), les questions de coexistence seront décisives. En fait, la ressource radio est connue pour être insuffisante. Les difficultés récentes dans la régulation de *Ultra Wide Bands (UWB)* montrent que chaque nouveau système doit faire un emploi efficace du spectre.
Cette chose là est appliquée aussi aux systèmes BAN personnels à faible portée, qui auront, peut-être, un immense taux de pénétration du marché.

Des normes différentes rivalisent pour répondre aux contraintes qui sont associées avec les possibles applications. Toutefois, chaque solution répond aux spécifications d'une seule application dans le domaine médical, sportif ou de loisirs. À l'avenir, les communications *on body* devraient avoir un standard qui gère tous ces communications. Dans ce contexte il faut définir un seul standard qui respecte des contraintes QoS très rigoureuses mais aussi très adaptables qui seront dépendantes de l'application. Les équipements BAN doivent aussi faire face aux problèmes de consommation d'énergie (spécialement pour les dispositifs implantés). Du coup, le paradigme d'un BAN peut être défini comme : *« offrir des communications fiables, avec un prix énergétique très faible et avec une QoS variable depuis la transmission des alarmes jusqu'à des flux d'information à haut débit et temps réel ».*

Avant de commencer à considérer ce genre de réseaux à grande échelle nous devons considérer quelques aspects très importants:

- ✓ ***Interopérabilité:*** Les systèmes WBAN devront garantir sans problèmes le transfert des données par des normes telles que Bluetooth ou Wi-Fi afin de favoriser l'échange des données. En outre, il y aura un besoin pour l'interconnexion de ce genre de réseaux.

- ✓ ***Système des dispositifs:*** Les capteurs qui seront utilisés devront être petits, avec une complexité faible, légers, faciles à utiliser, reconfigurables… ils auront aussi besoin d'une mémoire pour garder l'information.

- ✓ ***Niveau de sécurité des dispositifs :*** Afin de garantir la confidentialité des données, ce système aura besoin d'un important système de sécurité. L'accès à ces données sera seulement possible pour les médecins qui seront en train de traiter les patients.

2.2. Couche MAC

2.2.1. Définition et Structure

Notre modèle BAN est composé de trois couches d'une façon similaire à celle du modèle OSI. Le standard pour la couche physique et le contrôle d'accès au support (*Media Access Control MAC*) qui sera utilisé sera celui du ***IEEE 802.15.3.***
La couche qui sera la première et sera aussi placée au plus bas niveau (base de la

tour de couches) sera la couche physique qui décrit les technologies de transmission basiques dans la partie matérielle du réseau.
Dans ce cas-ci elle définira le maximum et minimum du débit binaire. Ils seront 11-55 Mbps et aussi la taille maximale et minimale des donnés qui sera entre 64 et 2044 octets. Elle défini aussi la fréquence de transmission qui sera de 2.4GHz.

Après, comme j'ai déjà dit, nous aurons la couche MAC que nous verrons dans ce chapitre (fondamentalement elle contrôle l'accès au support pour chaque nœud) et, finalement, nous aurons la couche application qui dira quand on a des flux d'information à envoyer et quel type d'information nous avons…
Le Contrôle d'accès au support (*Media Access Control* en anglais ou MAC) est une sous-couche, selon les standards de réseaux informatiques IEEE 802.*x*, de la partie inférieure de la couche de liaison de données dans le modèle OSI. Elle sert d'interface entre la partie logicielle contrôlant la liaison d'un nœud (Contrôle de la liaison logique) et la couche physique (matérielle). Par conséquent, elle est différente selon le type de média physique utilisé (Ethernet, Token Ring, WLAN…). Les rôles principaux de cette sous-couche sont à *insérer les addresses MAC de chaque nœud* qui transmet dans la supertrame (dans notre cas elle insérera le numéro de chaque nœud qui envoie ou reçoit les données) et *contrôler l'accès au média physique lorsque celui-ci est partagé*. De cette façon les collisions entre paquets envoyés par tous le nœuds qui essaient d'accéder au réseau au même temps sont évitées.

Pour le standard IEEE 802.15.3 les supertrames de la couche MAC auront le schéma suivant :

- ***Entête:*** L'entête de la couche MAC correspond à 20 octets, auquel s'ajoute un « Frame check sequence » de 4 octets.

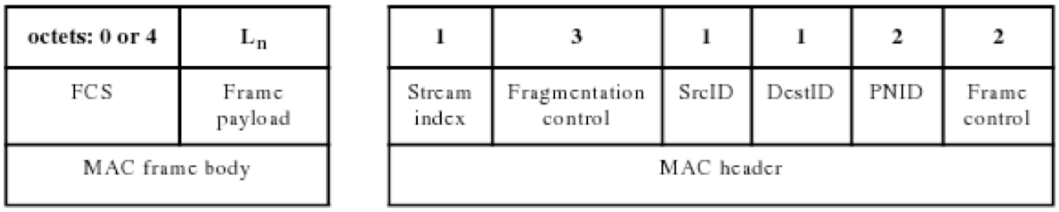

Figure 2. Schéma de l'entête. Source IEEE 802.15.3 page 103

- ***Supertrame:*** La supertrame correspond à un découpage temporel en slot permettant d'organiser la communication entre les différents nœuds du réseau. Cette supertrame est définie par le coordinateur et elle est découpée en trois phases :

  • ***Beacon***: paquet envoyé par le coordinateur ou PNC (Piconet Node Coordinator) pour informer les autres nœuds du réseau de la composition de la supertrame et de la position des différentes phases. Il a un rôle de synchronisation.

  • ***Contention Access Period (CAP)***: accès disponible pour tous les nœuds du réseau grâce à un mécanisme d'accès par contention

CSMA/CA. Le CAP est utilisé par les nœuds pour une association request/response (demande/réponse), channel time request/response mais des envois de données sont également possibles.
C'est pour ça qu'on verra plus tard dans ce rapport un mode de relayage où les données sont envoyées dans cette partie de la supertrame.

• ***Contention Free Period (CFP)***: mécanisme d'accès sans contention par des slots réservés à des flux particuliers (d'un nœud à un autre). Chaque réservation de flux (appelée *Channel Time Allocation* CTA) permet de définir une taille de slots variables.
Le CFP peut être composé de deux types de CTAs, *Guaranteed Time Slots* (GTS) et *Management Time Slots* (MTS) avec une structure de trame TDMA. Les slots MTS sont utilisés pour porter l'information de gestion depuis les esclaves jusqu'au PNC. Les slots GTS sont utilisés par les esclaves pour les communications de données synchrones et asynchrones. Nous utiliserons les derniers.
Quand un nœud réserve un slot avec un protocole *Channel Time Request/Response*, il reçoit un slot dedans le CFP qui lui permet d'envoyer l'information dans la période de temps que dure ce slot.

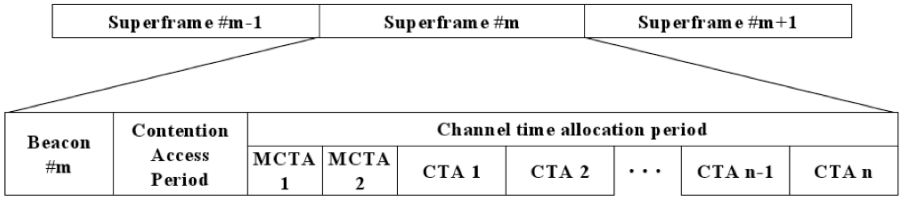

Figure 3. Schema de la supertrame. Source IEEE 802.15.3 page 170

2.2.2. Trames beaconnés et non-beaconnés

C'est important de se rendre compte que dans quelques protocoles MAC il y a deux types de schémas, un schéma qui a une partie de la supertrame appelée beacon et un autre schéma qui ne l'a pas.
Nous pouvons trouver un exemple dans le protocole IEEE 802.15.4. Ce protocole est utilisé par exemple en ZigBee (utilisé pour communications radio à courte portée comme par exemple le Bluetooth).
Dans ce protocole il y a un mode beacon et un mode non-beacon.

Dans le mode beacon c'est le coordinateur qui va être le nœud qui va envoyer à intervalles prédéterminés le beacon (comme dans le IEEE 802.15.3) pour confirmer sa présence aux autres nœuds. De cette manière les nœuds vont se réveiller seulement si le beacon est diffusé, ces nœuds dont l'adresse est indiquée dans le beacon seront maintenus réveillés alors que les autres dormiront. C'est une façon, par exemple, d'allonger la durée de la batterie.

Dans le mode non-beaconné nous n'avons pas de beacon envoyé par le coordinateur, donc un mécanisme de contrôle d'accès au support CSMA/CA sans slots sera utilisé. Dans ce mode, les nœuds doivent être réveillés tout le temps pour savoir s'ils doivent maintenir ou pas un transfert de données avec le coordinateur.

Évidemment dans ce cas nous aurons une dépense énergétique significative, qui dans un réseau de capteurs sans fil avec sa propre batterie est très importante.
C'est pour ça que l'utilisation d'une trame beacon dans la supertrame est recommandée.

2.2.3. Relai

Le relayage c'est une technique utilisée lorsqu'un paquet envoyé par un nœud émetteur n'arrive pas au nœud destinataire. Ce qu'on fait c'est envoyer le paquet à un autre nœud qui va être le relai et qui enverra ce paquet au nœud destinataire si ce nœud destinataire n'ait pas réussi à recevoir le paquet envoyé depuis l'émetteur (donc l'acquittement n'est pas envoyé par le nœud destinataire).
On va voir maintenant différents mécanismes de relai qu'on va implémenter et simuler.

Comme on a déjà dit, la structure d'une supertrame dans le protocole 802.15.3 est divisée en quatre parties principales :

- Beacon
- CAP
- CFP
- Période Inactive (c'est une période où tous le nœuds sont en train de dormir jusqu'au début de la supertrame suivante).

La partie du beacon est réservée pour porter information des CTAs (*Channel Time Allocations*) et elle est utilisée par le coordinateur, donc seulement la CAP, le CFP et le période inactive peuvent être utilisés pour émettre les données relayés.

On va voir maintenant les différents schémas qu'on va utiliser pour faire le relayage :

Schéma A

On utilise le CFP pour émettre les données relayées par le nœud relayeur. Dans ce cas-ci un GTS est réservé pour la transmission d'un paquet (ou une quantité de paquets). Donc, si la transmission ne réussit pas, le nœud relayeur retransmet le paquet dans le même GTS (bien sûr, s'il y a suffisamment temps jusqu'à la fin de ce GTS).

Dans cette figure, on montre un exemple où on peut voir que le récepteur n'a pas envoyé l'acquittement pour le paquet des données P1, donc le nœud relayeur retransmet ce paquet dans le même GTS :

Figure 4. Structure de la supertrame proposé pour le schéma A: exemple de la retransmission du paquet P1

Le principal avantage de ce schéma c'est que les GTS sont garantis pour les paquets à transmettre et comme ça on peut améliorer la fiabilité de la transmission, avec un délai contrôlé *end-to-end.*

Ce schéma est, donc, plus approprié pour applications médicales, alarmes… où le débit requis n'est pas très grand mais la latence est une contrainte importante.

Schéma B

On utilise la CAP pour émettre les données relayées par le nœud relayeur. Donc, si la transmission ne réussit pas le nœud relayeur accédera au support d'une manière aléatoire pendant la période de la CAP de la supertrame suivante avec un protocole CSMA/CA pour retransmettre le paquet.

Dans l'exemple présenté dans la prochaine figure on voit que le récepteur n'a pas envoyé l'acquittement (NACK, *Not Acknowledgement*) pour le paquet des données envoyé par le nœud #1 (P1). Donc, le nœud relayeur retransmettra ce paquet dans la période de la CAP de la supertrame suivante.

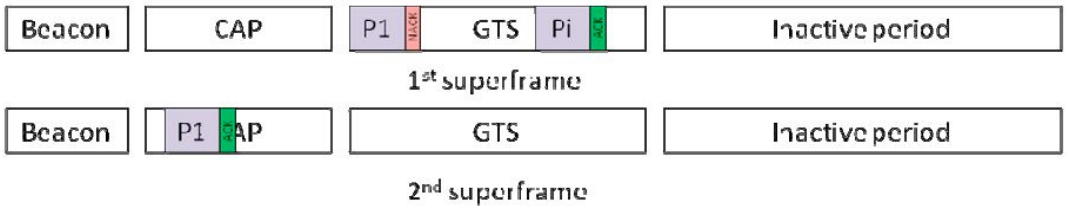

***Figure 5. Structure de la supertrame proposée pour le schéma B: exemple de la retransmission du paquet envoyé par le nœud #1* (P1)**

L'avantage principal de cette solution c'est que la réservation du GTS est optimale en référence au débit désiré, et les erreurs de transmission sont compensées par le nœud relayeur dans la prochaine CAP. Toutefois, ce schéma peut introduire une latence parce que les relais doivent attendre jusqu'à la CAP de la supertrame suivante pour pouvoir retransmettre les paquets. De même, les relais rivalisent avec les paquets de contrôle qui sont aussi envoyés dans la CAP (par exemple les paquets tels que le *Channel Time Request* et *Channel Time Response*).

Comme ça, cette solution est parfaite pour applications de *vidéo streaming* qui ont besoin d'un haut débit (donc le CFP ne doit pas être gaspillé avec retransmissions comme dans le schéma A) et a une tolérance pour petits délais.

On peut optimiser ce schéma pour diminuer la latence. Si le CFP et la CAP sont intervertis, alors un autre nœud retransmettra le paquet dans la CAP qui suit au CFP dans la même supertrame. Ici on peut voir un exemple (ce schéma n'a pas été implémenté dans les simulations):

Figure 6. Structure de la supertrame proposée pour le schéma B: exemple du changement entre le CFP et la CAP

Schéma C

Ce dernier schéma qui est proposé utilise la période inactive quand tous les nœuds sont endormis pour faire la retransmission.
On peut voir maintenant que ce schéma serait:

***Figure 7. Structure de la supertrame proposée pour le schéma C: exemple de la retransmission du paquet envoyé par le nœud #1* (P1)**

Normalement, la période inactive n'est pas utilisée parce que tous les nœuds dorment. Le protocole doit, donc, dire aux nœuds émetteurs qu'ils doivent écouter pendant cette période quand un paquet n'est pas acquitté.
De même, pendant cette période inactive un protocole d'accès au support doit être défini. On peut utiliser encore le CSMA/CA.
Ce protocole est le pire car certains nœuds doivent être réveillés pendant la période inactive avec la dépense énergétique correspondante. C'est pour ça qu'on ne l'a pas simulé.

2.2.4. Retransmission

De la même manière qu'on a un relai on peut avoir aussi une retransmission. Dans nos simulations on verra qu'on utilise soit le relai soit la retransmission.
La retransmission se fait quand l'acquittement envoyé par le nœud récepteur n'est pas reçu par le nœud émetteur.

La retransmission se fait toujours dans le même GTS réservé dans le CFP. Ça veut dire, si le nœud émetteur envoie un paquet et celui-ci n'est pas acquitté par le récepteur ça sera le même nœud émetteur qui émettra encore son paquet dans le même GTS.
Il est important de noter que la retransmission se fait seulement s'il y a suffisamment temps dans le GTS.

Une autre caractéristique de la retransmission c'est qu'elle se fait seulement une fois. Ça veut dire, si le paquet retransmis n'est pas acquitté alors on rejette le paquet de données et on envoie le suivant.

On verra les résultats obtenus en utilisant la retransmission et aussi en utilisant le relai (schémas A et B) dans les différents scenarios qu'on a simulé sur WSNet.
Maintenant on va expliquer un peu comment on a implémenté la couche MAC

comme une machine à états.

2.2.5. Implémentation machine à états

Dans ce point, on va présenter la machine à états de l'implémentation de la couche MAC qu'on utilise pour nos simulations dans WSNet.

En résumé, ce qu'on a modélisé c'est que tous les nœuds sont en état IDLE. À partir de là ils passeront d'un état à l'autre en fonction de ce qu'ils doivent faire. Par exemple, si nous sommes un nœud émetteur et ce que nous voulons faire est envoyer une commande CHANNEL TIME REQUEST au coordinateur on changera d'état comme ça. Le prochain état sera l'état BACKOFF où le nœud fera le backoff (temps aléatoire qu'il doit attendre jusqu'à envoyer le paquet pour voir s'il y a un autre nœud dans le support). Après ça il passera à l'état COMMAND pour envoyer la commande.

Un autre exemple serait quand le coordinateur commence la supertrame. En commençant, il doit envoyer le beacon à tous les nœuds du réseau. Ce qu'il fera sera changer son état jusqu'à BEACON et là il créera le beacon et il l'enverra. Après ça, il changera jusqu'au état TXING où il mettra à jour son horloge et changera encore son état jusqu'à IDLE.
Ici, on peut voir un schéma de tous les changements d'état qu'on a programmé dans la machine à états:

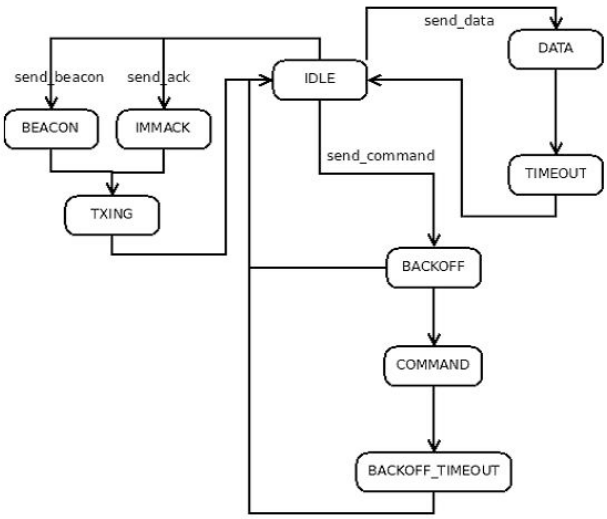

Figure 8. Machine à états de la couche MAC pour le cas sans relai

Dans la figure suivante on présente le schéma pour la machine à états dans le cas où on ajoute le relai.

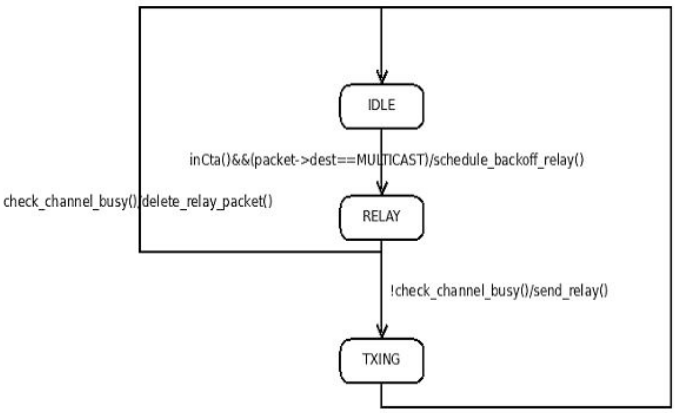

Figure 9. Intégration du relai dans la machine à états de la couche MAC

2.3. WSNet

WSNet (Wireless Sensor Network simulator) est le simulateur qu'on utilise pour faire toutes les simulations de notre protocole.
Il est un simulateur de réseaux de capteurs sans fil crée par Guillaume CHELIUS en interne au laboratoire CITI. Ce simulateur a été développé en langage C. L'architecture de WSNet est représentée sous forme de blocs modélisant les propriétés du médium radio et les caractéristiques des nœuds simulés.

On constate l'existence de trois blocs principaux dans WSNet:

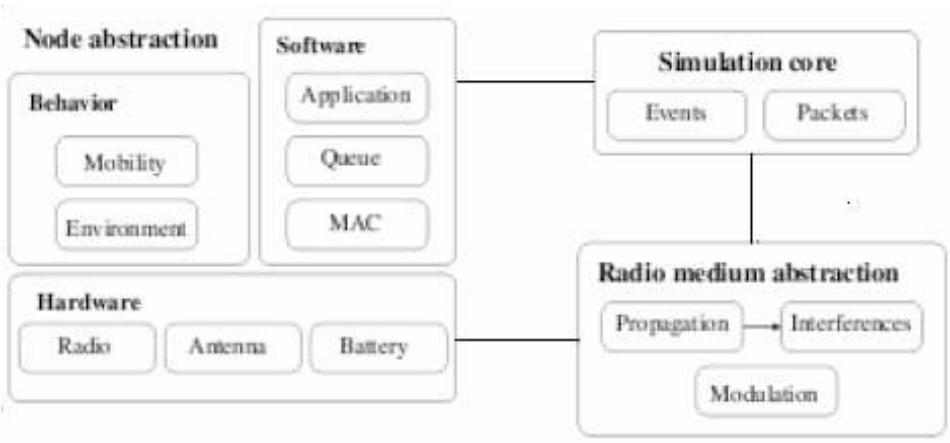

Figure 10. Architecture de WSNet

- Le cœur du simulateur, permet de générer et de traiter les événements et les paquets nécessaires lors d'une simulation. Ce composant est un élément clé de la performance d'un simulateur, en ce sens qu'il détermine le degré de rapidité et d'efficacité des simulations.

- Le médium radio, lui même constitué de plusieurs blocs: propagation, interférences et modulation.

- Le nœud, défini par plusieurs blocs également:
  - un bloc comportemental décrivant sa mobilité et l'environnement dans lequel il peut évoluer (feu, obstacles...).

- le hardware listant les propriétés physiques de nœud, telle la fréquence d'émission ou réception, la puissance d'émission, ou l'énergie...
- la pile protocolaire donnant les couches MAC, routage et applicative utilisées.

Lors d'une simulation sous WSNet, le comportement d'un bloc est instancié à l'aide d'un modèle. Un modèle est une implémentation particulière des fonctionnalités d'un bloc.

Maintenant, on va expliquer un peu comment configurer le simulateur pour faire une simulation.

Quand on utilise WSNet, il faut un fichier de simulation spécifié par le command –c. Donc la commande qu'on devrait écrire quand on commence une simulation c'est :

wsnet –c config_file.xml

Ce fichier de configuration c'est un fichier xml décrivant le scénario de la simulation. Il est composé de cinq grandes sections : **global parameters**, **entities**, **environment**, **bundles** et **node**.

- Dans les ***global parameters*** qui sont définis au début du fichier on définit par exemple le nombre de nœuds, la durée de la simulation et la taille de la région de simulation.

- Un ***entity*** c'est une instanciation d'une librairie dynamique. On définit l'entité en écrivant le nom qu'on va lui donner, le nom de la librairie qu'on veut instancier et les valeurs des paramètres avec lesquelles on veut commencer la simulation.

- Le bloc d'environnement (***environment***) décrit le moyen radio ainsi que l'environnement physique de la simulation.
  On définit le type de propagation, le type d'interférences, le type de modulation et l'entité de monitoring.

- Le ***bundle*** définit l'architecture de nœuds qu'on veut utiliser. On décrit l'ordre d'entités qu'on va suivre sur chaque nœud (typiquement, l'ordre de couches depuis la haute jusqu'à la plus basse).
  On doit, donc, créer un bundle pour chaque nœud.

- Finalement, on a la partie nœud (***node***). Elle n'est qu'une instantiation d'un bundle. Ce qu'on fait c'est dire le numéro de chaque noeud et lui attribuer le bundle correspondant crée avant.

On peut voir dans l'Annexe I un exemple du fichier xml que j'ai utilisé pour réaliser une simulation déterminée.

Finalement et en résumé, on doit dire que le simulateur implémente trois modules principaux:

- Le module de transmission qui définit la puissance émise, la fréquence, le débit, ainsi que d'autres paramètres de transmission.

- Le module de propagation qui calcule la puissance de réception à partir des paramètres de transmission, de la distance et des modèles de propagation implémentés (déterministe: freespace ; ou probabiliste: en considérant les phénomènes de fading et de shadowing).

- Le module de réception qui décide ou non de la réception d'un paquet, de la présence d'erreurs, et de l'état du lien radio, en se basant sur la puissance de réception (filtre en réception, sensibilité de l'antenne ...) et les opérations de calcul du SINR (Signal to Interference and Noise Ratio) et du BER (Bit Error Rate).

3. Performances de WSNet

Comme on a déjà dit, afin que le simulateur puisse simuler les scénarios il faut introduire quelques modifications au logiciel.
Donc le but de mon PFE a été au début de commencer à travailler sur celles là.

On a travaillé sur deux paramètres à chaque simulation: PER (probabilité d'erreur) et énergie.

Pour la PER ce que j'ai fait c'est obtenir le nombre de paquets reçus par le nœud coordinateur (parce que c'est le nœud où tous les paquets sont adressés) et faire la division entre le nombre de paquets envoyés par tous les nœuds. La PER est obtenu comme :

$$1 - (paquets_reçus / paquets_envoyés)$$

Donc, dans la couche MAC j'ai dû ajouter des instructions pour que le simulateur puisse me donner ces données à la fin de la simulation.

Par rapport à l'énergie consommée en transmission, j'ai dû faire des changements dans le code interne du simulateur.
Le dispositif qu'on utilise comme capteur c'est le **CC2520** de Texas Instruments. Donc, j'ai regardé le datasheet pour obtenir les équations qui peuvent nous donner la dépense énergétique en prenant en compte les données du fabricant.

Pour obtenir l'énergie on sait qu'elle est :

$$E = P \cdot t$$

Donc, dans le datasheet on sait la courant électrique dépensée par le capteur depuis 5dBm de puissance de transmission jusqu'à -35dBm donc on peut facilement obtenir une équation linéaire qui nous donne l'énergie et la mettre dans le fichier *linear.c* (on peut voir le fichier dans l'annexe II).

Par rapport à l'énergie consommée en réception, c'est la même indépendamment de la puissance de transmission. On connait le courant électrique dépensé par le capteur en réception (18.5mA) et la tension d'utilisation (3V), donc on calcule la puissance (0.0555W) et après on fait la multiplication par le temps.

Considérations par rapport aux simulations

Les simulations que j'ai faites ont eu un temps simulé de 100000s pour obtenir la PER (c'est un valeur que j'ai obtenu et j'ai cru suffisamment grand pour obtenir des valeurs réelles). Avec plus de temps de simulation les valeurs obtenues ne changent pas beaucoup mais, toutefois, le temps réel de simulation augmente beaucoup.

Par rapport à l'énergie, les simulations ont été plus courtes parce qu'en fait le

temps n'est pas important pour les valeurs obtenus (on est seulement intéressés, comme on verra plus tard dans les Joules consommés par paquet reçu donc le fait d'avoir plus de temps seulement augmente l'énergie mais en fait augmente aussi le nombre de paquets donc le résultat va être le même).
Finalement les simulations pour l'énergie ont eu un temps simulé de 1000s.

En ce qui concerne la puissance de transmission, j'ai toujours simulé depuis 0dBm jusqu'à -45dBm en accord avec le modèle de capteur que j'ai utilisé (CC2520). C'est important de savoir que pour la PER les valeurs obtenus de 0dBm jusqu'à -13dBm ne sont pas prises en compte parce que la PER est si basse qu'elle est presque 0.

La durée de la supertrame est de 0.06s tandis que la durée de la CAP est de 0.00211s (de toutes manières dans l'analyse du troisième scénario on changera cette dernière durée pour voir l'effet sur la probabilité d'erreur).

La couche application donne un paquet à la couche MAC chaque 0.06s (ça veut dire, chaque supertrame). La taille de ces paquets est de 900 octets.

Finalement, on doit dire que les simulations qu'on a faites ont été sans inclure ni le shadowing ni le fading. La raison a été parce qu'on n'avait pas les données de ces deux phénomènes pour tous les liens qu'on a simulé. De toutes manières, à la fin de ce rapport on fait une explication théorique des facteurs qu'on doit prendre en compte quand on travaille avec le shadowing et le fading.

4. Simulations

4.1. Scénario 1. Trois nœuds (Emetteur-Relai-Recepteur)

4.1.1. Analyse de la PER

Le premier scénario qu'on a simulé est constitué par trois nœuds, un émetteur, un relai et un récepteur.
C'est le scénario le plus simple afin de pouvoir tester notre software sur WSNet.
Les nœuds sont placés dans l'espace à une distance de 0.5m chacun de l'autre. Donc ils forment un triangle de 0.5m de côté. Maintenant je donne les numéros de chaque nœud et ses coordonnés.

Nœud 0 (Coordinateur): x=0, y=0, z=0
Nœud 1 (Émetteur): x=0.5, y=0, z=0
Nœud 2 (Relai): x=0.25, y=0.433, z=0

Toutes les unités de distance sont en mètres.

La première chose qu'on va voir c'est la probabilité d'erreur calculée comme on a expliqué dans le chapitre avant.
On a utilisé les algorithmes suivants :

- ✓ Sans relai et sans retransmission
- ✓Avec relai dans la CAP
- ✓Avec relai dans le CTA
- ✓Avec retransmission (donc dans le CTA)

On présente ici les résultats :

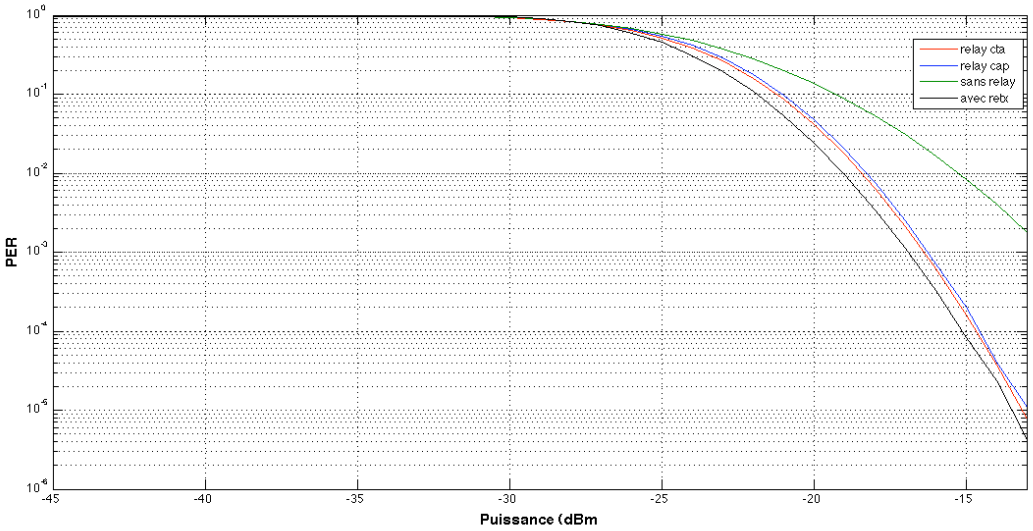

Figure 11. PER dans le scénario 1

Si on compare les deux algorithmes de relayage (CAP et CTA), on peut voir que le comportement est presque le même mais de toutes manières le relai dans le CTA

est un peu mieux.
Il y a deux raisons qui expliquent ce résultat.

1) Le relai dans la CAP a un problème si le relayage se fait dans cette partie de la supertrame, on dépend de si on a suffisamment temps pour finir le relayage jusqu'à la fin de cette partie. Dans ce cas là, la simulation est faite avec un temps de CAP de 2110000ns. On fera des simulations avec un modèle plus complexe de 8 nœuds avec différentes tailles de la CAP pour voire qu'en effet, si on fait une CAP plus petite la PER va être plus grande.

2) Dans la CAP il n'y a pas seulement des données à transmettre sinon aussi des commandes comme le CHANNEL TIME REQUEST ou le CHANNEL TIME RESPONSE donc le nœud relayeur doit utiliser un protocole comme le CSMA/CA d'accès au support pour pouvoir envoyer des données et quelques fois il ne pourra pas parce qu'il y a beaucoup des commandes à envoyer.

Il faut dire aussi que la transmission sans aucun algorithme est bien sûr la pire parce que si un paquet est perdu il est directement rejeté.

On peut voir aussi la grande différence entre utiliser le relai ou la retransmission et ne pas l'utiliser. C'est important de noter que l'utilisation de la retransmission est mieux que celui du relai.

L'explication est facile, le schéma que j'ai utilisé est avec une distance équivalente (0.5m) entre chacun des trois nœuds, à ce moment là c'est mieux la retransmission parce que:

1) Le paquet ne doit pas être envoyé au nœud relai (qui est à la même distance du nœud récepteur) et de là au nœud récepteur sinon qu'il va directement jusqu'à là. En plus, il y a un risque de perdre du paquet lorsqu'on l'envoie au nœud relai.

2) La retransmission se fait par le nœud émetteur, l'unique façon qu'il a de savoir si le paquet est arrivé ou non est de voir si l'acquittement a été reçu ou pas.
   En cas qu'il n'est pas reçu (soit parce que le paquet n'est pas arrivé au nœud récepteur, soit parce que c'est le propre ACK qui s'est perdu) il renvoie le paquet. Toutefois le relai se fait seulement si le paquet n'est pas arrivé au nœud récepteur (et donc l'ACK n'est pas envoyé) donc il y aura beaucoup plus de retransmissions que de relais.

Le relai est un algorithme, donc, qui vaut mieux l'utiliser en ayant un nœud (relai) qui est plus près du récepteur que la source. C'est pour ça que j'ai aussi simulé ce cas là en ayant le nœud relai a 0.25m du récepteur tandis que la source et le récepteur se maintiennent à 0.5m, le résultat est bien sûr celui qui était prévu.

Maintenant les coordonnées sont :

Nœud 0 (Coordinateur): x=0, y=0, z=0
Nœud 1 (Émetteur): x=0.5, y=0, z=0
Nœud 2 (Relai): x=0.25, y=0, z=0

Les résultats maintenant sont ceux prévus :

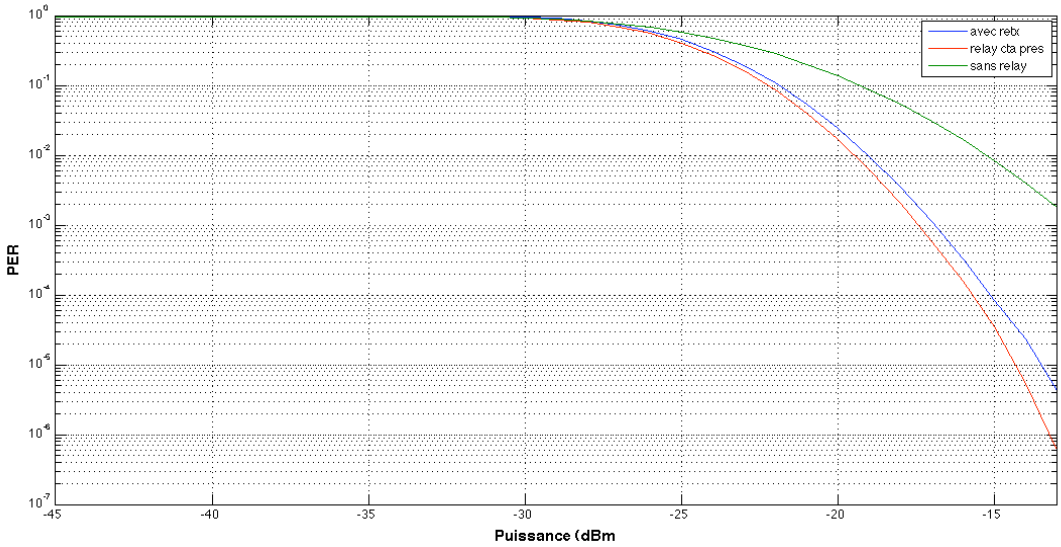

Figure 12. PER dans le scénario 1 avec le relai plus près du nœud récepteur

On peut voir, qu'en ayant le nœud relai plus près du récepteur le résultat avec le relai améliore celui de la retransmission.
C'est important aussi de noter que quand la puissance est si faible que presque aucun paquet arrive à sa destination, même si l'on utilise le relayage ou la retransmission le résultat est presque le même que sans aucune de ces méthodes (à peu près à partir de $2\mu W = -27dBm$).

4.1.2. Analyse de l'énergie dépensée

Maintenant on va faire l'analyse de l'énergie dépensée par chaque nœud. Comme on a déjà dit, on calcule l'énergie dépensée par paquet reçu par le nœud coordinateur. Comme ça on peut savoir pour quelle énergie on a le meilleur comportement en prenant en compte l'énergie dépensée et le nombre de paquets reçus.
Par exemple on ne peut pas dire, on va utiliser une puissance très faible pour que la consommation d'énergie soit faible, parce que là le nombre de paquets reçus va être très bas. On doit arriver à un compromis entre l'énergie et le nombre de paquets reçus.

Dans la prochaine page je montre les résultats. J'ai utilisé encore la structure en triangle. Comme on pourra voir, le comportement énergétique ne dépend pas beaucoup de l'algorithme de relayage qu'on utilise (soit dans la CAP soit dans le CTA).

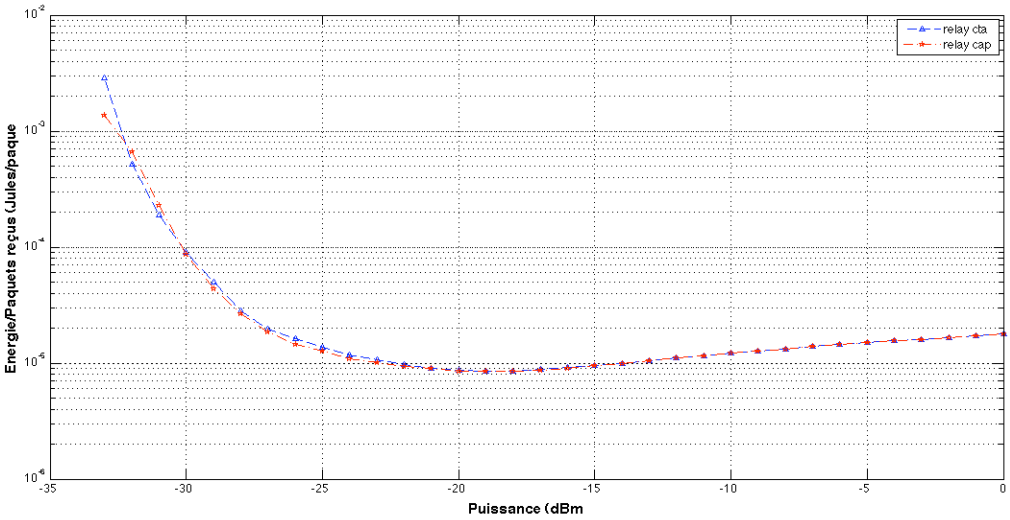

Figure 13. Énergie dépensée par le nœud émetteur

Dans ce cas, on peut voir que la puissance optimale de transmission est plus ou moins autour **-19dBm**. Avec cette puissance on a pas mal de paquets reçus par rapport à l'énergie dépensée. Avec une puissance plus faible le nombre de paquets reçus est si petite que la courbe augmente beaucoup et avec une puissance plus grande la consommation d'énergie est très haute par rapport au nombre de paquets reçus.

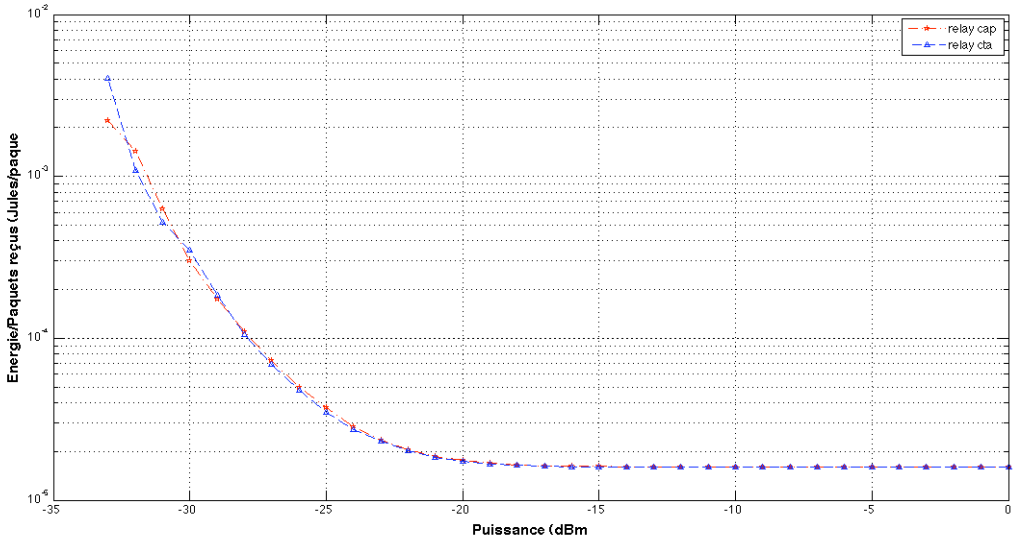

Figure 14. Énergie dépensée par le nœud relai

Dans ce cas, on voit que la consommation d'énergie est, comme il est logique, constante jusqu'à -17dBm parce qu'il n'y a presque pas de relayage jusqu'à cette puissance. La puissance de transmission idéale est autour -16dBm mais on pourrait émettre aussi à -19dBm sans que la courbe remonte beaucoup.

Remarque: *Je n'ai pas simulé le cas pour le nœud récepteur parce qu'en fait il n'envoie pas de paquets et seulement les acquittements et les commandes ainsi que le beacon, donc la courbe va être très pareille à celle du relai (presque constante jusqu'à -17dBm et après elle remontera dû au fait de recevoir moins paquets).*

4.2. Scénario 2. Trois nœuds (Deux émetteurs/relais-Recepteur)

4.2.1. Analyse de la PER

Maintenant j'ai simulé une situation avec trois nœuds aussi mais maintenant on a deux émetteurs et deux relais. Les nœuds 1 et 2 vont émettre des paquets et être chacun le relai de l'autre.
Donc, maintenant la structure qu'on a simulée est encore:

Nœud 0 (Coordinateur): x=0, y=0, z=0
Nœud 1 (Émetteur et Relai): x=0.5, y=0, z=0
Nœud 2 (Émetteur et Relai): x=0.25, y=0.433, z=0

On a aussi simulé les mêmes algorithmes qu'avant, les résultats obtenus sont les suivants :

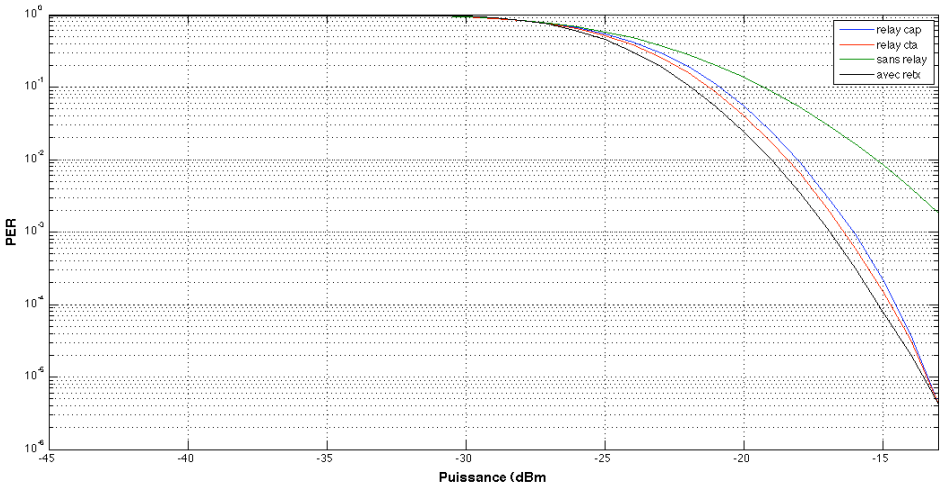

Figure 15. PER dans le scénario 2

On peut voir que les probabilités d'erreur sont plus ou moins les mêmes qu'avec le scénario 1 mais, néanmoins, on voit quelques différences surtout entre le relai dans la CAP et le relai dans le CTA.

Maintenant la différence entre ces deux algorithmes est plus grande, le relai dans le CTA est toujours mieux et le relai dans la CAP est pire que dans le scénario 1. L'explication est très facile si on se rappelle de ce qu'on a dit dans le précédent scénario.

On avait dit que le relai dans la CAP est, en résumé, pire que dans le CTA parce qu'il y a moins temps dans la CAP que dans le CTA donc quelquefois le relai n'a pas le temps d'envoyer les données relayés et, en plus, il n'y a pas seulement paquets de données à envoyer sinon aussi commandes comme le CHANNEL TIME REQUEST.
Maintenant on a deux nœuds qui sont relais donc il y aura plus de relais et, bien sûr, plus de nœuds qu'avant qui vont essayer de relayer dans la CAP, donc il y aura pas mal de fois qu'il n'y aura pas assez de temps pour envoyer les données relayées dû au manque de temps dans la CAP.

Par rapport aux autres algorithmes les comportements sont plus au moins comme avant.
Pour ce scénario je n'ai pas fait la simulation en plaçant le relai au milieu parce que ce n'est pas possible dû au fait que chaque nœud émetteur est le relai de l'autre.

4.2.2. Analyse de l'énergie dépensée

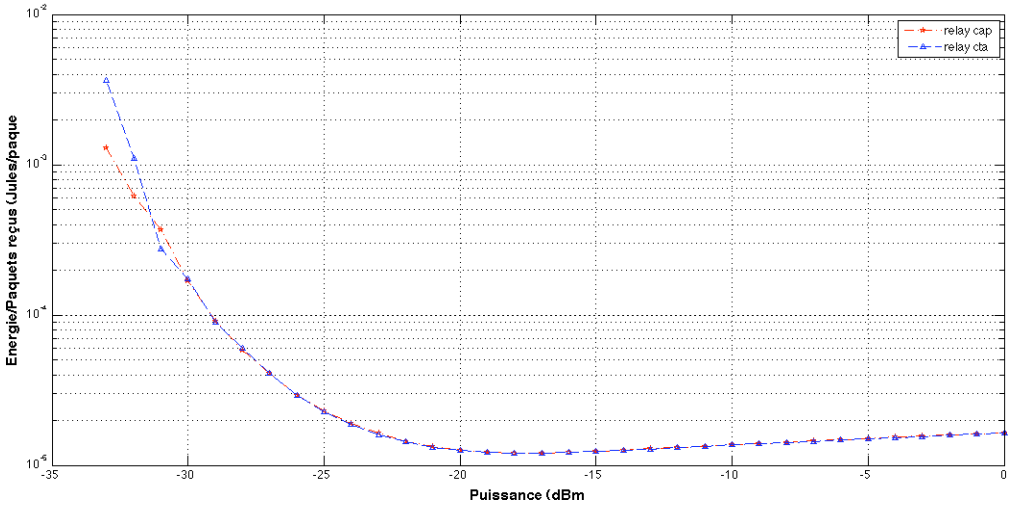

Figure 16. Énergie dépensé par le noeud 1 (source+relai)

Maintenant on a seulement analysé le cas pour le nœud 1 dû au fait que le nœud 2 fait exactement le même travail (il est aussi relai et source) donc les valeurs d'énergie sont plutôt pareilles.
Comme on peut voir le résultat est pareil au scénario précédent. La puissance de transmission optimale est autour **-18dBm**.
Il y a seulement un changement et c'est que l'énergie dépensée par paquet est plus grande dû au fait que maintenant le nœud est aussi relai et source donc il doit envoyer plus de paquets.

Comme avant, le comportement énergétique ne dépend pas beaucoup de l'algorithme utilisé.

4.3. Scénario 3. Neuf nœuds (Huit émetteurs/relais-Recepteur)

4.3.1. Analyse de la PER

Ce troisième scénario est composé par neuf nœuds, dont un le coordinateur (récepteur) et les autres huit nœuds sont à la fois émetteurs et relais.
Les nœuds sont placés de la manière suivante:

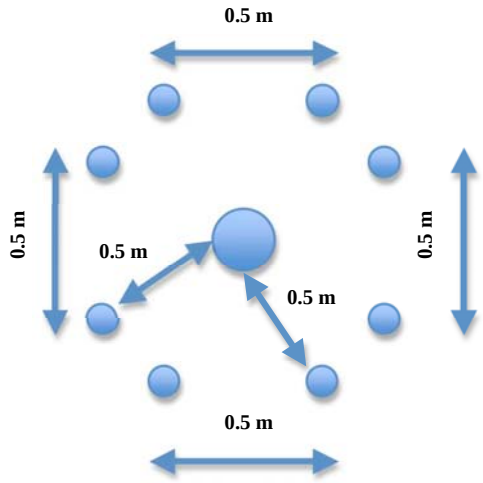

Figure 17. Placement des nœuds dans le scénario 3

Au milieu on a le nœud coordinateur et les autres huit nœuds sont placés autour de lui à une distance de 0.5m et aussi à 0.5m deux à deux comme on peut voir dans la figure. Du coup, la distance entre les nœuds 1-2, 3-4, 5-6 et 7-8 est de 0.5m.
Ils sont placés comme ça parce que le relai est fait aussi deux à deux, le 1 relai le 2 et vice-versa, et le même pour le 3-4, 5-6 et 7-8.

Donc, maintenant chaque nœud est placé dans ces positions :

Nœud 0 (Coordinateur): x=0.433, y=0.433, z=0
Nœud 1 (Émetteur et Relai): x=0.683, y=0.866, z=0
Nœud 2 (Émetteur et Relai): x=0.183, y=0.866, z=0
Nœud 3 (Émetteur et Relai): x=0, y=0.683, z=0
Nœud 4 (Émetteur et Relai): x=0, y=0.183, z=0
Nœud 5 (Émetteur et Relai): x=0.183, y=0, z=0
Nœud 6 (Émetteur et Relai): x=0.683, y=0, z=0
Nœud 7 (Émetteur et Relai): x=0.866, y=0.183, z=0
Nœud 8 (Émetteur et Relai): x=0.866, y=0.683, z=0

On a fait encore la simulation des quatre algorithmes:

- ✓ Sans relai et sans retransmission
- ✓Avec relai dans la CAP
- ✓Avec relai dans le CTA
- ✓Avec retransmission (donc dans le CTA)

Voici les résultats:

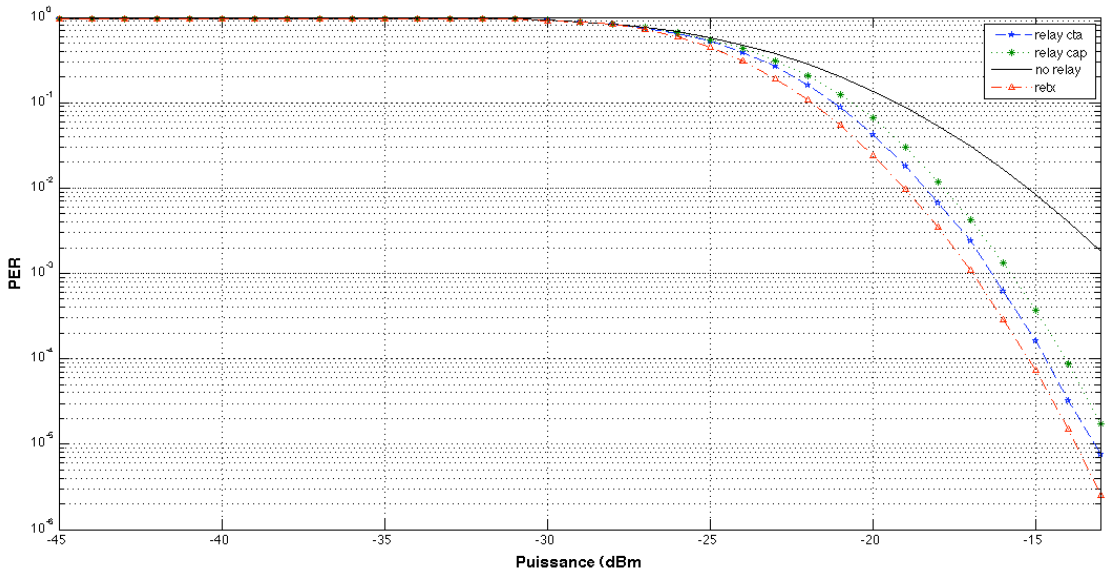

Figure 18. PER dans le scénario 3

Comme il est logique, on voit que la différence maintenant entre le relais dans la CAP et dans le CTA est plus grande que lorsqu'on avait avec 2 nœuds (quelquefois il y a beaucoup de nœuds qui envoient dans la CAP donc il y a moins de relayage avec succès que dans l'algorithme où on le fait dans le CTA parce qu'il n'y a pas de temps pour tous dans la CAP). Cette fois en ayant huit nœuds qui vont faire le relai l'espace temporel dans la CAP est quelquefois insuffisant.
En plus il faut rappeler qu'avec l'algorithme de la CAP on a moins relayage aussi dû à la situation suivante:

- Si le beacon n'est pas reçu par un nœud il ne peut pas relayer. Mais, néanmoins, si le relayage est dans la CTA, c'est seulement dans la supertrame actuelle qu'on ne peut pas faire le relayage, mais si le relayage est dans la CAP, comme on doit faire le relayage dans la supertrame suivante, ils sont deux le paquets affectés. D'un côté dans la supertrame suivante on ne peut pas relayer. Et de l'autre, dans la supertrame actuelle si le nœud allait relayer il ne le fait pas non plus parce qu'il ne peut pas utiliser la CAP sans recevoir le beacon.

Dans le cas avec seulement retransmission on a le meilleur comportement dû au fait que les nœuds sont situés à 0.5 m du coordinateur et aussi à 0.5 m de ses relais. Ça veut dire qu'il est mieux de retransmettre soi même ses paquets que de les envoyer à ses relais qui sont à 0.5 m d'eux et en plus à des autres 0.5 m du coordinateur.

On a déjà dit que quelquefois l'espace dans la CAP est insuffisant parce qu'on a maintenant huit relais. On a fait la simulation du relai dans la CAP avec une CAP de 2110000ns (2.11ms). Ce qu'on a fait c'est simuler certains cas avec des CAPs plus courtes pour prouver que la taille de la CAP a une influence sur la PER.

Voici les résultats obtenus :

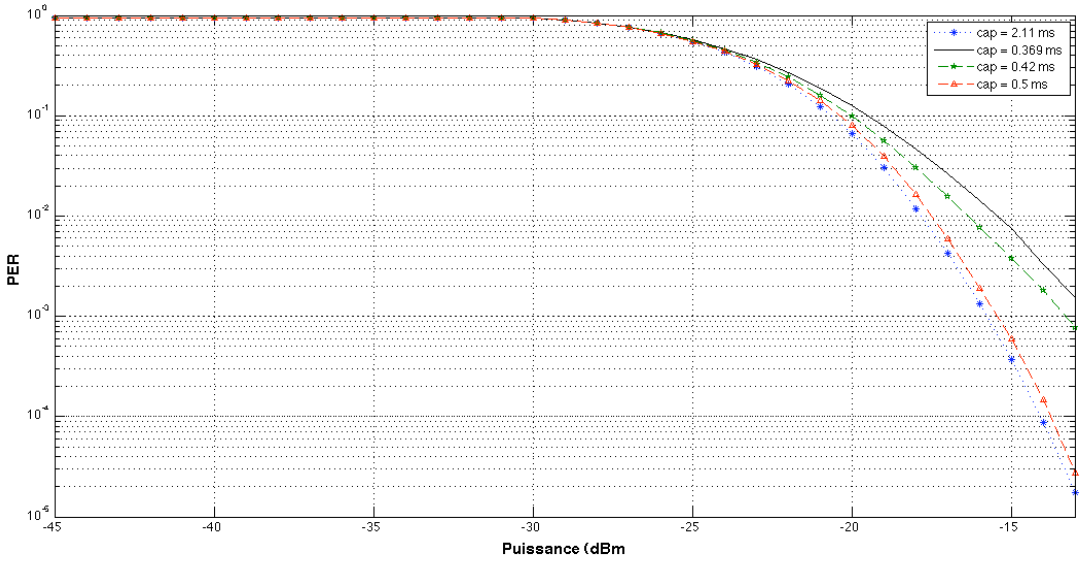

Figure 19. Comparaison entre différentes tailles de la CAP, scénario 3

On peut vérifier que si la CAP est plus petite la PER est pire dû au fait qu'il y a moins de relais.
La CAP est si petite qu'en dépendant du numéro de backoffs qui sont faits il y a de l'espace pour faire le relayage ou pas. Par exemple j'ai utilisé une longueur de CAP de 0.369ms parce que c'est la longueur minimale qu'on peut choisir pour avoir relayage (au delà, l'espace est si petit qu'il n'y a pas suffisamment temps pour finir un relayage). Je veux dire qu'avec cette longueur de CAP seulement les relais qui ont une valeur de backoffs de 0 peuvent être faits. À mesure que la longueur est plus grande les relais avec une valeur de backoffs aussi plus grande peuvent être faits et c'est pour ça qu'on a plus relayage et évidemment une PER plus basse.
Quand on arrive a une longueur plus grande de 0.5ms presque tous les relais sont faits, et c'est pour ça que la courbe se rapproche beaucoup à celle de 2.11ms.

Finalement, on se rend compte qu'avec une longueur de 0.8ms le résultat est presque le même qu'avec 2.11ms :

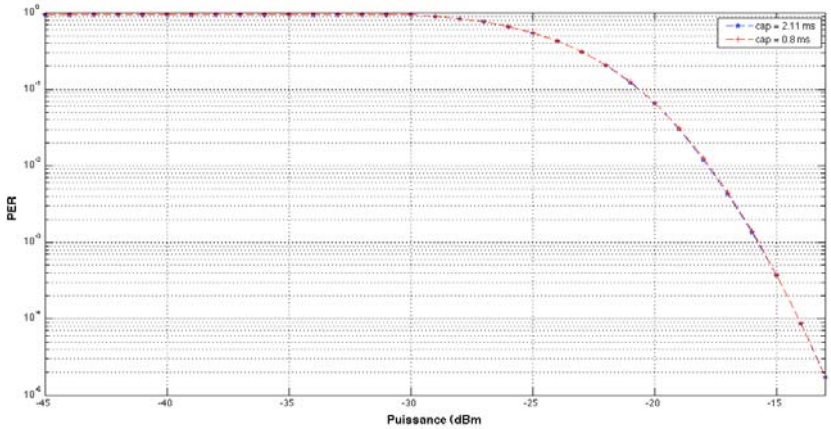

Figure 20. Comparaison entre CAP=0.8ms et CAP=2.11ms

C'est important de trouver ce résultat parce qu'avec une longueur de 0.8ms (1.31ms de moins que le précédent) on a le même comportement donc on peut économiser de l'espace pour le CTA et donc avoir là plus d'espace pour envoyer des données et du coup avoir un débit plus grand.

Jusqu'à maintenant, on a vu la probabilité d'erreur (PER) calculée comme le nombre de paquets reçus divisé par le nombre de paquets envoyés par **l'antenne**.
Les prochains résultats qu'on va voir ont été construits en faisant la division entre le nombre de paquets reçus et le nombre de paquets envoyés par **la couche application**. Il existe une différence importante quand on parle des paquets envoyés par l'antenne et des paquets envoyés par l'application. Alors, le nombre de paquets envoyés par l'antenne est plus petit que le nombre de paquets envoyés par l'application à mesure qu'on a une puissance plus faible.
La cause est que quand la puissance est plus faible le beacon envoyé par le nœud coordinateur n'est pas reçu par quelques nœuds, donc ces nœuds ne peuvent pas envoyer ses paquets pendant la supertrame actuelle. Donc, bien qu'ils avaient un paquet pour envoyer transmis par l'application il ne peut pas être envoyé.

On va voir les résultats qu'on a obtenus:

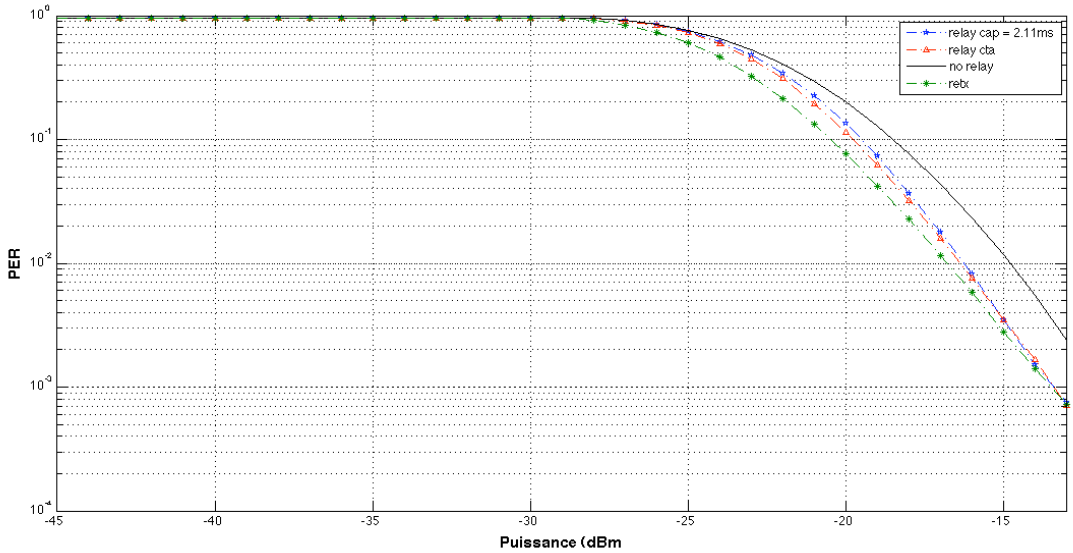

Figure 21. PER dans le scénario 3 en fonction du nombre de paquets envoyés par l'application

Même si la différence entre les quatre courbes est un peu plus petite qu'avant, le comportement est le même. De toutes façons c'est important de se rendre compte que maintenant la PER est **plus haute** parce que le nombre de paquets envoyés par la couche application est toujours constante (on envoie depuis l'application toujours un paquet chaque 0.06s, c'est à dire un paquet chaque supertrame).
Mais comme on a déjà dit tandis que ce numéro est constant le nombre de paquets envoyés par l'antenne diminue avec des puissances plus faibles. Cela fait que la PER soit plus bas dans ce cas là.
Le mieux est toujours l'algorithme avec retransmission, après c'est mieux le relai dans le CTA, après dans la CAP et pour finir le pire comportement est celle sans relai et sans retransmission.

4.3.2. Analyse de l'énergie dépensée

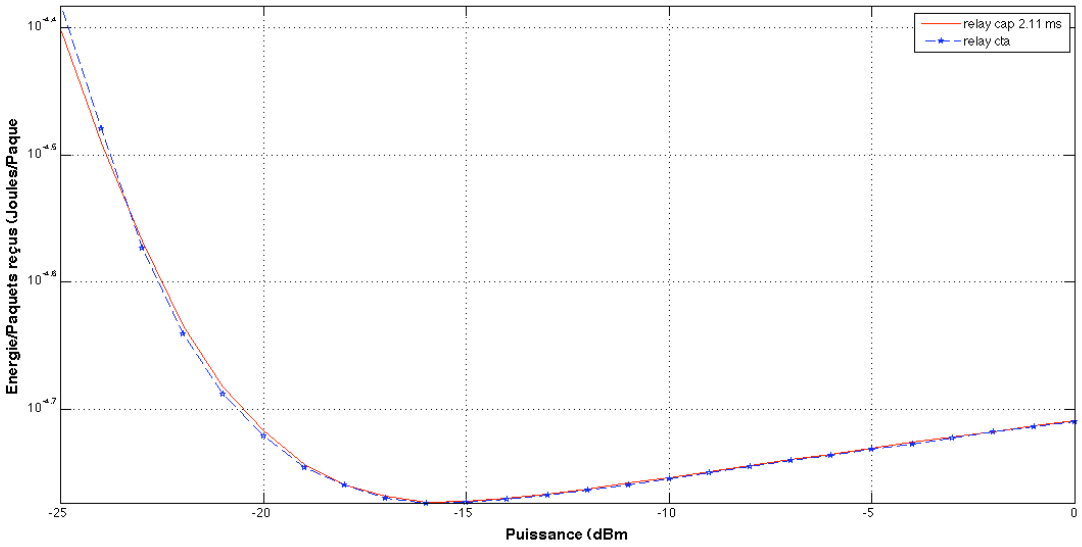

Figure 22. Énergie dépensée par le nœud 1 sur le scénario 3

En termes d'énergie on peut voir qu'avec 8 émetteurs/relais on a presque le même comportement qu'avec 2 émetteurs/relais, l'énergie dépensé par chaque algorithme est presque la même.
Si on veut avoir un minimum d'énergie dépensée en termes de paquets reçus (la plus efficace) on devrait émettre à une puissance de **-16dBm**. Avec cette puissance on aurait une PER de l'ordre de 10^{-3}.
On pourrait aussi émettre à une puissance un peu supérieure sans dépenser beaucoup plus d'énergie mais en gagnant beaucoup en termes de PER.

5. Shadowing et Fading

Comme on a déjà dit, tous les scénarios qu'on a simulés ont été faits sans shadowing ni fading. C'est à dire, il n'y a pas des pertes dues à ces phénomènes.

Pour rappeler, le *shadowing* est la déviation dans l'atténuation qu'un signal de télécommunication subit dans un moyen de propagation déterminé.
Il varie avec le temps et sa cause sont les obstacles qui affectent la propagation de l'onde.
Le *fading* est aussi une déviation dans l'atténuation du signal mais dans ce cas la cause est le *multipath*. Ça veut dire la propagation d'un signal radio par plusieurs chemins. Les causes de l'atténuation *multipath* sont la réflexion sur l'ionosphère, réfraction, réflexion et diffraction par les obstacles naturels ou des bâtiments.

Actuellement, nous n'avons pas toutes les données du shadowing et fading pour tous les liens qui vont être crées dans la version finale du réseau BAN (ces données doivent être obtenus par le laboratoire CEA dans le cadre du projet BANET). Le placement final des nœuds sera:

- Nœud 0 (coordinateur). Placé dans la hanche.
- Nœud 1. Placé dans le dos.
- Nœuds 2 et 4. Placés dans la cuisse gauche et droite respectivement.
- Nœuds 3 et 5. Placés dans le pied gauche et droit respectivement.
- Nœud 6. Placé dans le torse.
- Nœuds 7 et 9. Placés dans le bras gauche et droit respectivement.
- Nœuds 8 et 10. Placés dans la main gauche et droite respectivement.
- Nœuds 11 et 12. Placés dans l'oreille gauche et droite respectivement.

Ici, on montre les valeurs du shadowing et fading pendant 4 seconds pour le lien hanche-pied droit:

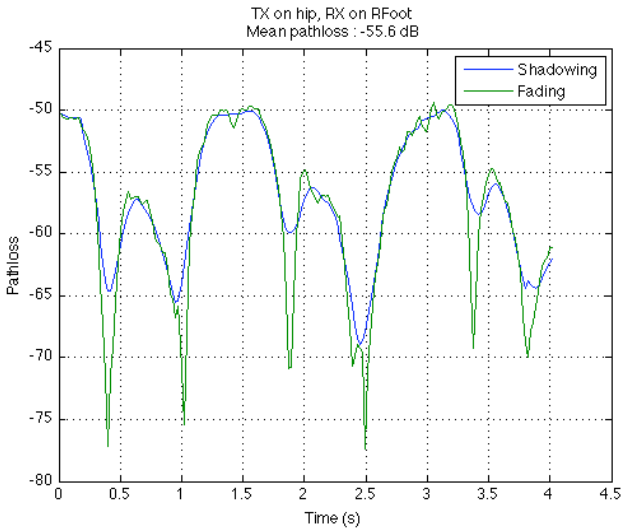

Figure 23. Shadowing et Fading pour le lien hanche-pied droit

On a crée un nouveau fichier de propagation pour WSNet où on va prendre en compte tous les valeurs pour tous les liens une fois qu'on les a reçus depuis le CEA. Toutes ces valeurs vont être importantes parce qu'ils pourront nous permettre de mieux choisir les nœuds qui doivent faire le relayage à chaque instant.

Par exemple si on a le nœud placé dans le pied droit et dans un instant donné les conditions de shadowing et fading nous font avoir des pertes importantes qui peuvent compromettre la réception du paquet, on doit chercher un nœud qui dans son lien jusqu'au même nœud destinataire a des conditions de shadowing et fading plus favorables.
Comme ça, on utilisera toujours comme relais les nœuds qui ont moins de pertes dans le lien et on diminuera la PER.
Une fois que ces simulations seront faites, on pourra voir comme dans tous les cas l'algorithme avec le relai (soit le relai dans la CAP, soit dans le CTA) sera mieux que la retransmission.

De toutes manières ce genre des simulations n'ont pas pu être réalisées pendant ce projet et sera un sujet traité plus tard, une fois que toutes les données de shadowing et fading seront envoyées par le CEA.

6. Conclusion

L'objectif de mon Projet de Fin d'Etudes était d'étudier l'implémentation d'un réseau BAN dans le simulateur WSNet. J'ai dû étudier différents scénarios et en utilisant différents algorithmes de transmission des données.

Pour m'approcher de cet objectif, j'ai du comprendre dans un premier temps la technologie BAN et le comportement du simulateur WSNet. J'ai dû aussi me documenter sur le protocole qu'on a utilisé pour la couche MAC (IEEE 802.15.3).

Dans le déroulement de mon PFE, j'ai intégré un laboratoire de recherche, le CITI, dans lequel j'ai pu découvrir le métier de chercheur en partageant des données, et en échangeant des idées. J'ai également pu approfondir mes connaissances dans les réseaux de capteurs et comprendre l'intérêt grandissant des chercheurs pour ce sujet. J'ai pu par ailleurs compléter mes compétences au niveau de programmation en C et XML. J'ai acquis des connaissances dans un simulateur de réseaux de capteurs comme le WSNet. Enfin, j'ai pu développer ma capacité à conduire un projet de bout en bout et ainsi évoluer en autonomie malgré les complications et les imprévus que j'ai pu avoir à chaque étape du projet, comme le fait que le simulateur WSNet n'aie pas la couche MAC du protocole IEEE 802.15.3 implémenté ou qu'il y avait quelques erreurs dans le code que j'ai reçu avant de commencer le PFE.

Par manque de temps, je n'ai pas pu faire les simulations avec les données du shadowing et fading. En plus, on a reçu seulement les données pour quelques liens. J'ai fait, quand même, avec Paul FERRAND le fichier *ban_cea.c* qui va permettre aux chercheurs qui travailleront sur ce sujet de simuler les situations en prenant en compte les données du shadowing et fading une fois qu'ils sont reçus. Ils devront seulement utiliser le fichier des données reçues et notre programme retournera les données du *pathloss*.

En résumé, je dois dire que le sujet du PFE a été très intéressant et l'ambiance dans le laboratoire a été toujours parfaite.
En général, je recommanderai à quelque soit la personne qui veuille venir à l'INSA de Lyon de faire son PFE et en particulier au laboratoire CITI de ne pas douter et venir parce qu'à la fin elle va être très contente pour le travail fait.

Bibliographie

[1] IEEE 802.15.4 Standard, "Wireless MAC and Physical Layer Specifications for Low Rate Wireless Personnal Area Networks", Aug. 2007

[2] C. Roblin and al. "Propagation channel models for BANs: an overview" TD 09 760 7th COST2100 MCM Braunschweig, Germany Feb. 2009

[3] Web Officiel Bluetooth: http://www.bluetooth.com/bluetooth/

[4] I. Pantazis, "Tracking Human Walking using Marg sensors", Thesis 2005

[5] Yazdandoost and Sayrafian, "Channel Model for Body Area Network", IEEE 802.15 Working Group for WPANs Document IEEE P802.15-05-0780-05-0006, Feb. 2009

[6] P.S. Hall et al., "Antennas and Propagation for On-Body communication Systems", in IEEE Antennas and Propagation Magazine, June 2007

[7] G.S. Ahn, M.J. Lee, S.S. Joo: "Versatile MAC for Body Area Network", TG6 Call for Proposals (15-08-0811-03-0006-tg6-call-proposals), May 2009

[8] Web Officiel WSNet: http://wsnet.gforge.inria.fr/

[9] Web Officiel du Projet BaNet: https://www-banet.cea.fr/

[10] Chin, K.W. and Lowe, D., MiniMesh: an opportunistic transmission protocol for the IEEE 802.15.3 MAC, 2007

[11] IEEE 802.15.3 Standard, "Wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wireless personal area networks (WPANs)", 2003

[12] Information en Wikipedia sur Zigbee: http://en.wikipedia.org/wiki/ZigBee

[13] Marandin, D. "ZigBee: Standard for Control and Sensor Networks", ITG FG 5.2.4 Workshop "Ad hoc, Sensor and Mesh Networking: New Opportunities for Wireless Communication" München, 10.02.2005

[14] Savigny, C. "Rapport BANET", Mai 2006

[15] Karowski, N., "Passive Discovery schemes for opportunistic message relaying schemes base don IEEE 802.15.4"

Annexe I

Dans cette annexe, je vais montrer le fichier de configuration de WSNet que j'ai utilisé pour le scénario 3 et pour une puissance de transmission de 0dBm.

```
<?xml version='1.0' encoding='UTF-8'?>
<worldsens xmlns="http://worldsens.citi.insa-lyon.fr">


<!-- == Worldsens
====================================================== -->
<simulation nodes="9" duration="100000s" x="2" y="2" z="2"/>

<!-- == Entities
====================================================== -->


<!-- == PROPAGATION, INTERFERENCES, MODULATION and ENERGY
==================== -->

<entity name="interf" library="interferences_orthogonal">
</entity>


<entity name="o-qpsk" library="modulation_oqpsk">
</entity>
<entity name="white" library="noise_white">
    <init white-noise-dBm="-84"/>
</entity>
<entity name="bat" library="energy_linear">
      <default energy="100000" tx="0.00468" rx="0.0555" idle="0"/>
</entity>



<!-- == RADIO and ANTENNA ========================================
-->
<entity name="omnidirectionnal" library="antenna_omnidirectionnal" >
  <default loss="0" angle-xy="random" angle-z="random"/>
</entity>

<entity name="radio" library="radio_half1d" >
    <default   sensibility="-92"      T_s="90"   dBm="0"   channel="0"
modulation="o-qpsk"/>
</entity>


<!-- == MAC ================================================== -->
<entity name="mac" library="mac_802153" >
    <init  modeack="1" moderelay="1" moderetx="0"/>
    <default           pnc="0"           superframeduration="60000000"
capduration="800000"  dataperiod="60000000" />
</entity>
<entity name="ban_propag" library="propagation_test" >
      <init data_description_file="ban_links_description.csv"/>
</entity>
<!--                            ==                            Application
================================================== -->
<entity name="cbr0" library="application_scenario1" >
<default period="0" start="0" size="900" destination="1"/>
```

```
</entity>
<entity name="cbr1" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>

<entity name="cbr2" library="application_scenario1" >
<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>

<entity name="cbr3" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>
<entity name="cbr4" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>
<entity name="cbr5" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>
<entity name="cbr6" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>
<entity name="cbr7" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>
<entity name="cbr8" library="application_scenario1" >

<default      period="60000000"      start="100000000"      size="900"
destination="0"/>
</entity>


<!-- == MOBILITY ======================================================
-->
<entity name="static" library="mobility_static">
      <default x="random" y="random" z="random"/>
</entity>
<entity name="static-pnc" library="mobility_static" >
    <default x="0.433" y="0.433" z="0"/>
</entity>

<entity name="static-dev1" library="mobility_static" >
    <default x="0.683" y="0.866" z="0"/>
</entity>
<entity name="static-dev2" library="mobility_static" >
    <default x="0.183" y="0.866" z="0"/>
</entity>

<entity name="static-dev3" library="mobility_static" >
    <default x="0" y="0.683" z="0"/>
</entity>
```

```
<entity name="static-dev4" library="mobility_static" >
    <default x="0" y="0.183" z="0"/>
</entity>
<entity name="static-dev5" library="mobility_static" >
    <default x="0.183" y="0" z="0"/>
</entity>
<entity name="static-dev6" library="mobility_static" >
    <default x="0.683" y="0" z="0"/>
</entity>
<entity name="static-dev7" library="mobility_static" >
    <default x="0.866" y="0.183" z="0"/>
</entity>
<entity name="static-dev8" library="mobility_static" >
    <default x="0.866" y="0.683" z="0"/>
</entity>

<!-- == Environment
====================================================== -->
<environment>
<interferences entity="interf"/>
<noise entity="white"/>
<modulation entity="o-qpsk"/>
</environment>

<!-- == Bundle ========================================================= -
->
<bundle name="emetteur1" worldsens="false" default="true" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>
  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr1"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr1">
    <down entity="mac"/>
</with>
</bundle>
<bundle name="emetteur2" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr2"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr2">
    <down entity="mac"/>
```

```
</with>
</bundle>

<bundle name="emetteur3" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr3"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr3">
    <down entity="mac"/>
</with>
</bundle>

<bundle name="emetteur4" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr4"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr4">
    <down entity="mac"/>
</with>
</bundle>

<bundle name="emetteur5" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr5"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr5">
    <down entity="mac"/>
```

```
</with>
</bundle>

<bundle name="emetteur6" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr6"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr6">
    <down entity="mac"/>
</with>
</bundle>

<bundle name="emetteur7" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr7"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr7">
    <down entity="mac"/>
</with>
</bundle>

<bundle name="emetteur8" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
<energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
  </antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

  <with entity="mac">
              <up entity="cbr8"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr8">
    <down entity="mac"/>
```

```
</with>
</bundle>

<bundle name="sink" worldsens="false" default="false" birth="0">
  <mobility entity="static"/>
  <energy entity="bat"/>
  <antenna entity="omnidirectionnal">
    <up entity="radio"/>
</antenna>

  <with entity="radio">
    <up entity="mac"/>
    <down entity="omnidirectionnal"/>
  </with>

 <with entity="mac">
       <up entity="cbr0"/>
 <down entity="radio"/>
  </with>
  <with entity="cbr0">
    <down entity="mac"/>
</with>

</bundle>

<!-- == Nodes ================================================== --
>
<node id="0" as="sink"/>
<node id="1" as="emetteur1"/>
<node id="2" as="emetteur2"/>
<node id="3" as="emetteur3"/>
<node id="4" as="emetteur4"/>
<node id="5" as="emetteur5"/>
<node id="6" as="emetteur6"/>
<node id="7" as="emetteur7"/>
<node id="8" as="emetteur8"/>
</worldsens>
```

Annexe II

Dans cette annexe, je vais montrer les changes que j'ai faits dans le fichier *linear.c* de WSNet pour modifier le comportement énergétique et ajouter les données qui sont publiés dans le datasheet du capteur CC2520.

```
/**
 *  \file   linear.c
 *  \brief  linearly decreasing battery
 *  \author Guillaume Chelius, modifications par Javier Cuadrado
 *  \date   2007
 **/
#include <include/modelutils.h>


/* ********************************************** */
/* ********************************************** */
model_t model =  {
    "Linearly decreasing battery",
    "Guillaume Chelius",
    "0.1",
    MODELTYPE_ENERGY,
    {NULL, 0}
};


/* ********************************************** */
/* ********************************************** */
int init(call_t *c, void *params) {
    return 0;
}

int destroy(call_t *c) {
    return 0;
}


/* ********************************************** */
/* ********************************************** */
struct nodedata {
    double energy;
    double initial;
    double tx;
    double rx;
    double idle;
};


/* ********************************************** */
/* ********************************************** */
int setnode(call_t *c, void *params) {
    struct nodedata *nodedata = malloc(sizeof(struct nodedata));
    param_t *param;

    /* default values */
    nodedata->energy  = 1000000;
    nodedata->initial = 1000000;
    nodedata->tx      = 1;
    nodedata->rx      = 1;
```

```
    nodedata->idle    = nodedata->rx;

   /* get parameters */
    das_init_traverse(params);
    //printf("valor inicial energia %f\n", nodedata->energy);
    while ((param = (param_t *) das_traverse(params)) != NULL) {
        if (!strcmp(param->key, "energy")) {
            if (get_param_double(param->value, &(nodedata->energy))) {
                goto error;
            }
        }
        if (!strcmp(param->key, "tx")) {
            if (get_param_double(param->value, &(nodedata->tx))) {
                goto error;
            }
        }
        if (!strcmp(param->key, "rx")) {
            if (get_param_double(param->value, &(nodedata->rx))) {
                goto error;
            }
        }
        if (!strcmp(param->key, "idle")) {
            if (get_param_double(param->value, &(nodedata->idle))) {
                goto error;
            }
        }
    }

    nodedata->initial = nodedata->energy;
    set_node_private_data(c, nodedata);
    return 0;

 error:
    free(nodedata);
    return -1;
}

int unsetnode(call_t *c) {
    free(get_node_private_data(c));
    return 0;
}


/* ********************************************** */
/* ********************************************** */
int bootstrap(call_t *c) {
    return 0;
}

int ioctl(call_t *c, int option, void *in, void **out) {
    return 0;
}


/* ********************************************** */
/* ********************************************** */
//changes faites pour l'energie consommée en transmission
void consume_tx(call_t *c, uint64_t duration, double txdBm) {
    struct nodedata *nodedata = get_node_private_data(c);
    nodedata->energy -= duration * (nodedata->tx*txdBm +
0.0774)*0.000000001;

    if (nodedata->energy <= 0) {
```

```
        nodedata->energy = 0;
        node_kill(c->node);
    }
    return;
}

void consume_rx(call_t *c, uint64_t duration) {
    struct nodedata *nodedata = get_node_private_data(c);
//pas de modification ici, le valeur est donné dans le xml (nodedata-
>rx)
    nodedata->energy -= duration * nodedata->rx*0.000000001;
    if (nodedata->energy <= 0) {
        nodedata->energy = 0;
        node_kill(c->node);
    }
    return;
}

void consume_idle(call_t *c, uint64_t duration) {
    struct nodedata *nodedata = get_node_private_data(c);
    nodedata->energy -= duration * nodedata->idle;
    if (nodedata->energy <= 0) {
        nodedata->energy = 0;
        node_kill(c->node);
    }
    return;
}

void consume(call_t *c, double energy) {
    struct nodedata *nodedata = get_node_private_data(c);
    nodedata->energy -= energy;
    if (nodedata->energy <= 0) {
        nodedata->energy = 0;
        node_kill(c->node);
    }
    return;
}

double energy_consumed(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    return nodedata->initial - nodedata->energy;
}

double energy_remaining(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    return nodedata->energy;
}

double energy_status(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    double status = nodedata->energy / nodedata->initial;
    if (nodedata->energy <= 0) {
        return 0;
    } else if ((status >= 0) && (status <= 1)) {
        return status;
    } else {
        return 0;
    }
}


/* ********************************************** */
/* ********************************************** */
```

```
energy_methods_t methods = {consume_tx,
                            consume_rx,
                            consume_idle,
                            consume,
                            energy_consumed,
                            energy_remaining,
                            energy_status};
```