MÉMOIRE

présenté en vue d'obtenir

LE DIPLÔME D'INGÉNIEUR I.I.E.

Rapport Final

Julien VAUDOUR

Élaboration de couches MAC et réseau pour un réseau de capteurs

Directeur du stage : M. Michel MAROT, enseignant chercheur
Institut National des Télécommunications
Département Réseaux et Services de Télécommunications
Laboratoire SAMOVAR - UMR 5157 CNRS

Soutenu le 29 juin 2006 devant le jury
D. ROUSSEL, Maître de Conférences
Y. GHAMRI, Maître de Conférences

À mes parents et amis

Remerciements

Je remercie Monique Becker, Michel Marot et Vincent Gauthier pour leur encadrement de qualité et pour leur sympathie. Je les remercie aussi, ainsi que Alexandre Delye de Mazieux, pour m'avoir permis de co-publier un état de l'art sur les réseaux de capteurs sans fil [6] avec eux et André-Luc Beylot, Riadh Dhaou, Rahim Kacimi de École Nationale Supérieure d'Électrotechnique, d'Électronique, d'Informatique, d'Hydraulique et des Télécommunications.

Je remercie également les personnels administratifs et de la direction de l'Institut National des Télécommunications et de l'Institut d'Informatique d'Entreprise sans qui ce stage n'aurait pas été possible.

De même, je remercie David Roussel et Yacine Ghamri Doudane pour avoir accepté d'être les examinateurs de ce mémoire.

Enfin je remercie Sandoche Balakrichenan, Mahendiran Prathaban ainsi que les autres thésards et stagiaires pour leur sympathie et la bonne ambiance qu'ils mettent.

Conservatoire National des Arts et Métiers

INSTITUT d'INFORMATIQUE d'ENTREPRISE

FICHE SIGNALÉTIQUE

Mémoire d'Ingénieur I.I.E.

Élaboration de couches MAC et réseau pour un réseau de capteurs

Auteur : Julien Vaudour

Directeur de stage : M. Michel MAROT, enseignant chercheur
Institut National des Télécommunications
Département Réseaux et Services de Télécommunications
Laboratoire SAMOVAR - UMR 5157 CNRS

Descriptif : Les réseaux de capteurs sans fils sont soumis à des contraintes, surtout en termes de consommation d'énergie. Ce mémoire présente des mécanismes concernant la couche MAC permettant d'économiser l'énergie.

Résumé

Les réseaux de capteurs sans fils apportent des perspectives intéressantes : celles de réseaux autonomes et spontanés. Un des principaux critères de performances pour ce type de réseaux est la consommation d'énergie. En effet pour faire des capteurs peu couteux, on veut qu'ils aient une durée de vie de l'ordre de quatre ans avec une pile de type AAA. Les réseaux sans fils "classiques" tel que le Wifi ne permettent pas de répondre à cette contrainte. Il y a donc une forte activité autour de la recherche de nouvelles solutions spécifiques aux réseaux de capteurs sans fils. Ce mémoire s'inscrit dans le cadre d'un projet du Réseau National de Recherche en Télécommunications ayant pour but, dans un premier temps, de proposer un réseau de capteurs de température pour surveiller une chaîne du froid.

BlueTooth étant un standard assez utilisé dans d'autres milieux, il paraissait intéressant de l'étudier pour une utilisation dans les réseaux de capteurs sans fils. L'utilisation courante de BlueTooth étant dans de très petits réseaux contenant moins d'une dizaine d'équipements, plusieurs solutions permettant de dépasser cette limite ont été étudiées. Cependant, pour des raisons physiques liées à BlueTooth, malgrès ces solutions, on est limité à moins d'une centaine d'équiments actifs dans le réseau. Or, pour surveiller une chaîne du froid, la solution retenue est de mettre au moins un capteur par pallette. A l'échelle d'un entrepôt, on a donc besoin de plusieurs milliers de capteurs. BlueTooth n'est donc pas adapté aux besoins du projet et a donc été finalement écarté.

On s'est orienté vers des solutions compatibles ZigBee. ZigBee s'appuie sur le standard 802.15.4 et offre une multitude de services au niveau applicatif. Il a été spécifiquement développé pour les réseaux de capteurs sans fils. Le standard 802.15.4 définit pour la couche MAC, deux topologies : une topologie en étoile et une topologie point à point. La description de la première est très précise dans le standard alors que l'on est plus libre pour la topologie point à point. Cette dernière est celle qui nous intéresse. On s'est donc attaché à l'étude des différentes solutions qui ont été proposées.

Les principales sont SMAC, TMAC et BMAC. SMAC et TMAC permettent de définir une alternance de phases de sommeil et de réveil pour permettre d'économiser l'énergie. De ce fait il faut que les nœuds se synchronisent pour pouvoir communiquer. BMAC, en introduisant le concept d'écoute à basse consommation, permet d'éviter d'avoir à se synchroniser.

À partir de l'étude de ces trois protocoles nous avons introduit une évolution possible de BMAC qu'il reste encore à évaluer.

Table des matières

Introduction

Les réseaux de capteurs sans fil apportent une perspective intéressante : celle de réseaux capables de s'autoconfigurer, de se gérer sans qu'il y ait besoin d'interventions humaines. De plus, les critères de performance pour un réseau de capteurs diffèrent de ceux des réseaux classiques et donc les solutions à apporter sont nouvelles. En effet, les capteurs sans fils ont vocation à devenir des objets "banaux" et donc doivent pouvoir s'utiliser facilement. Le réseau doit devenir transparent pour l'utilisateur.

Les réseaux de capteurs trouvent des applications dans des réseaux sans fil et sans infrastructure ne nécessitant pas un débit élevé tel que la domotique ou l'agriculture intelligente. La réalisation de ce type de réseau requiert la mise en œuvre de techniques développées pour les réseaux adhoc ; cependant, la plupart des protocoles développés pour ceux ci ne sont pas transposables tels que aux réseaux de capteurs.

L'objet de ce document concerne principalement les couches MAC pour les réseaux de capteurs sans fils en s'intéressant principalement à la question de l'économie d'énergie. À partir d'une analyse des principaux protocoles existants, un nouveau protocole a été conceptualisé.

Dans un premier temps, nous allons voir comment utiliser BlueTooth dans un réseau de capteurs. Mais BlueTooth ne répondant pas aux contraintes du projet sur lequel travaille l'équipe du laboratoire notamment à cause du dimensionnement du problème, nous verrons les solutions compatibles ZigBee, qui a justement pour vocation les réseaux de capteurs sans fil. Enfin, à partir de notions tirées de ces différentes couches MAC, une nouvelle couche MAC est proposée.

Chapitre 1

Contexte

1.1 Présentation de l'INT

Mon stage a lieu à l'Institut National des Télécommunication (*INT*). L'INT est un établissement d'enseignement supérieur et de recherche situé à Évry et membre du Groupe des Écoles de Télécommunication (*GET*) qui regroupe également *Télécom Paris*, l'*ENST Bretagne*, *Télécom Lille 1* et *EURECOM*.

1.2 Présentation du département RST

Créé le 1er Janvier 1994, le département "Réseaux et Services des Télécommunications" (RST) est l'un des huit départements de l'Institut National des Télécommunications et est dirigé par M. Gérard HEBUTERNE.

Le Département assure, au sein de l'INT, la compétence dans le domaine des réseaux de télécommunications, fixes ou mobiles : analyse des performances et qualité de service, optimisation, techniques et services sans fil, protocoles pour les réseaux circuit ou paquets, domotique, etc.

1.3 Présentation du laboratoire SAMOVAR

J'effectue mon stage au sein de l'équipe ARMOR (**AR**chitecture et **MO**délisation de **R**éseaux) du laboratoire SAMOVAR ("**S**ervices répartis **A**rchitectures, **MO**délisation, **V**alidation, **A**dministration des **R**éseaux").

Le laboratoire SAMOVAR est une **U**nité **M**ixte de **R**echerche associée au CNRS et est dirigé par Mme Monique BECKER.

L'équipe ARMOR étudie les problèmes de qualité de service tant dans le réseau d'accès (y compris mobiles) que dans le réseau de cœur (ou fixe). Les

travaux de recherche de l'équipe s'organisent autour de trois axes majeurs :
- Architectures de réseaux (fixes et mobiles)
- Ingénierie du trafic
- Méthodes de modélisation et d'analyse de performances

1.4 Équipe de travail

L'équipe que j'ai intégrée travaille autour du projet *Capteurs*. Elle est composée (personnes travaillant sur le projet *Capteurs*) de :
- Monique BECKER, directrice de SAMOVAR
- Michel MAROT, encadrant de projet
- Vincent GAUTHIER, doctorant
- Alexandre DELYE de CLAUZADE de MAZIEUX, doctorant

Monique Becker encadre, avec mon maître de stage, Michel Marot, Vincent Gauthier et Alexandre Delye pour leurs thèses. Mon stage est en partie encadré par Vincent Gauthier, qui prépare une thèse sur l'utilisation des simulations multiniveaux dynamiques pour l'étude de la QOS dans les réseaux Ad Hoc (MANET) et la conception et la validation d'algorithmes de routage dans certains de ces réseaux hétérogènes.

Chapitre 2

Présentation du sujet

2.1 Le projet *Capteurs*

2.1.1 Présentation

L'unité de recherche que j'ai intégrée travaille actuellement sur le projet *Capteurs*. Ce projet est financé par le **R**éseau **N**ational de **R**echerche en **T**élécommunications. Il s'agit, dans un premier temps, de proposer un réseau de capteurs sans fil pour surveiller la chaîne du froid. La solution actuellement envisagée est d'utiliser un capteur par palette.

Ce projet est divisé en plusieurs parties : réseau, sécurité, etc. Notre équipe travaille sur la partie réseau : élaboration des couches MAC et réseaux, définition d'un protocole de routage, etc. Il faut concevoir le mode de communication entre les capteurs contrôlant la température et expérimenter le réseau conçu.

2.1.2 Les Problématiques

Les différentes problématiques autour des réseaux de capteurs concernent l'autoconfiguration d'un tel réseau, dans un contexte de mobilité et l'économie des ressources énergétiques de ses nœuds (l'objectif est qu'un capteur tienne quatre ans avec une pile AAA). Pour économiser au mieux la batterie d'un nœud, il faut que ses transmetteurs radio soient coupés la plupart du temps. Ceci pose le problème de la synchronisation des nœuds et la répartition des périodes de réveil.

2.2 Programme d'initiative *Réseaux autonomes et spontanés*

Le projet *Capteurs* est dans la logique du Programme d'Initiative *Réseaux autonomes et spontanés* du GET qui font partie de ses axes de développement pour l'avenir.

2.2.1 Description du Programme d'Initiative

Les principes de gestion centralisée s'appliquent de plus en plus difficilement aux nouveaux types de réseau, faisant intervenir de grandes quantités de nœuds hétérogènes, dépendant d'entités multiples, et qui souvent fonctionnent en contexte de mobilité. Pour y remedier, ont été introduites les notions de réseaux « autonomes » et de réseaux « spontanés ». Un réseau autonome est un réseau qui parvient à offrir un service en s'auto-configurant et en coopérant directement sans intervention extérieure d'un quelconque système de gestion. La notion de réseau spontané implique de plus que, parmi une grande quantité d'éléments capables de s'interconnecter, certains d'entre eux « décident » de se configurer entre eux pour former un réseau qui fonctionne de manière autonome ; il y a donc création d'une communauté spontanée de nœuds communicants, puis auto-configuration de cette communauté qui forme alors un réseau autonome pour la fourniture d'un service donné.

2.3 Les réseaux de capteurs

Les réseaux de capteurs, tout comme les réseaux Ad Hoc, souffrent de nombreuses limitations en terme de performance, du fait du manque d'infrastructure et de la nature du medium sans fil.
Actuellement une grande partie de la recherche dans les réseaux porte sur les réseaux mobiles et sur les problématiques liées aux réseaux Ad Hoc car elles couvrent tout le domaine lié aux réseaux (routage, gestion de la mobilité, autoconfigurabilité, qualité de service, etc.).

Les réseaux de capteurs trouvent des applications dans des domaines ne nécessitant pas un débit élevé. La réalisation d'un réseau de capteurs réutilise des techniques développées pour les réseaux Ad Hoc mais en ajoutant des spécificités qui nécessitent une autre approche du problème :

- faible consommation électrique
- faible débit
- autoconfigurabilité

– faible puissance de calcul
– forte densité des nœuds
– réseau redondant et collaboratif
– tolérance aux erreurs

2.4 Notion de *cluster* [5]

L'agrégation de nœuds en clusters permet de réduire la complexité des algorithmes de routage, d'optimiser la ressource medium en la faisant gérer localement par un chef de cluster (le *clusterhead* aussi appelé *caryomme*), de faciliter l'agrégation des données, de simplifier la gestion du réseau et en particulier l'affectation d'adresses, d'optimiser les dépenses d'énergie, et enfin de rendre le réseau plus scalable. L'utilisation de clusters permet aussi de stabiliser la topologie et la gestion du réseau si les tailles de clusters sont grandes par rapport aux vitesses de nœuds mais cela ne fonctionne que dans le cas d'une faible mobilité.

2.5 Généralités sur les couches MAC

Il existe deux grandes familles de protocoles MAC :
– les protocoles à contention
– les protocoles ordonnancés

Avec un protocole à contention, lorsqu'un nœud veut émettre un paquet, il regarde si le medium est libre. Si oui, il émet le paquet. Sinon il attend que le medium soit libéré. En fonction des mécanismes utilisés, il y a des risques de collision, ce qui représente du gaspillage d'énergie, ressource qu'il est important de préserver pour un réseau de capteurs.

Avec un protocole ordonnancé, chaque nœud se voit attribuer une fenêtre temporelle durant laquelle il pourra émettre des paquets. On évite ainsi les risques de collisions. Toutefois ceci nécessite un coordinateur et la synchronisation des nœuds.

Ces deux familles de protocoles ne sont pas totalement disjointes et certains protocoles utilisent des propriétés de ces deux familles.

2.6 Objet du stage

L'objet de ce stage est de définir les couches MAC et réseau pour un réseau de capteurs, le but étant de satisfaire au mieux les contraintes liées à un

tel réseau. Dans un premier temps, il s'agit de mener une étude bibliographique sur les différentes solutions pour la couche MAC pour un réseau de capteurs afin de déterminer les solutions qui peuvent être retenues. Il s'agit ensuite soit de mener une étude comparative par simulation de différentes solutions.

Dans un premier temps, l'étude bibliographique devait porter essentiellement sur BlueTooth mais finalement, suite à cette étude, la solution Bluetooth a été écartée car BlueTooth n'est pas adapté aux besoins du réseau que l'on veut mettre en place (trop faible nombre de nœuds et consommation excessive). L'étude s'est donc ensuite orientée vers des solutions compatibles ZigBee. Nous allons effectuer des simulations pour comparer ces différentes solutions.

Chapitre 3

BlueTooth pour les réseaux de capteurs

On aimerait faire un réseau de capteurs dans lequel tout ajout de nouveau matériel soit pris en compte même s'il ne provient pas du même fabricant. BlueTooth étant un standart largement déployé dans d'autres domaines et ayant fait l'objet de nombreux articles pour les réseaux de capteurs, sa mise en œuvre semblait être une idée intéressante.

3.1 La norme BlueTooth

La norme Bluetooth est définie dans [7]. Bluetooth opère sur une bande de 83,5 Mhz de large. Dans cette bande, 79 canaux sont définis. Un canal Bluetooth occupe succesivement ces canaux suivant une séquence aléaoire : à chaque tranche de temps (*time slot*) de 625 μs (1/1600 s), on réalise un saut de fréquence. Cependant l'émission d'un paquet peut durer 1, 3 ou 5 *time slot* et on ne change pas de canal radio pendant l'émission d'un paquet. Après l'émission d'un paquet qui prend plusieurs *time slots*, on reprend la séquence là où on en serait si on n'avait émis que des paquets qui prennent un seul *time slot*.
Les dispositifs utilisant un même canal Bluetooth forment un *piconet*, formé d'un maître et d'esclaves (voir figure 3.1). L'adresse du maître (BD_ADDR) détermine la séquence des sauts de fréquence. Un *piconet* peut comporter jusqu'à 255 esclaves mais seuls 7 peuvent être actifs, les autres sont "parqués". Il n'y a pas de communication maître-maître ni esclave-esclave.

Il existe quatre types d'adresses Bluetooth :
– **BD_ADDR** (Bluetooth Device Address) : Chaque objet Bluetooth possède une adresse unique sur 48 bits

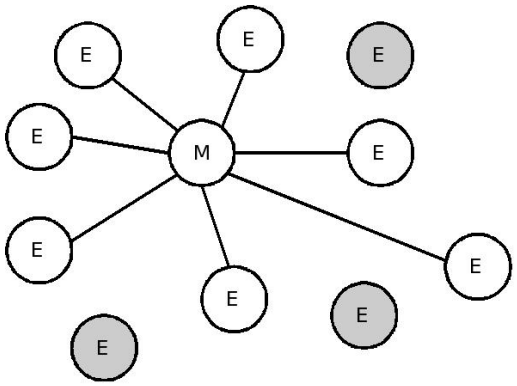

FIG. 3.1 – Piconet

- **AM_ADDR** (Active Member Address) : Cette adresse sur 3 bits identifie les esclaves actifs d'un canal Bluetooth
- **PM_ADDR** (Parked Member Address) : Cette adresse sur 8 bits identifie les esclaves "parqués".
- **AR_ADDR** (Access Request Address) : Cette adresse permet à un esclave "parqué" de connaitre le *time slot* qu'il est autorisé à utiliser pour demander au maître à être déparqué

Les principaux états d'un dispositif BlueTooth dans un *piconet* sont :

- **Active** : esclave actif, décode les adresses pour voir si les paquets lui sont destinés
- **Sniff** : esclave toujours synchronisé avec le maître. N'écoute le réseau que pendant certains *time slots* spécifiés. Mise en veille périodique et automatique
- **Hold** : l'esclave actif d'un piconet stoppe sa radio, conserve son adresse de membre actif et reste synchronisé avec le maître et se place en mode basse consommation pour un intervalle de temps déterminé, après quoi il redevient actif
- **Park** : esclave toujours synchronisé avec le maître mais ne participe pas au trafic. Il n'a pas d'adresse de membre actif. Le maître du *piconet* lui envoie à intervalles de temps réguliers des paquets de synchronisation et, à ces instants, il peut demander à redevenir actif.

L'interconnexion de plusieurs *piconets* est appelée un *scatternet*. L'interconnexion de 2 *piconets* se fait au moyen d'un nœud qui soit est esclave dans les deux *piconets* soit est esclave dans l'un des *piconets* et maître dans l'autre (voir figure 3.2). Un nœud ne peut être maître que dans un seul piconet.

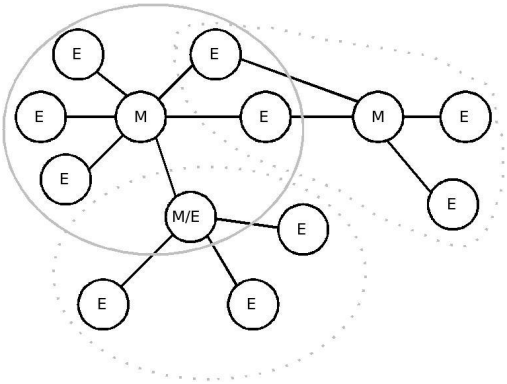

FIG. 3.2 – Scatternet

3.1.1 La pile protocolaire

La figure 3.3 présente la pile BlueTooth.

La couche baseband gère les différents types de liaison Bluetooth :
- **Liaison synchrone (SCO)** : ce type de liaison offre un débit synchrone de 64 kb/s dans les deux directions. En théorie, Bluetooth permet 3 liaisons synchrones simultanées.
- **Liaison asynchrone (ACL)** : ce type de liaisons offre un débit maximal de 732 kb/s pour la transmission de données sans garantie de délai.

La couche LMP (Link Manager Protocol) s'occupe de l'établissement, du contrôle et de la sécurité de la liaison. Elle s'occupe égalemenent de placer les dispositifs dans les différents modes (park, hold, sniff et actif).

La couche L2CAP (Logical Link Control and Adaptation Layer Protocol) permet aux protocoles de niveau supérieur d'échanger des paquets de données jusqu'à 64 kbits par des liaisons en mode connecté ou non. Elle s'occupe de la segmentation et du réassemblage des paquets, ainsi que du multiplexage des protocoles.

3.1.2 Établissement de la connection

On ne s'intéresse ici qu'au processus d'établissement d'une connexion asynchrone. L'établissement de la connexion se fait à travers deux procédures : inquiry et page. L'inquirying n'est pas nécessaire si la station connait déjà certains paramètres.

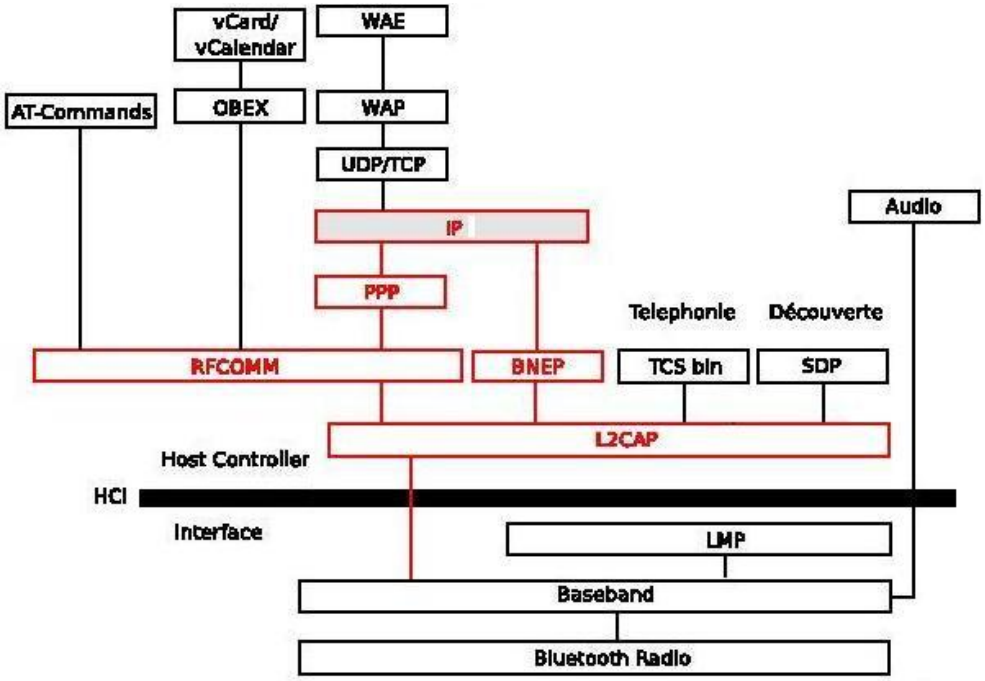

FIG. 3.3 – couches Bluetooth.

3.1.2.1 La procédure d'*inquiry*

La procédure d'inquiry permet à un dispositif de découvrir quels autres dispositifs sont dans la zone de portée, et lui affecte une adresse et une horloge.
La source envoie des paquets de recherche (inquiry packets) à différentes fréquences (32), suivant une séquence particulière, l'*inquiry hop sequence*, et qui doit recevoir une réponse (*inquiry reply*).
La destination des paquets d'inquiry doit être dans l'état d'*inquiry scan* pour pouvoir les recevoir. Dans cet état, la destination écoute à intervalles de temps réguliers (toutes les 1,28 secondes) une fréquence particulière, pendant un temps suffisant pour que cette fréquence soit balayée par la séquence suivie par la source (soit au maximum 32 *time-slots*, soit 20 ms). Lorsqu'elle a détecté un *inquiry packet*, la destination se place alors dans l'état d'*inquiry response* et envoie une réponse à la source, sous forme d'un paquet contenant l'adresse et des informations sur l'horloge de la destination.

3.1.2.2 La procédure de *paging*

Elle établit la connexion réelle entre les deux dispositifs. En règle générale, cette procédure suit la procédure d'*inquiry*, mais elle peut aussi être

exécutée seule si les dispositifs connaissent mutuellement leurs adresses. La connaissance préalable de l'horloge du maître accélère également la procédure. C'est l'unité qui veut se connecter qui pilote la procédure.
Le dispositif source qui est en mode page, "page" l'autre (la destination), qui est en mode page_scan. La destination reçoit un paquet avec son code d'accès DAC, et passe en mode slave_response. Elle répond par un paquet identique à la source qui passe en mode master_response. La source envoie un paquet qui comporte son adresse de maître BD_ADDR et son horloge, ce qui va permettre à la destination de déterminer la séquence de sauts de fréquence, et la phase. La destination répond par un paquet identique au premier. Pendant toutes ces étapes, source et destination suivent toujours la séquence particulière de sauts de fréquence liée à la procédure de paging. Source et destination passent dans l'état connecté, où ils suivent alors la séquence de sauts de fréquence liée à la connexion.

3.2 La formation du scatternet

Pour la formation du scatternet, plusieurs solutions sont proposées :
- formation immédiate
  - sans esclaves parqués
  - avec esclaves parqués
- formation à la demande

Pour former le scatternet, un nœud peut endosser un double rôle, c'est à dire qu'il peut être maître dans un piconet et esclave dans un ou plusieurs autres piconets. Cependant, un nœud ne peut occuper qu'un seul canal à la fois. Donc pour passer d'un piconet à un autre, un nœud doit se placer sur le canal BlueTooth du piconet sur lequel il souhaite aller et se synchroniser sur ce piconet. Donc quand un nœud qui est maître dans un piconet endosse un double rôle, le piconet dont il est maître est inutilisable quand il bascule dans un autre piconet.
De plus, plusieurs articles ont souligné le fait qu'un nœud ne devait pas appartenir à plus de deux piconets sous peine d'une dégradation importante des performances. Ceci est également du au fait qu'un nœud ne peut écouter qu'un seul canal à la fois et au temps de synchronisation lors du passage d'un canal à un autre.

3.2.1 formation immédiate sans esclaves parqués

L'avantage de ne pas avoir d'esclaves parqués est que tous les nœuds peuvent participer au trafic. Plusieurs solutions ont été proposées. En voici quelques-unes (cette liste n'est pas exhaustive) :

3.2.1.1 BlueTree

Dans BlueTree [14], il s'agit de créer un arbre hiérarchique, c'est à dire que chaque maître du niveau *n* est également esclave du niveau *n-1*. L'avantage majeur de cette solution est qu'elle permet d'utiliser un algorithme de routage simplissime dans le cas où il faut juste remonter les informations vers la station les collectant. En effet dans ce cas, il suffit que chaque nœud, dans le piconet où il est esclave, envoie les informations à son maître (ce qui est d'ailleurs la seule chose qu'il sait faire) et que la station collectante soit la racine de l'arbre. L'autre avantage de cette architecture est que chaque maître peut agréger les données qu'il reçoit avant de les retransmettre. Les gros désavantages de cet algorithme sont que l'ajout d'un nouveau nœud est difficile et que l'on a une perte de performance à cause du double rôle de la plupart des nœuds.

3.2.1.2 Bluenet

Bluenet [12] est basé sur le concept de graphe de visibilité. Dans une premiere phase, chaque nœud collecte des informations sur ses voisins (au sens spatial) en état d'*inquiry* pour former un graphe de visibilité. Ensuite il *page* ses voisins au hasard. À la fin de cette phase, les piconets sont formés. Ensuite dans une deuxième phase, chaque nœud relance une phase d'*inquiry* pour se connecter à d'autres nœuds. Ainsi, à la fin d'une troisième phase, le scatternet est formé.
L'inconvénient de cet algorithme est qu'il fait appel deux fois à la procédure d'*inquiry* qui est assez consommatrice d'énergie. De plus l'ajout d'un nouveau nœud n'est pas aisé.

3.2.2 formation immédiate avec esclaves parqués

L'idée est d'avoir un maximum d'esclaves parqués En effet le mode *Park* est celui qui consomme le moins d'énergie. De plus, étant donné qu'un maître peut avoir jusqu'à 255 esclaves parqués, il est facile de rajouter un nouveau nœud. Les deux problèmes majeurs de cet algorithme sont la connectivité et le réveil des nœuds. En effet, si un nœud parqué a des informations à envoyer, il faut que le maître puisse mettre en sommeil un esclave actif. Il faut donc prendre des précautions pour ne pas parquer un esclave qui sert de passerelle vers un autre piconet, ou sinon, il faut trouver un mécanisme pour pouvoir mettre une nouvelle passerelle vers ce piconet. De plus chaque mise en sommeil ou réveil de nœud peut avoir une incidence sur les tables de routages.

3.2.3 formation à la demande

On voit bien l'incidence de la formation du scatternet sur le routage. Or, la formation initiale ne prend pas (ou pas assez) compte du routage. Avec la formation à la demande, le scatternet est créé en fonction des destinations que l'on veut atteindre. Ainsi la formation du réseau et le routage ne sont pas séparés, d'où l'idée de formation à la demande [9].

Lorsqu'un nœud veut envoyer des paquets à un autre, il regarde s'il connaît une route vers cette destination. Si c'est le cas, il peut envoyer ses paquets. Sinon il émet un paquet de découverte de route (RDP) qui est envoyé à tout le réseau. Lorsque la destination reçoit le paquet, elle envoie un paquet de réponse de route (RRP) en utilisant la route découverte. Avec le passage du paquet RRP, des liens Bluetooth point-à-point sont créés pour connecter les nœuds faisant partie de cette nouvelle route. En même temps, les tables de routages de ces nœuds sont remplis avec les informations nouvellement découvertes. Quand la source reçoit le paquet RRP, elle peut donc émettre ses paquets.

Pour diffuser le paquet, trois mécanismes sont proposés :
- le *broadcast* L2CAP
- le *broadcast* LMP
- le *broadcast* EID

Le *broadcast* L2CAP est la manière conventionnelle. La source engage un processus d'inquiry pour découvrir ses voisins. Ensuite elle "page" ses voisins pour établir une connexion point-à-point au niveau LMP. Ensuite le paquet RDP peut être broadcasté. Enfin les liens avec les voisins sont supprimés pour qu'ils puissent à leur tour broadcaster le paquet.
Le *broadcast* LMP est assez semblable. La différence est qu'après la phase d'inquiry, la source "page" ses voisins un à un et envoie le paquet RDP par le protocole LMP juste après le paging pour relibérer le nœud tout de suite. C'est donc une amélioration du processus précédent.
Le *broadcast* EID va encore plus loin. En effet on met le paquet RDP dans le message d'inquiry.

Dans le réseau ainsi formé, on alterne sur chaque route, un maître, un esclave. Cependant, il peut y avoir certains nœuds qui doivent assumer un double rôle, comme par exemple quand deux routes se croisent.

3.3 Le routage dans BlueTooth

Dans BlueTooth, un piconet est identifié par l'adresse BlueTooth du maître. Dans la suite de cette section, on appelle *MacAddr* l'adresse de membre actif d'un esclave dans le piconet. L'adresse 000 est réservée pour le broadcast à l'intérieur du piconet. Un paquet de niveau 2 est composé d'un champ *Access Code* sur 72 bits, d'un en-tête sur 54 bits et d'une charge utile comprise ente 0 et 2745 bits. Le champ *Access Code* est utilisé pour la synchronisation du canal, les procédures d'inquiry et de paging. Un des champs de l'entête est l'adresse MacAddr. Le maître du piconet peut envoyer un paquet à un esclave en remplissant le champ MacAddr avec l'adresse de membre actif du destinataire ou broadcaster sur le piconet en mettant ce champ à 000. Un paquet envoyé d'un esclave au maître contient l'adresse de membre actif de cet esclave pour permettre au maître de l'identifier. Il n'y a donc pas de possibilité au niveau 2 pour un esclave d'indiquer au maître que le paquet est à destination d'un autre esclave du piconet.

Le protocole de routage utilise le niveau 3. Pour indiquer au maître du piconet qu'un paquet reçu d'un esclave doit être transmis à un autre esclave du piconet, il faut mettre le flag *forwarding* (FF) dans l'entête de niveau 3 à 1 et mettre dans le champ DA (adresse de destination) la MacAddr de l'esclave à qui ce paquet est destiné. Si le flag FF est à 0, alors le paquet est à destination du maître.
Une des solutions pour router dans le scatternet peut être le routage par la source. En effet le routage en maintenant des tables de routages dans les nœuds servant à interconnecter plusieurs piconets présente de nombreux inconvénients. En effet nous sommes dans un contexte de mobilité et des nœuds peuvent tomber en panne. Maintenir ces tables est donc assez complexe et dans un réseau de capteurs, les ressources sont assez limitées.

Cependant, inclure la liste des identifiants de nœuds dans le paquet ne serait pas très approprié car pour chaque saut on aurait besoin de 51 bits (48+3). Un des principaux algorithmes de routage dans un scatternet BlueTooth déjà formé est le *Routing vector method*. Pour cette méthode, on introduit la notion de *LocId*. Au niveau de chaque nœud qui sert de pont entre plusieurs piconets, on identifie localement les différents piconets auxquels le nœud appartient par un identifiant sur 3 bits : le *LocId*. Le LocID 000 designe le pont lui-même.
Ainsi, on a besoin de 6 bits par saut (3+3). La route est decouverte par un protocole de découverte de route. La route à utiliser sera dénotée par une séquence : Locid, MacAddr, LocId, MacAddr, etc.

3.4 Les limites de BlueTooth

Les principaux inconvénients de BlueTooth sont :
- piconets limités à 8 membres actifs
- risques d'interférences entre les différents piconets

Il est possible que pendant un time slot, plusieurs piconets se retrouvent à utiliser le même canal radio. Pour n piconets en présence, la probabilité de non-interférence est d'environ : $79!/((79^n) * ((79-n)!))$. Pour 8 piconets, la probabilité de non interférence vaut environ 0,69 et pour 9 piconets elle vaut 0,62. Ce calcul est approché car il ne prend pas en compte que les différents piconets ne sont pas forcément synchronisés entre eux. En général, au delà de 8 piconets, on considère la qualité du réseau trop mauvaise. Ceci nous limite donc à un nombre de nœuds inférieur à 64, ce qui est très en dessous de nos besoins si on veut mettre un capteur par palette.
Pour ces raisons, suite à la présentation faite sur BlueTooth pour le groupe de travail du projet Capteurs, l'équipe travaillant sur le projet, a décidé de rejeter l'idée d'utiliser BlueTooth pour les réseaux de capteurs.
À la place, l'équipe préfère s'intéresser à des solutions compatibles ZigBee.

Chapitre 4

Les systèmes d'exploitation pour les capteurs

ZigBee offre un ensemble de services pour réseaux sans fil au dessus de l'IEEE 802.15.4. ZigBee définit donc les couches 3 et supérieures.
Plusieurs systèmes d'exploitation ont été développés pour répondre aux contraintes particulières des réseaux de capteurs. Celui qui s'est imposé comme la référence est celui développé à l'Université de Berkeley : TinyOS.

4.1 TinyOS

TinyOS est un système d'exploitation open source conçu pour les capteurs sans fils et développé par l'Université de Berkeley. Il est basé sur une architecture à base de modules : pilotes pour des capteurs, des protocoles réseau et des services distribués. Les composants sont programmés en nesC, un langage de programmation dérivé du C adapté aux faibles ressources physiques des capteurs. Un certain nombre de plateformes sont déjà directement programmables comme par exemple les tmote (moteiv) ou les MicaZ (Crossbow) (ces deux modèles sont compatibles ZigBee). TOSSIM est un simulateur de capteurs pour les programmes TinyOS. Tout programme en nesC peut être compilé de manière à être exécuté dans TOSSIM, ce qui permet de simuler le comportement d'un ou plusieurs capteurs ainsi programmés. Cependant TOSSIM ne permet pas de simuler plusieurs programme nesC en même temps. On ne peut donc pas simuler un réseau de capteurs ayant des nœuds hétérogènes.
Il est important de noter que pour l'instant, TinyOS n'implémente pas le standard IEEE 802.15.4 en entier. Pour les capteurs utilisant des chips radio compatibles ZigBee, TinyOS n'offre que la couche physique de ce standard et pour la couche MAC utilise BMAC.

4.2 MOS

MOS [3] est un système d'exploitation open-source pour les capteurs développé par le *MANTIS group* à l'Université du Colorado. MOS est développé en C. Il supporte les plateformes de la famille MICA et de la famille Telos.

4.3 SOS

SOS [4] est un système d'exploitation open-source pour les réseaux de capteurs développé par le laboratoire réseaux et systèmes embarqués de l'Université de Los Angeles. Il est basé sur une architecture modulaire. SOS est écrit en C. Il supporte les plateformes de la famille MICA. Avrora est un simulateur pour les programmes SOS. SOS a l'avantage de posséder une implémentation complète de la topologie en étoile du standard 802.15.4.

4.4 Conclusion

TinyOS est de loin le système d'exploitation le plus utilisé pour les réseaux de capteurs sans fils. TinyOS utilise principalement SMAC comme couche MAC et utilise BMAC pour les capteurs compatibles ZigBee. De plus l'équipe du projet Consensus [2] a porté TMAC dans la branche de développement de TinyOS. De plus, par rapport à MOS et SOS, il a l'avantage d'être écrit dans un langage spécifique optimisé pour les capteurs.

Chapitre 5

Les différentes couches MAC

5.1 La norme 802.15.4 (LR-WPAN)

La norme 802.15.4 ([8] et [5]) est la norme retenue par ZigBee Alliance.

La norme IEEE 802.15.4 a été spécialement définie en fonction des caractéristiques des réseaux de capteurs, un faible débit et une faible consommation électrique. Elle décrit le fonctionnement de la couche physique, et de la couche MAC. Pour la couche physique deux bandes de fréquence ont été retenues la bande 2.4 GHz et la bande 868-915 MHz, c'est au total 27 canaux de communication avec 3 différents débits possibles soit 16 canaux à 250 kbps dans la bande 2.4 GHz, 10 canaux à 40 kbps dans la bande 915 MHz, et 1 canal à 20 kbps dans la bande 868 MHz. Les réseaux qui supporteront la norme 802.15.4 pourront librement choisir d'utiliser l'un des 27 canaux utilisables en fonction de sa disponibilité et du débit recherché.

Pour la couche MAC, deux topologies sont supportées par le protocole 802.15.4, la topologie en étoile, et la topologie point à point. Dans la topologie en étoile, les communications s'établissent directement entre le nœud central (coordinateur) et les capteurs, le coordinateur étant le nœud qui initie et gère les communications dans le réseau. La topologie point à point (Peer to Peer) permet à chaque nœud du réseau de communiquer avec n'importe quel autre nœud, ce qui permet de réaliser des réseaux ayant une architecture beaucoup plus complexe.
La topologie point à point nécessite l'utilisation d'un protocole de routage qui doit être géré par la couche réseau, mais cette partie n'est pas définie dans le protocole IEEE 802.15.4.

Le standard LR-WPAN permet l'utilisation de deux méthodes d'accès au canal de communication ; la première utilise une méthode de type CSMA-

CA similaire à celle utilisée dans les réseaux 802.11, la deuxième est une méthode de type slotted CSMA-CA (accès multiples durant un intervalle de temps). La méthode Slotted CSMA-CA utilise des supertrames ; Celles ci sont définies par le coordinateur, et ont pour principal avantage de permettre la synchronisation de l'ensemble des éléments entre eux, et ainsi d'économiser l'énergie de chacun des nœuds du réseau. Les supertrames sont tout d'abord divisées en deux parties : une partie active et une partie inactive ; durant la partie inactive le coordinateur n'interagit pas avec le réseau et se place en mode veille. Chaque supertrame commence par un beacon qui est envoyé par le coordinateur du réseau, et qui permet la synchronisation entre les nœuds et le coordinateur. Le beacon est immédiatement suivi par une zone appelée *Contention Access Period* (CAP), qui permet à chaque élément du réseau d'envoyer ou de recevoir des trames de commandes et de données.
L'accès au canal durant la période CAP suit le mécanisme CSMA-CA : l'émetteur écoute le canal pendant une durée aléatoirement tirée (algorithme de backoff) puis émet si le canal est libre. L'algorithme CSMA-CA induit une consommation électrique importante (surtout durant les périodes de fort trafic), ce qui a conduit à réduire l'intervalle valeur que peut prendre l'algorithme de backoff de 0 à 2. Ce procédé réduit la période de contention pendant laquelle l'émetteur doit écouter le canal.
Une autre zone optionnelle disponible dans la supertrame (CFP) est utilisée pour les transmissions nécessitant de la qualité de service. Le CFP est divisé en slots, dont le coordinateur gère l'allocation en fonction des besoins. Une station qui se voit attribuer un slot est assurée qu'aucune autre station ne transmettra durant cette période. L'allocation d'un slot fait suite à une négociation entre un nœud et le coordinateur. Si la négociation aboutit le coordinateur informe le nœud en plaçant une information (stream index) dans le prochain beacon (l'algorithme d'allocation n'est pas défini dans le standard).
Le mode de communication utilisant les supertrames garantit un très faible taux d'utilisation des ressources (souvent inférieur à 10%), car les nœuds formant le réseau ont un temps de veille garanti très long (les nœuds ne sont actifs que durant la période CAP).

5.1.0.1 Transfert de données

La norme IEEE 802.15.4 définit trois modèles transactionnels possibles :
- Transfert de données du coordinateur vers un nœud du réseau,
- Transfert de données d'un nœud du réseau vers le coordinateur,
- Transfert de données de deux nœuds du réseau entre eux.

Dans le cas d'une topologie en étoile, seules les deux premières transactions sont possibles car les échanges se font exclusivement avec le coordinateur

du réseau (comme dans un piconet BlueTooth). Alors que, dans le cas d'une topologie point à point, les trois types de transactions sont disponibles.

Transfert vers le coordinateur Un nœud, désirant communiquer avec le coordinateur dans un réseau utilisant la structure des supertrames, doit d'abord se synchroniser en utilisant le beacon comme référence, puis envoyer un paquet de données durant un slot de temps et attendre un acquittement de la part du coordinateur(optionnel). Dans un réseau n'utilisant pas la structure supertrame, le nœud émet une trame de données selon le mécanisme CSMA-CA, quand il le désire.

Transfert du coordinateur vers un nœud Quand le coordinateur souhaite émettre une trame vers un nœud dans un réseau utilisant le mécanisme des supertrames, il doit attendre que ce nœud soit synchronisé et qu'il demande (Data request) qu'on lui envoie les données en attente.
Le même mécanisme est mis en place dans un réseau n'utilisant pas les supertrames, le nœud émet alors régulièrement une trame(Data Request), pour demander au coordinateur, s'il n'a pas de données en attente à lui envoyer.

Transfert point à point Dans le cas du transfert de données point à point, les nœuds doivent être constamment synchronisés et écouter constamment le canal radio, alors le mode de transfert de données ne peut être que de type CSMA-CA sans slot de temps et sans structure de supertrames. On se retrouve alors dans les conditions d'un réseau ad hoc classique.

5.1.0.2 Mediation Device

La consommation électrique des composants dans un réseau est grandement réduite, si on utilise la technologie *slotted CSMA-CA*, avec une structure en supertrame. Mais cette technologie ne peut être déployée qu'avec une topologie en étoile. La topologie en étoile n'est utilisée que dans le cas où on a de petits réseaux. Les réseaux utilisant cette topologie sont donc restreints à des applications bien particulières, alors qu'une topologie point à point(peer to peer) permet le déploiement de réseaux de grande envergure. Dans les réseaux point à point, l'accès au canal de transmission ne peut se faire qu'avec une technologie CSMA-CA simple. Mais les nœuds étant actifs tout le temps (comme dans un réseau Wifi), cette solution est coûteuse en énergie. Pour y remédier, une technologie hybride appelée *Mediation Device* compatible avec la norme IEEE 802.15.4 peut être déployée dans les réseaux de capteurs utilisant une technologie CSMA-CA pour permettre de sauvegarder l'énergie des nœuds.

Un nœud du réseau, le coordinateur ou un autre nœud choisi aléatoirement, va jouer le rôle de médiateur entre tous les nœuds se trouvant dans son infosphère radio.
Quand un nœud X veut communiquer avec un nœud Y, il envoie une requête au médiateur. Tous les nœuds du réseau interrogent régulièrement le médiateur pour savoir si on cherche à les joindre. Donc quand le nœud Y interroge le médiateur, celui ci l'avertit que le nœud X veut entrer en contact avec lui. X et Y entament alors une communication point à point.

5.2 SMAC

5.2.1 généralités

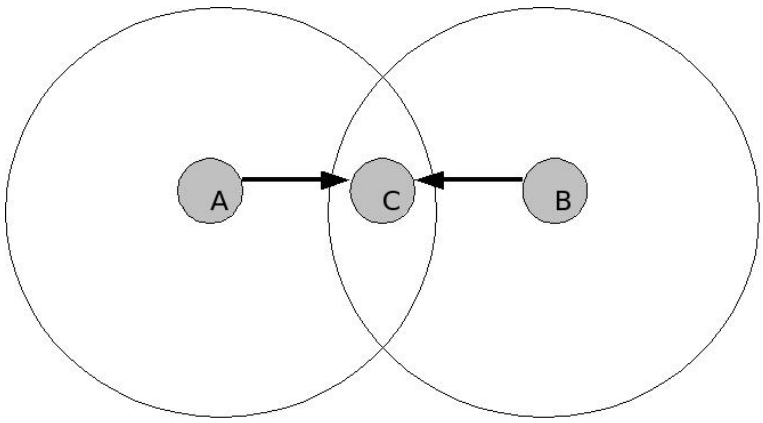

FIG. 5.1 – Problème de stations cachées

On peut avoir un phénomène de station cachée quand par exemple, 2 stations A et B qui ne se voient pas, voient par contre toutes les 2 une troisième station C comme sur la figure 5.1. Si A et B veulent toutes les deux envoyer des paquets à C, A ne voyant pas B et B ne voyant pas A, elles pensent toutes les deux que le medium pour envoyer des informations à C est libre et il va donc y avoir collision.

Dans un mécanisme de type RTS-CTS (*Request To Send*, *Clear To Send*), avant d'envoyer des données, l'émetteur envoie une trame RTS au récepteur qui lui répond par une trame CTS pour accepter ou refuser la conversation (si le CTS n'est pas reçu, on estime que le RTS a subi une collision). Les paquets RTS et CTS contiennent des informations sur la taille des données à transmettre, ainsi les voisins de l'émetteur des données (qui entendent donc le paquet RTS) et du recepteur (qui entendent donc le CTS) peuvent

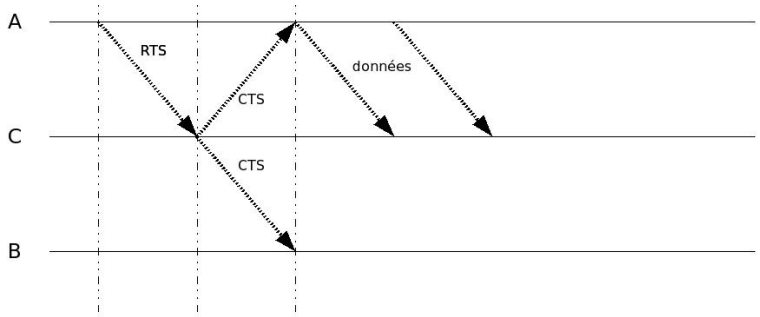

FIG. 5.2 – Mécanisme de RTS-CTS

calculer une estimation du temps durant lequel ils ne devront pas émettre sur le canal radio. On résoud ainsi le problème des stations cachées et on assure ainsi une réservation du canal. Ainsi, en reprenant l'exemple précédent, le nœud A, comme cela est illustré sur la figure 5.2 peut réserver le canal et ainsi empêcher B d'envoyer des paquets au nœud C qui provoqueraient une collision.

5.2.2 SMAC

5.2.2.1 Bases

SMAC (*Sensor MAC*) [13] est un protocole de couche de liaison relativement similaire au protocole 802.11 et qui utilise une méthode d'accès au canal de type CSMA-CA RTS/CTS (*Request-To-Send*, *Clear-To-Send*) qui permet d'éviter les collisions et le problème de station cachée. La principale innovation, apportée par ce protocole, est d'avoir développé un mécanisme de mise en veille distribué à chaque nœud du réseau dans le but de réduire la consommation énergétique des équipements réseau. Avec le protocole S-MAC chaque nœud peut périodiquement se mettre en veille (cf figure 5.3), la principale difficulté est alors de synchroniser les nœuds entre eux pour que la communication soit toujours possible. S-MAC définit donc une frame composée d'une période de sommeil et d'une période d'éveil. Si un nœud est en train de transmettre ou recevoir des données, il ne suit pas son ordonnanceur de sommeil avant la fin de la transmission.

5.2.2.2 synchronisation

Pour synchroniser les nœuds entre eux, le protocole S-MAC permet à chacun des noeuds d'émettre des paquets SYNC qui permettent aux autres nœuds de se synchroniser. Le paquet SYNC est broadcasté à tous les voisins immédiats. La période active est divisée en deux phases, la première étant

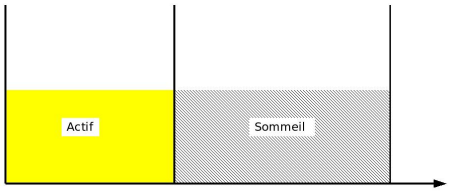

FIG. 5.3 – Protocole S-MAC

reservé à la synchronisation.

Quand un nœud se réveille, il commence par écouter. Si il n'entend rien pendant un certain temps, il choisit un ordonnancement de frame et le transmet par un paquet SYNC qui contient le temps jusqu'à son endormissement. Si au démarage, le nœud reçoit un paquet SYNC d'un autre nœud, il suit l'ordonnancement défini dans ce paquet et transmet son propre SYNC en conséquence. Les nœuds retransmettent leur SYNC de temps en temps. Si un nœud a déjà adopté un ordonnancement et qu'il reçoit un paquet SYNC d'un autre nœud avec un autre ordonnancement, alors il doit adopter les 2 ordonnancements. Il doit aussi transmettre un paquet SYNC à l'autre nœud pour que celui-ci apprenne la présence d'un autre ordonnancement. Le fait d'adopter les deux ordonnancement signifie que le nœud va se réveiller au début des deux frames. Il faut donc éviter qu'un nœud adopte plusieurs ordonnancements.

5.2.2.3 Overhearing Avoidance

Chaque nœud maintient un vecteur d'allocation du réseau (*network allocation vector NAV*) dans lequel il indique l'activité de ses voisins. Chaque paquet de données contient un champ indiquant le temps restant. Lorsqu'un nœud reçoit un paquet de données qui ne lui est pas destiné, il met à jour son NAV. A chaque intervalle de temps, la valeur du NAV est décrémentée (si elle est non nulle). Ainsi une valeur non nulle du NAV signifie qu'il y a une transmission. Le nœud peut donc s'endormir et ne se réveiller ensuite que lorsque le NAV devient nul.

5.3 TMAC

5.3.1 Bases

Dans T-MAC (*Timeout MAC*) [11], chaque nœud se réveille périodiquement pour communiquer avec ses voisins. On a donc, comme dans S-MAC

des périodes actives et inactives. Les noœud communiquent entre eux en utilisant un mécanisme de RTS/CTS (*Request-To-Send*, *Clear-To-Send*) qui permet d'éviter les collisions et le problè de station cachée. Un nœud écoute le canal radio et peut potentiellement transmettre tant qu'il est dans la période active. Une période active se termine quand aucun évènement d'activation ne se produit pendant un temps *TA*. Un évenment d'activation est :

- le déclenchement d'un temporisateur de frame
- la reception de données sur la radio
- la détection (grâce à un indicateur de puissance du signal sur la radio) d'une communication sur le canal radio
- l'acquittement de ses propres paquets
- la connaissance grâce à l'écoute qui précède l'émission de RTS et CTS que la transmission de données d'un voisin s'est terminée.

TA est donc le temps minimum d'écoute à vide par frame.

5.3.2 Synchronisation

Quand un nœud se réveille, il commence par écouter. Si il n'entend rien pendant un certain temps, il choissit un ordonnancement de frame et le transmet par un paquet SYNC qui contient le temps jusqu'au début de la frame suivante. Si au démarage, le nœud reçoit un paquet SYNC d'un autre nœud, il suit l'ordonnancement défini dans ce paquet et transmet son propre SYNC en conséquence. Les nœuds retransmettent leur SYNC de temps en temps. Si un nœud a déjà adopté un ordonnancement et qu'il reçoit un paquet SYNC d'un autre nœud avec un autre ordonnancement, alors il doit adopter les deux ordonnancements. Il doit aussi transmettre un paquet SYNC à l'autre nœud pour que celui-ci apprenne la présence d'un autre ordonnancement. Le fait d'adopter les deux ordonnancement signifie que le nœud va avoir un évènement d'activation au début de chacune des deux frames. Les nœuds ne doivent débuter une transmission de données qu'au début de leur propre période active. A ce moment, à la fois les voisins ayant le même ordonnancement et ceux l'ayant adopté en plus sont réveillés.
TA doit être suffissament long pour prendre en compte l'attente avant l'émission du paquet SYNC et le temps de la réception éventuelle d'un paquet RTS.

5.3.3 Overhearing Avoidance

Le mécanisme utilisé dans S-MAC peut-être utilisé dans T-MAC en option. C'est juste une option. Cela peut entrainer dans certaines conditions qu'un nœud manque des RTS et des CTS.

5.3.4 FRTS

Avec T-MAC, il peut y avoir un problème quand le trafic est essentiellement unidirectionnel (ce qui est le cas quand les capteurs doivent surtout remonter les informations vers une station collectante). Par exemple (voir figure 5.4) supposons qu'un nœud A doit envoyer des données à un nœud D en passant par les nœuds B et C, A ayant pour voisin B, B ayant pour voisins A et C, C ayant pour voisins B et D et D ayant pour voisin C. Considérons le nœud C avec des paquets en attente pour le nœud. Le nœud C peut perdre la contention à cause de B (en recevant un paquet RTS de B) ou à cause de A (en recevant un paquet CTS de B en réponse à un paquet RTS que B a reçu de A). Considérons le deuxième cas. C ayant reçu le CTS de B, il doit rester silencieux et donc D ne reçoit rien. Par conséquent il retourne en sommeil. Quand A a fini de transmettre les données, B émet un paquet d'aquitement (ACK) que tous ses voisins reçoivent (et donc C). C peut donc espérer obtenir la contention pour transmettre ses paquets à D mais D étant en sommeil, il ne peut pas recevoir le paquet RTS de C.

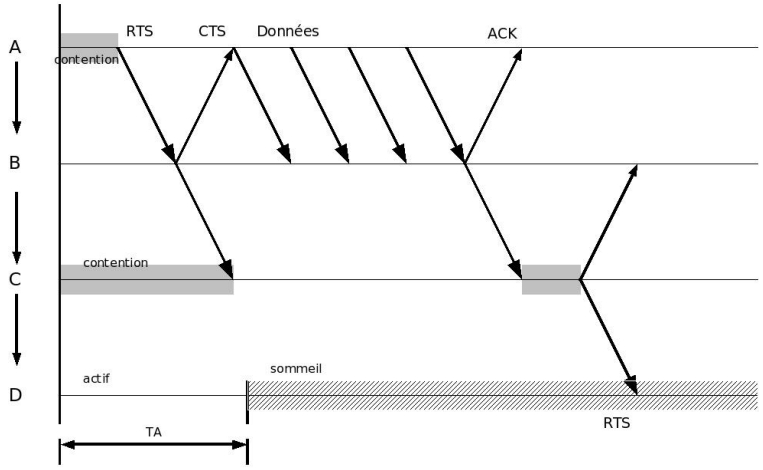

FIG. 5.4 – Problème de l'endormissement précoce

Une des solutions est l'introduction du *future request-to-send* (FRTS). Dans l'exemple précédent (voir figure 5.5). quand C reçoit le CTS de B, C envoie un paquet FRTS à D (B entend ce paquet) qui contient la longueur de la communication de données bloquante (cette information est contenue dans le CTS que C a reçu). Ainsi le nœud peut déterminer qu'il sera ensuite le destinataire d'un paquet RTS et ne pas être endormi quand ce paquet arrivera. (Il faut tenir compte du temps de récepetion d'un éventuel paquet FRTS pour le calcul de *TA*). L'ajout du mécanisme de FRTS entraine

qu'il faut rajouter un paquet *Data-Send* (DS) avant l'émission de paquets de données. En effet puisque C émet le paquet FRTS après avoir reçu le paquet CTS de B, A envoie un paquet DS (de même taille qu'un paquet FRTS) aprés avoir reçu le paquet CTS. Ainsi il ne risque pas de perdre pas la contention et les données qu'il doit envoyer à B n'entrent pas en collision avec le paquet FRTS que B reçoit.
Le mécanisme de FRTS ne doit être utilisé que sous certaines conditions de trafic car sinon cela rajoute une consommation d'énergie qui peut être évitée.

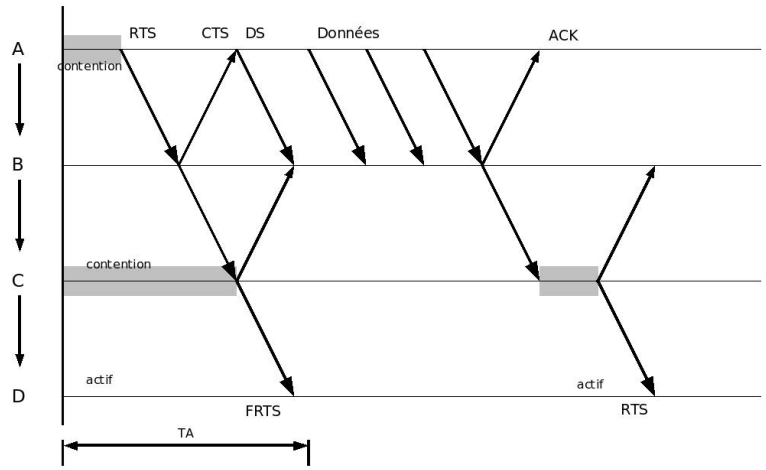

FIG. 5.5 – Mécanisme de FRTS

5.4 BMAC

BMAC (*Berkeley MAC*) [10] a été développé par l'Université de Berkeley et est actuellement utilisé dans TinyOS avec la couche physique de la norme IEEE 802.15.4 pour les capteurs compatibles ZigBee.
BMAC est basé principalement sur deux principes : l'analyse du bruit sur le canal radio et sur l'écoute basse consommation.
Quand un nœud veut envoyer un paquet, il détermine si le canal radio est utilisé par un autre nœud ou pas en écoutant le "bruit" en se basant sur un indicateur de puissance du signal. Si il n'y a pas de bruit, le canal est libre et il peut donc émettre. Avant d'envoyer des données il doit émettre un préambule.
Les nœuds sont en sommeil la plupart du temps et se réveillent à intervalles réguliers (comme illustré sur la figure 5.6). A leur réveil ils écoutent le bruit sur le canal radio. Si il n'y a pas de bruit sur le canal radio, le nœud retourne

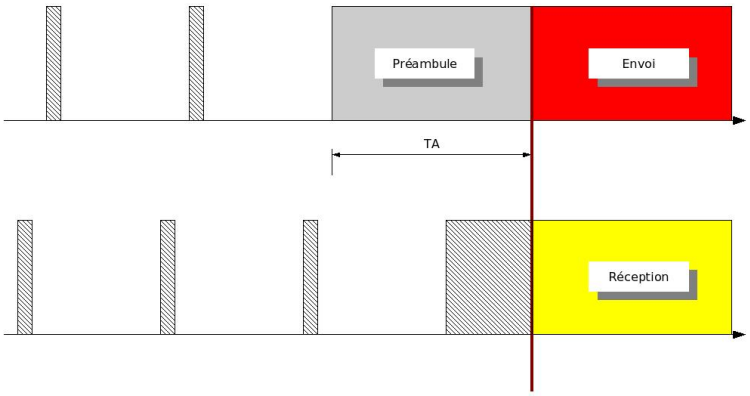

FIG. 5.6 – BMAC : réveil périodique des nœuds

en sommeil. Si il y a du bruit, cela signifie que des données vont arriver (à cause du préambule). Le préambule doit être suffissament long, pour que tous les nœuds (et donc le destinataires des données) puissent l'entendre. Ensuite quand les données commencent à arriver, les nœuds qui ne sont pas destinataires de ces données retournent en sommeil.
Un des avantages de BMAC est qu'il ne nécessite pas de synchronisation entre les nœuds.

Chapitre 6

Le simulateur

Nous allons comparer TMAC et BMAC par simulation.

6.1 Présentation du simulateur OMNeT++

OMNeT++[1] est un simulateur réseau à évènements discrets. Il utilise une archtitecture orientée objet. Un modèle OMNeT++ est une hiérarchie de modules. Un modèle de réseau est décrit en utilisant un langage d'interface appelé NED (NEtwork Descriptor).

Un module est une entité dans laquelle on peut définir des fonctionnalités. Un module peut être un nœud, un protocole, une couche, la zone de portée d'une onde radio (dans le cas de réseaux sans fil), etc. Les modules peuvent avoir leurs propres paramètres. Ces paramètres peuvent être utilisés pour modifier les comportements du module et paramétrer la topologie du modèle. Les modules au plus bas niveau de la hierarchie de module décrivent le comportement. Ces modules sont appelés modules simples et ils sont implémentés en C++ en utilisant la librairie de simulation.

Les modules communiquent entre eux en se passant des messages. Les messages peuvent contenir des structures de données arbitrairement complexes. Les modules simples peuvent envoyer des messages soit directement à leur destination soit en utilisant un chemin prédéfini en utilisant des portes et des connexions. Les portes sont les interfaces d'entrée et de sortie des modules. Une connection (ou lien) est créée entre une porte de sortie et une porte d'entrée. Du fait de la structure hierarchique du modèle, des messages peuvent traverser une série de connections pour partir et arriver dans des modules simples. Une telle série de connections d'un module simple vers un module simple est appelée une route.

6.2 Le *framework* MACSimulator

MACSimulator[2] est un simulateur basé sur OMNeT++ et écrit à l'origine par les auteurs de [11]. Il permet de simuler différentes couches MAC en utilisant plusieurs modèles de trafic. Les résultats de simulation obtenus avec ce simulateur pour chaque nœud :

- pour la couche application : le nombre de paquets envoyés et reçus
- pour la couche réseau : le nombre de paquets envoyés, reçus et encore en queue ainsi que le nombre de paquets dropés
- pour la couche mac : le nombre de paquets envoyés et reçus ainsi que le temps passé à envoyer et recevoir des paquets, le temps d'*overhearing* et le temps d'*overhead*
- pour la couche radio : le temps passé avec la radio coupée, le temps passé en reception et en envoi en mode normal et en mode basse consommation.

Ce mode basse consommation n'a rien à voir avec le concept d'écoute basse consommation introduit dans BMAC. Il s'agit ici juste d'un état actif de la radio avec une consommation plus faible que peuvent adopter certains modèles de radio quand ils ne sont pas en train de recevoir ou de transmettre des paquets.

MACSimulator est donc bien adapté à l'étude de la consommation d'énergie des capteurs. De plus un certain nombre de couches MAC sont déjà implémentées dans le simulateur dont SMAC, TMAC et CSMA. De plus un mécanisme d'écoute basse consommation est prévu ce qui facilite la mise en place de BMAC. Dans les résultats de simulations, on utilise la moyenne de la consommation d'énergie sur l'ensemble du réseau.

Chapitre 7

Etude comparative de BMAC, SMAC et TMAC

7.1 Le modèle de simulation

Le réseau considéré est composé de cent nœuds répartis sur une grille de dix par dix. Pour calculer les consommations d'énergie, on a utilisé les caractéristiques de la puce radio RFM1000. Il n'y a pas de mode basse consommation et donc l'émission et la reception en mode basse consommation dans la simulation seront considérées comme si elles étaient en mode normal pour effectuer les calculs de consommation d'énergie. La consomation de la puce est la suivante :

- 0,02 mA en sommeil
- 4,0 mA en réception
- 10,0 mA en émission

7.2 Modèle *Nodes-to-Sink*

Dans le modèle de trafic Node-To-Sink, les nœuds envoient des messages à un nœud unique que nous avons choisi dans un coin de notre grille, en l'occurence, le nœud 0. Les messages sont routés en utilisant un algorithme non déterministe du plus court chemin. Aucune aggrégation de données n'est réalisée. Ce modèle est particulièrement intéressant car cela peut représenter le trafic vers une station collectante ou le trafic vers un *clusterhead* si on veut utiser de la clusterisation.

Pour TMac, on utilise une période de cycle de six cent dix millisecondes et une longueur de quinze millisecondes pour le *timeout*. On utilise TMAC avec l'*overhearing avoidance* et le mécanisme de FRTS. Pour SMac, on utilise une période de cycle de une seconde et on fait varier la longueur de la période active.

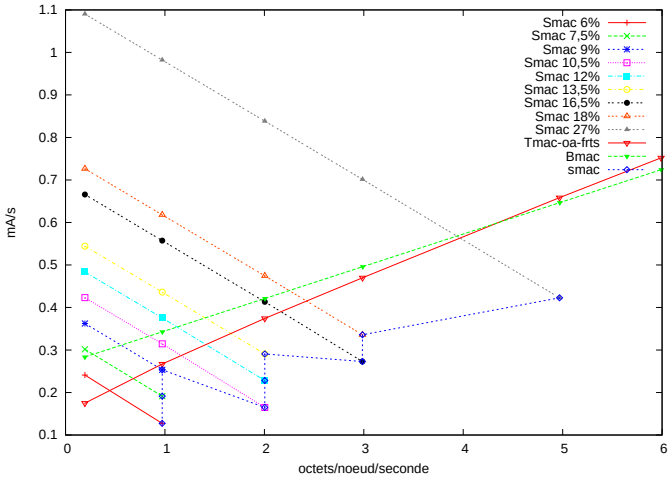

FIG. 7.1 – réseau à un saut avec une taille de paquet de 20 octets

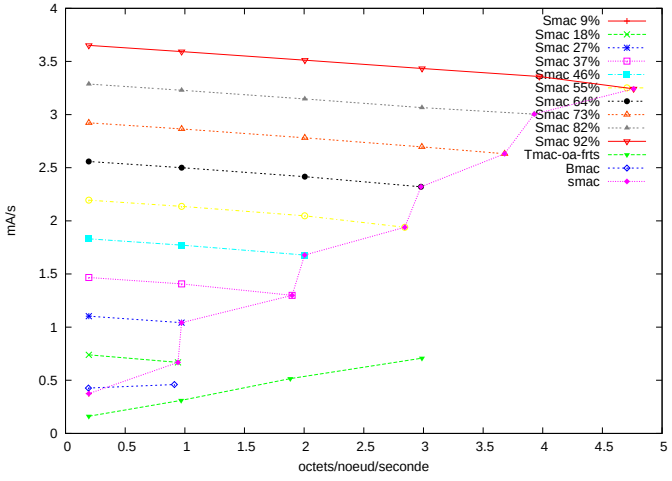

FIG. 7.2 – réseau multi-saut avec une taille de paquet de 20 octets

Sur la figure 7.1, tous les nœuds sont dans le même voisinage radio. Dans le cas d'un réseau multisaut (figure 7.2), chaque nœud n'a à sa portée radio que les huits nœuds qui l'entourent (sauf dans le cas des nœuds du bord de la grille).
Sur les figures 7.1 et 7.2, on trace la courbe de SMAC pour différentes valeurs de la période active (exprimée ici en pourcentage de la période de cycle). Dans les deux cas, pour SMAC, on contaste que, à longueur de période active fixée, la consommation d'énergie moyenne sur le réseau décroit quand la charge de trafic croit, pour atteindre pour un certain trafic une consommation minimum. Au delà de cette charge de trafic, il y a saturation avec des pertes de paquets dues à des débordements de buffer. La baisse de la consommation d'énergie quand le trafic croît est due au mécanisme d'*overhearing avoidance*. En effet, plus un nœud émet, plus il "oblige" ses voisins à dormir pendant leur période active, ce qui entraine qu'ils consomment moins d'énergie. La saturation est du au fait que si un nœud contraint ses voisins trop souvent à dormir pendant leur période active, ceux-ci n'auront plus le temps d'envoyer leur paquets, d'où un débordement de buffer.
Dans un réseau à un saut (figure 7.1), SMAC présente de meilleures performances que TMAC et BMAC. On peut également constater que TMAC est plus performant que BMAC sur les faibles débits tandis que pour des débits un peu plus élever c'est l'inverse car la consommation d'énergie pour BMAC croît moins vite avec le débit que pour TMAC.
Dans un réseau multisaut (figure 7.2), BMAC et TMAC sont meilleurs que SMAC. La consommation d'énergie par rapport au débit croît beaucoup plus vite pour SMAC que pour TMAC et BMAC, en considérant àchaque fois pour SMAC la consommation minimale pour le débit considéré. Cependant BMAC sature très tôt, et on ne peut pas aller au delà de 1 octet par seconde ; ceci est notamment du à un problème de collisions car BMAC n'a aucun mécanisme pour éviter le problème de stations cachées. Cependant la consommation d'énergie pour BMAC semble croître moins avec le débit que pour TMAC. On peut donc imaginer, au vue de l'allure des courbes, que si on n'avait ce problème de saturation dans BMAC, que cette couche MAC serait plus performante que TMAC à partir de débits relativement peu élevés. SMAC peut tenir des débits plus importants que TMAC.

7.3 Modèle Unicast local

Dans le modèle unicast utilisé ici, l'émetteur choisi aléatoirement parmi ses voisins un nœud qui sera le destinaire du message, chaque nœud ayant pour voisin les 8 nœuds qui l'entourent (sauf dans le cas d'un nœud au bord de la grille).

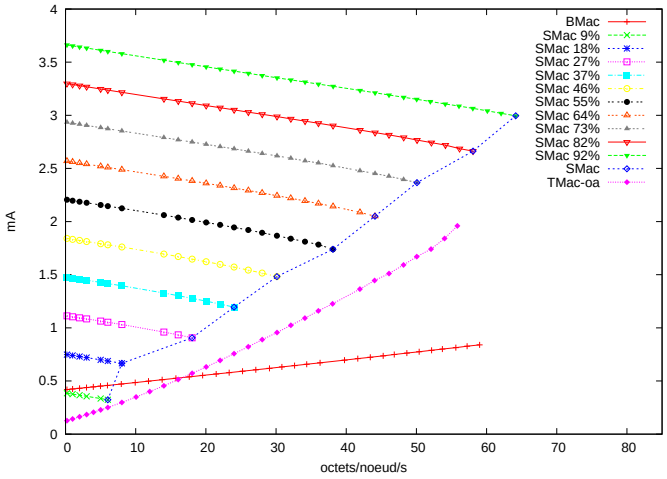

FIG. 7.3 – Local Unicast avec une taille de paquet de 20 octets

Pour TMac, on utilise une période de cycle de six cent dix millisecondes et une longueur de quinze millisecondes pour le *timeout*. On utilise TMAC avec l'*overhearing avoidance*, ce qui donne les meilleures performances. On n'a pas garder les autres configurations d'option de TMAC pour assurer une meilleure lisibilité de la courbe. Pour SMac, on utilise une période de cycle de une seconde et on fait varier la longueur de la période active. Pour tracer la consommation de SMAC sur la figure 7.3, nous avons utiliser la même manière que pour les figures 7.1 et 7.2. La consommation d'énergie pour SMAC est supérieure à celle de TMAC et BMAC. TMAC est plus preformant que BMAC pour les faibles débit, mais l'évolution de la consommation d'énergie par rapport au débit étant plus faible pour BMAC que pour TMAC, BMAC est meilleur sur le débit plus élevé (environ pour les débits supérieurs à 20 octets/secondes). Contrairement à ce qu'on observe pour le modèle Node-to-Sink, TMAC et BMAC sature pour des débits semblables. SMAC peut tenir des trafics légèrement plus élevés que ces derniers.

7.4 Conclusion

Les simulations effectuées nous ont permis de montrer que TMAC et BMAC étaient meilleurs que SMAC en terme de consommation énergétique. Dans les simulations effectuées nous avons aussi montré que BMAC consommait moins d'énergie que TMAC et SMAC dans le cas de débit élevé, alors que dans le cas de faibles débits, TMAC est meilleur que BMAC.

Cependant il est à noter que BMAC sature à des débits relativement faibles, cela est dû au phénomène de stations cachées.

Chapitre 8

Développements futurs : évolution de BMAC

8.1 Limitations de BMAC

L'envoi de paquets dans BMAC est précédé d'un long préambule pour que les nœuds puissent avoir connaissance que des paquets vont être envoyés. Dans le cas où un nœud doit émettre un flux continu important de paquets de données, émettre des préambules "longs" pour chaque paquet représente une consommation d'énergie pas forcement négigeable et qui peut être évitée (on appelle par la suite préambule long un préambule qui est suffisemment long pour couvrir au moins un cycle endormissement-éveil dans un protocole utilisant un principe d'écoute basse consommation comme dans BMAC). De plus, pendant l'émission de ce flux, les nœuds voisins de l'émetteur autre que le destinataire des paquets vont continuer à se réveiller périodiquement inutilement. Cela augmente la consommation d'énergie, surtout que ces nœuds, chaque fois qu'ils se réveillent, sont obligés d'attendre de recevoir l'entête d'un paquet pour se rendre compte que le bruit sur le canal radio ne correspond pas à des paquets qui leur sont destinés.
Par ailleurs BMAC, avec son système d'évaluation du bruit sur le canal pour déterminer s'il est libre ou non, ne permet pas de se protéger du problème de stations cachées.

Cependant BMAC, par sa simplicité, est très intéressant. Par son absence de besoin de synchronisation, il permet que l'ajout de nouveaux nœuds soit aisé et sans trop de risque de dégradation de performances. Dans SMAC et TMAC, lorsque l'on ajoute un nouveau nœud, il y a toujours le risque que ce nouveau nœud adopte un autre cycle, obligeant ainsi ces voisins à se synchroniser sur ce cycle en plus de celui qu'ils suivaient déjà, dégradant

ainsi leurs performances en terme de consommation d'énergie.

Les auteurs de BMAC [10] proposent d'implémenter un mécanisme de RTS-CTS dans une couche au dessus de BMAC. Cependant cette solution, bien que resolvant le problème de terminal caché, laissent certaines des autres limitations décrites ci-dessus sans réponse.

8.2 la nouvelle couche MAC

Notre solution consiste à implémenter un mécanisme de RTS-CTS directement dans BMAC et à partir de celà en tirer des optimisations. Cette couche MAC, bien que reposant sur des principes tirés de BMAC et de SMAC, constitue une nouvelle couche MAC car elle en diffère de manière significative. En voici le principe de fonctionnement.

8.2.1 Fonctionnement

Comme dans BMAC, les nœuds se réveillent à intervalles réguliers et chaque phase de réveil commence par une phase d'écoute du canal. Si il n'y a pas de bruit sur le canal, le canal est libre. Dans le cas contraire, le canal est occupé et le nœud reste éveillé jusqu'au début de la réception du paquet. Lorsqu'un nœud veut émettre un ou des paquets de données, au début de sa phase d'éveil il écoute le canal pendant une duré alétoire dans un intervalle préderminé. Si il n'y a pas de bruit sur le canal, il émet un paquet RTS précédé (comme les paquets dans BMAC) d'un préambule suffisamment long pour que tous ses voisins puissent savoir qu'un paquet va arriver. Si il peut accepter ce transfert de données, le destinataire du RTS répond par un paquet CTS avec lui aussi un préambule long et reste éveillé. L'émission du CTS est précédée d'une periode de contention, comme pour le RTS. Le CTS contenant des informations sur la taille des données qui vont être transmises, les voisins du destinataire des données vont pouvoir estimer la durée de cet échange et se mettre totalement en sommeil durant cette période, c'est-à-dire que durant cette période, ils abandonnent les réveils réguliers. Aprés la réception du CTS, la source des données émet un paquet DS (Data Send) précédé d'une période de contention et d'un préambule long, ce paquet contenant des informations sur la taille des données qui vont être émises avant que ses voisins puissent passer en sommeil total durant la période d'émission des données. Les paquets de données sont ensuite émis sans préambule long.

8.2.2 Comparaison par rapport à BMAC et TMAC

L'avantage de cette solution par rapport à BMAC est valable si les nœuds ont un nombre suffisamment grand de données à transmettre à la fois. En effet, si les nœuds ont relativement peu de données à transmettre (comme par exemple un ou deux paquets à la fois), BMAC est bien meilleur et notre solution est plus lourde et entraine une plus grande consommation d'énegie sauf peut-être si la répartition du trafic et la topologie du réseau font qu'il y a un grand risque de collision à cause du problème de terminaux cachés. L'avantage de cette solution, par rapport à l'implémentation d'un mécanisme de RTS-CTS dans une couche au dessus de BMAC, est que l'on peut tirer tous les avantages de ce mécanisme puisqu'il est directement implémenté dans la couche MAC. Ainsi on n'émet pas des préambules longs pour les paquets de données et les nœuds ne pouvant pas participer au trafic (un voisin de l'émetteur ou du récepteur ne doit pas émettre pendant la transmission sous peine de provoquer une collision) sont mis en sommeil total.

L'avantage par rapport à SMAC et TMAC est que cette solution ne nécessite pas de synchronisation.

Conclusion

Pour essayer d'avoir un réseau évolutif (dans le sens où l'on peut rajouter d'autres capteurs ne provenant pas forcément du même constructeur), l'utilisation de BlueTooth peut se justifier. Mais les spécifications de BlueTooth rendent difficile, voire impossible, la création d'un grand réseau de capteurs.

Les protocoles SMAC, TMAC et BMAC sont adaptées à la principale contrainte des réseaux de capteurs, la gestion de l'énergie, Cependant TMAC et BMAC sont plus souples que SMAC qui a un fonctionnement optimale pour un débit donné et qui est, dans la plupart des cas, moins performant que les deux autres.

Dans les simulations effectuées nous avons aussi montré que BMAC consommait moins d'énergie que TMAC et SMAC dans le cas de débit élevé, alors que dans le cas de faibles débits, TMAC est meilleur que BMAC. Cependant il est à noter que BMAC sature à des débits relativement faibles, cela est dû au phénomène de stations cachées. C'est pourquoi il est intéressant d'intégrer un mécanisme de RTS/CTS dans BMAC par pallier à ce problème. L'introduction de ce mécanisme s'accompagne de quelques optimisations permettant d'économiser un peu d'énergie lors de l'émission de paquets de données.

Bibliographie

[1] Omnet++ community site. http://www.omnetpp.org.

[2] the consensus project site. http://www.consensus.tudelft.nl.

[3] the mantis project home page. http://mantis.cs.colorado.edu/.

[4] the sos operating system. http://nesl.ee.ucla.edu/projects/sos/.

[5] Alexandre Delye de Mazieux, Vincent Gauthier, Michel Marot, and Monique Becker. État de l'art sur les réseaux de capteurs. *INT 05001RST*, avril 2005.

[6] Alexandre Delye de Mazieux, Vincent Gauthier, Michel Marot, Julien Vaudour, Monique Becker, Rahim Kacimi, Riadh Dhaou, and André-Luc Beylot. État de l'art, réseaux de capteurs sans fil. *Délivrable, projet Capteurs*, mai 2006.

[7] Bluetooth Special Interest Group. *Specification of the Bluetooth system*. http ://www.bluetooth.com, novembre 2004.

[8] IEEE. 802.15.4-2003 standard for information technology - telecommunication and information exchange between systems - lan/wan - part 15.4 : Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (lr-wpam)., 2003.

[9] Yong Liu, Myung J. Lee, and Tarek N. Saadawi. On-demand formation of bluetooth scatternet. In *Proceedings of MILCOM 2002*, volume 2, pages 1069– 1074, Octobre 2002.

[10] Joseph Polastre, Jason Hill, and David E. Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004*, pages 95–107, Baltimore, MD, USA, November 2004. ACM.

[11] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems,*

SenSys 2003, pages 171–180, Los Angeles, California, USA, November 2003. ACM.

[12] Zhifang Wang Wang, Robert J. Thomas, and Zygmut Haas. Bluenet – a new scatternet formation scheme. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, volume 2, page 61, Big Island, Hawaii, Janvier 2002. HICSS'02.

[13] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, USA, June 2002. IEEE.

[14] Gergely V. Zaruba, Stefano Basagni, and Imrich Chlamtac. Bluetrees-scatternet formation to enable bluetooth-based ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, volume 1, pages 273–277, St. Petersbourg, Russie, Juin 2001. IEEE.