
# Implémentation d'une couche MAC (accès au support) dans OMNET++/Castalia pour des réseaux de capteurs pour la surveillance.

DOS SANTOS Leonel et LALANNE Clément

6 mai 2011

Master Informatique Technologies de l'Internet
Université de Pau et des Pays de l'Adour

Professeurs Encadrants : CARIOU Eric, PHAM Congduc

# Table des matières

# Table des figures

# 1 Introduction

Ce rapport présente le travail que nous avons effectué sur le projet tutoré proposé au second semestre de la première année du Master Technologies de l'Internet. Nous avons étudié les réseaux capteurs sans-fils et la manière de les faire communiquer entre eux, plus particulièrement le fait de faire transiter les informations qu'ils récoltent vers une destination précise.

Une problématique se pose dans ce contexte précis puisque les capteurs ont une énergie limitée et ne sont pas tous directement à portée d'émission d'un destinataire susceptible de recueillir les données mesurées. Comment procéder pour que chaque capteur puisse envoyer ses données recueillies sans interférer avec les autres ? Comment effectuer cela le plus rapidement possible ? Pour répondre à cette problématique nous avons produit une nouvelle couche MAC permettant une propagation des messages de capteur en capteur.

Dans un premier temps nous allons présenter les réseaux de capteurs, ensuite, dans un deuxième temps nous allons présenter l'application Castalia qui nous a permis de développer notre couche MAC. Dans un troisième temps nous allons aborder en détails le travail que nous avons effectué en expliquant notamment l'algorithme de fonctionnement de la couche MAC, et pour finir dans un quatrième temps nous présenterons le cahier des charges ainsi que les diagrammes de Gantt relatifs à la gestion de notre projet. Pour finir nous conclurons sur le projet ainsi que les difficultés qu'une telle discipline implique.

# 2 Réseaux de capteurs

## 2.1 Réseaux de capteurs

Un capteur est un petit appareil autonome capable d'effectuer des mesures simples sur son environnement immédiat. L'utilisation de ces capteurs n'a rien d'une nouveauté, ceux-ci sont utilisés depuis longtemps dans des domaines comme l'aéronautique ou l'automobile. Ce qui est novateur, c'est la possibilité pour ces capteurs de communiquer de manière radio (réseaux sans-fils de type WiFi) avec d'autres capteurs proches (quelques mètres) et pour certains d'embarquer de la capacité de traitement (processeur) et de la mémoire.. On peut ainsi constituer un réseau de capteurs sans-fils (RCSF) qui collaborent sur une étendue assez vaste. Comme les ressources d'un capteur sont très limitées, on peut même envisager que la réalisation d'un service complexe puisse être effectuée grâce à une composition de services plus simples et donc à une forme de collaboration "intelligente" des capteurs. Ces réseaux de capteurs soulèvent un intérêt grandissant de la part des industriels ou d'organisations civiles où la surveillance et la reconnaissance de phénomène physique est une priorité. Par exemple, ces capteurs mis en réseau peuvent surveiller une zone délimitée pour détecter soit l'apparition d'un phénomène donné (apparition de vibrations, déplacement linéaire...) soit mesurer un état physique comme la température (détection des incendies en forêts) ou la pression. Dans beaucoup de scénarios de gestion de crise (séismes, inondations,...) ces réseaux de capteurs peuvent permettre une meilleure connaissance du terrain afin d'optimiser l'organisation des secours, ou bien même renseigner précisément les scientifiques sur les causes d'un phénomène physique particulier.

## 2.2 Couches MAC existantes

Actuellements il existe plusieurs couches MAC pour la communication des capteurs sans-fil : on trouve la couche S-MAC, la couche Timeout MAC (TMAC, qui est un amélioration de S-MAC), B-MAC qui est une couche MAC basée sur la norme IEEE 802.15.4 (Wi-Fi).

# 3 Présentation de l'application Castalia

Castalia[1] est un simulateur pour des réseaux de capteurs sans fils Wireless Sensor Networks (WSN), Body Area Networks (BAN) et généralement pour des réseaux de capteurs embarqués avec peu d'énergie ou une énergie limitée. Castalia est basé sur la plateforme OMNeT++ et peut être utilisé par des chercheurs et des développeurs qui veulent tester leurs algorithmes distribués et ou leurs protocoles dans de vrais modules de communication sans-fil, avec un comportement réaliste plus particulièrement en ce qui concerne l'accès à la couche Radio. Castalia peut aussi être utilisé pour évaluer les caractéristiques des supports utilisés pour des applications spécifiques, grâce au fait que celui-ci est entièrement paramétrable et peut simuler une large gamme de supports sans-fil.

OMNeT++[2] est une bibliothèque de simulation écrite en C++ pour construire des simulateurs de réseaux au sens large ; c'est à dire réseaux filaires et sans-fils, mais également des réseaux internes aux machines (BUS de processeur par exemple).

Castalia offre la possibilité de manipuler différentes couches du modèle OSI [1] du découpage des réseaux. En effet il est possible de définir des couches MAC (accès au support de communication), des couches Réseau (routage des paquets) et des couches Application permettant de créer des réseaux de nœuds statiques. Castalia permet aussi définir des modules pour les réseaux sans-fils avec des nœuds mobiles (pour simuler le déplacement d'un téléphone portable par exemple).

La figure 1 explique brièvement le fonctionnement d'une application Castalia. On trouve les modules de communication à savoir "Routing" et "MAC" présentés ci-dessus et le module "Radio" qui simule le moyen de transport physique de l'information ("l'air" dans notre cas puisqu'il s'agit de réseaux sans-fil). Ensuite vient la couche "Application" et le gestionnaire de capteurs qui vont définir et gérer où sont placés les capteurs, qu'elles sont leurs caractéristiques techniques comme leur énergie, leur portée d'émission, etc...

1. de l'anglais Open Systems Interconnection, "Interconnexion de systèmes ouverts" OSI est un modèle de communications entre ordinateurs proposé par l'ISO (Organisation internationale de normalisation)

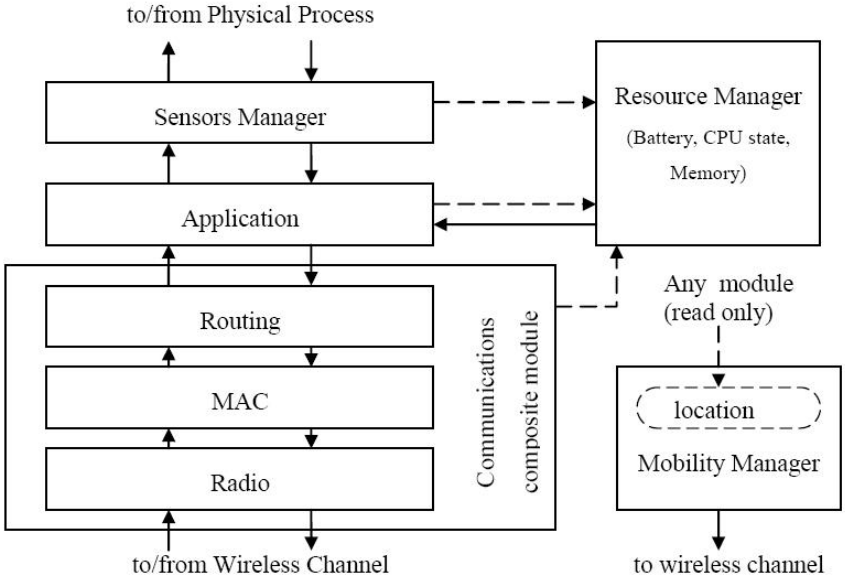


FIGURE 1 – Schéma général d'un module Castalia

Pour créer un nouveau module on utilise le langage NED provenant de OMNeT++ qui permet de définir facilement un nom de module, ses paramètres, ses interfaces (entrées et sorties) et éventuellement un sous-module. Ces informations sont écrites dans un fichier *.ned*. Pour chaque type de module, Castalia fournit une classe dont il faut hériter ainsi que des méthodes virtuelles à redéfinir pour écrire le code C++ du module lui-même.

C'est dans ce fichier que va se trouver l'implémentation du protocole de communication, la couche MAC développée lors de notre projet. La suite du rapport va également décrire plus précisément un fichier de configuration *.ned*.

# 4 Présentation de notre travail

Notre travail, mis à part le fait de devoir apprendre à utiliser l'environnement de simulation, a été de concevoir une couche mac qui répond à la problématique posée. Pour cela nous avons du passer par une phase de mise place du module MAC dans l'environnement de simulation Castalia.

## 4.1 Mise en place de l'environnement de travail

### 4.1.1 La couche mac

Pour pouvoir développer une couche mac qui répond à nos objectifs, deux solutions s'offraient à nous. Modifier une des couches MAC déjà existantes ou en développer entièrement une nouvelle. Dans un premier temps nous avons utilisé la première solution. Nous avons cherché à comprendre le fonctionnement de la couche TMAC et ensuite la modifier pour la faire correspondre à nos objectifs. Cependant cette démarche a été assez compliquée car dans un premier lieu nous ne maîtrisions pas le sujet et dans un second lieu les codes sources disponibles pour utiliser cette couche MAC dans Castalia étaient difficiles à appréhender. Donc nous sommes passé à la seconde solution, nous avons créé une nouvelle couche MAC afin de pouvoir lui implémenter les fonctionnalités désirées au fur et à mesure de l'avancement de notre étude.

La mise en place d'un nouveau module dans Castalia 3.2 est une tache très simple. Elle débute par la création d'un dossier au sein de la hiérarchie de fichiers sources de Castalia. L'emplacement est diffèrent selon le module que l'on souhaite développer. Dans notre cas nous développons un module MAC, on doit donc placer ce dossier à l'emplacement suivant :

*Castalia/src/node/communication/mac*

L'étape suivante de la mise en place du module, est la création d'une interface pour celui-ci. Cette interface est définie à l'aide d'un fichier propre au framework OMNet++. Ce fichier de type NED possède une syntaxe du même nom. Cette syntaxe, assez simple à prendre en main, permet de définir les entrées/sorties d'un module. Son utilisation permet en outre de paramétrer le module de façon dynamique. Ceci permet de tester et de régler le module sans avoir à passer par des phases de compilations qui peuvent être longues, alors que l'on a besoin de modifier que très peu de paramètres. Dans ce fichier nous devons indiquer où se situe le module dans la hiérarchie, puis que l'on étend le module de base **iMac**. Vient ensuite la définition du module en lui même. Celle-ci comporte une partie de déclaration des paramètres dynamiques du module, puis une partie qui concerne ses entrées/sorties.

Après avoir déclaré le module à l'aide du fichier NED, nous pouvons créer les fichiers sources C++. Les fichiers qui implémentent le module doivent avoir le même nom que le nom donné au module dans l'interface NED. Le fichier de déclaration C++ doit contenir un héritage du module de base. Dans le cas d'une couche mac il doit hériter de **VirtualMac**. Ce module comporte deux méthodes qui doivent être redéfinies au minimum, mais dans la pratique il est nécessaire de redéfinir six d'entre elles.

Tout ce qui précède est le minimum à faire pour avoir un module qui fonctionne. Cependant cela n'est pas suffisant pour des simulations réellement utilisables. En effet il faut également définir les paquets d'encapsulation de données propres au fonctionnement de la couche mac. Cette définition se fait dans un fichier d'extension *.msg*. La syntaxe du fichier est également en langage NED, mais à la différence d'un module mac qui lui étend simplement un autre module, le paquet de données doit étendre **MacPacket** et doit également être indiqué comme étant de type "packet". Pour que ce fichier soit utilisable il doit également exister les fichiers d'implémentation de celui-ci. Mais dans ce cas ci, ce n'est pas au développeur de le faire. Ces paquets étant destinés à être envoyés entre les différentes couches de simulation de Castalia et même de manière plus générale de OMNet++, il doivent comporter une structure interne propre à l'architecture choisie pour la simulation. C'est pour cela que le framework OMNet++ met à disposition un outil, appelé "opp msgc", qui génère automatiquement l'implémentation du paquet de données.

### 4.1.2 L'application de simulation

Pour effectuer une simulation dans Castalia nous devons choisir chacun des éléments nécessaires à celle-ci. Nous aurions pu développer l'intégralité des modules nécessaires à la simulation, mais notre objectif dans ce projet étant de concevoir une couche mac, nous avons décidé de réutiliser des modules déjà existants pour toutes les autre parties de la simulation. Nous avons notamment utilisé pour la couche application, le module *valueReporting* en ne modifiant que la couche mac par défaut afin qu'il utilise celle que nous avons développée. Cette application consiste à propager régulièrement, jusqu'à un puits de données, des informations recueillies au niveau de chaque nœud par un capteur.

## 4.2 Présentation de la couche mac LDSMac

### 4.2.1 Définition

Afin de permettre une meilleure compréhension du sujet, nous devons poser les définitions suivante :

- Distance d'un nœud à l'origine : nombre minimal de sauts entre un nœud et le nœud d'origine.
- Durée de navigation d'un paquet : nombre de sauts qu'a effectué un paquet avant d'arriver à un nœud.

### 4.2.2 Principes de fonctionnement

L'objectif final du projet est de créer une couche MAC pour réseaux sans fils à capteur autonome. Cela engendre donc des contraintes fortes au niveau énergétique. Le gros problème des capteurs sans-fil est que le seul moyen pour qu'ils consomment peu d'énergie est de les éteindre complètement. C'est sur cette idée que s'est basé le développement de la couche TMAC. Elle s'appuie sur une synchronisation des périodes d'écoute et de transmission afin de pouvoir éteindre le capteur lorsque l'on est sur qu'il n'aura pas à recevoir de données ni à les retransmettre. Cette démarche est très bonne pour économiser de l'énergie, mais elle a également plusieurs défauts. Elle peut entraîner des périodes d'attentes inutiles car aucun nœud voisin ne veut émettre et des périodes où le nœud peut transmettre alors qu'il n'a rien à émettre. Le second défaut que l'on peut lui reprocher est le fait que chaque nœud doit attendre son tour pour émettre et même s'il obtient son tour assez rapidement il ne peut émettre tout ce qu'il a à transmettre d'un seul coup parce qu'il se peut que sa durée d'émission soit plus longue que le temps qui lui est imparti.

Pour notre couche mac nous avons donc adopté une approche similaire. C'est à dire que l'on passe par des phases ou le capteur est complètement éteint, mais nous sommes passés à un modèle de transmission de type asynchrone. C'est à dire que le nœud peut essayer d'émettre des données dès qu'il en a le besoin. Cette technique permet de réduire la latence de transmission mais entraîne cependant plus de périodes où le capteur est alimenté.
Cependant les nœuds peuvent aboutir à des économies d'énergie lorsqu'ils ne servent pas à la transmission d'un paquet. On créé un tunnel de transmission de donnée entre l'émetteur et le récepteur, ainsi tous les nœuds qui reçoivent des données mais qui ne sont pas dans ce tunnel sont éteints lors de la transmission.(voir figure 2). Ceux-ci restent éteints pendant un certain temps, temps au bout duquel ceux-ci écoutent s'il y a des transferts de données. Si il

y a des transferts de données alors ceux-ci s'éteignent à nouveau. Les nœuds éteints effectuent cette phase d'écoute/extinction tant que des transferts sont en cours. Dès que plus aucun transfert n'est en cours les nœuds se remettent en attente de transmission.

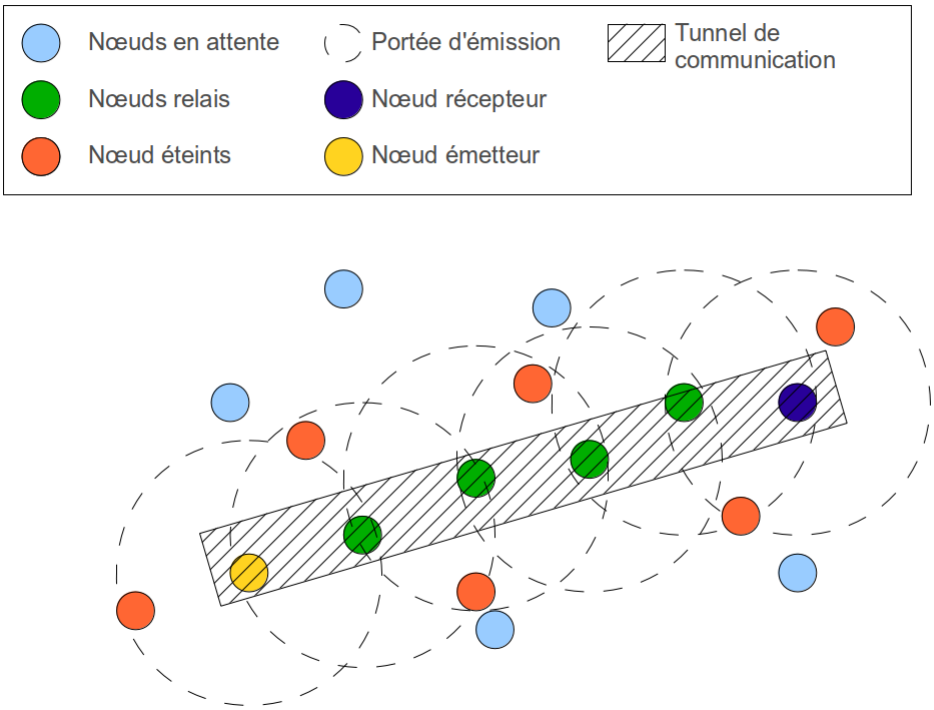


FIGURE 2 – État des nœuds pendant un transfert de données

Comme vu dans l'exemple de la figure 2 dès que le tunnel est créé il est très simple d'effectuer la communication des données entre les différents nœuds, cependant cette approche peut engendrer des problèmes de duplication de paquets.

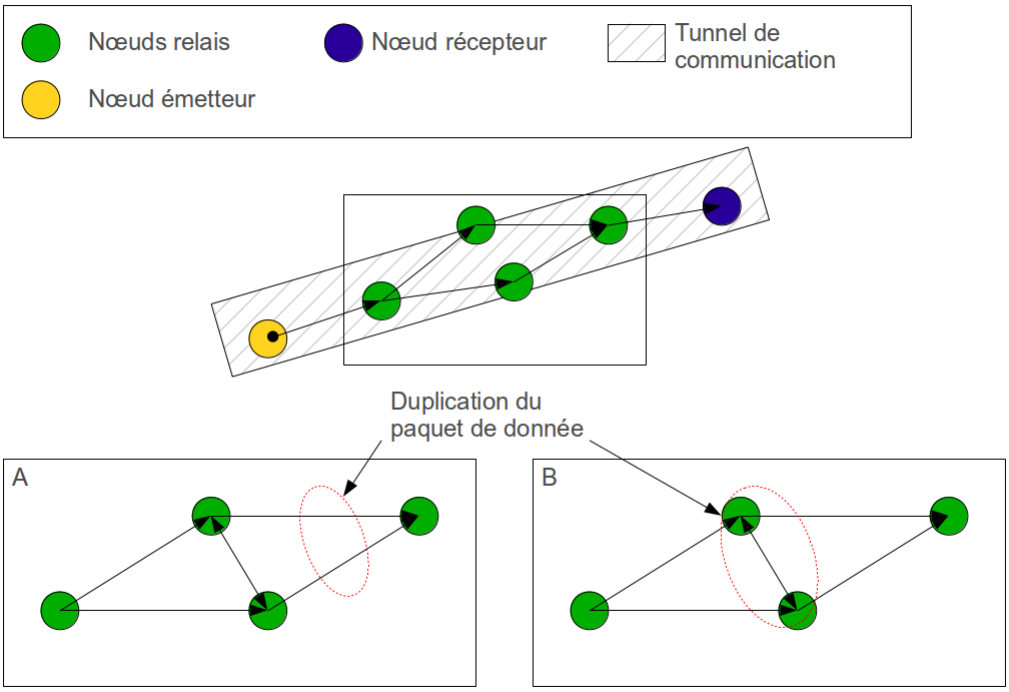


FIGURE 3 – Types de duplication de paquet rencontrée

Dans la figure 3 on voit les deux types de duplications de paquets que nous avons rencontré. Pour résoudre ces problèmes de duplication nous avons choisi une approche différente selon le type. Le premier que nous avons résolu est celui de la figure 3-B. Celle-ci apparaît lorsque deux nœuds proches sont tous les deux dans le tunnel de communication. La première solution que nous avons apporté à ce problème se situe au niveau de la retransmission de paquet. En effet si un nœud reçoit un paquet qui a une durée de navigation supérieure à la distance que le nœud a de l'origine celui-ci ne le retransmet pas.
Pour le problème montré dans la figure 3-A, le nœud reçoit le même paquet avec une durée de navigation inférieure a sa distance à l'origine. Nous avons résolu le problème en enregistrant les paquets déjà reçus afin de ne retransmettre que les nouveaux paquets. Cependant cette solution peut être coûteuse en terme d'espace mémoire donc nous avons cherché à améliorer cette solution.

C'est dans cette recherche d'amélioration que nous sommes arrivé à une solution qui résout les deux problèmes cités. On voit très vite que le problème qui se pose est que deux nœuds peuvent retransmettre une même donnée. Il suffit donc d'éteindre un des deux nœuds. La solution proposé se situe donc directement au niveau du protocole de construction du tunnel. Lorsque deux nœuds sont aptes à retransmettre une donnée, un seul des deux est choisi et l'autre est éteint. Généralement le nœud choisi sera celui qui a validé le chemin en premier.
Cette technique permet en plus de résoudre les problèmes de duplication, de faire des économies d'énergie car un des nœuds est éteint. Le protocole que nous avons défini abouti donc à la représentation montré dans la figure 4

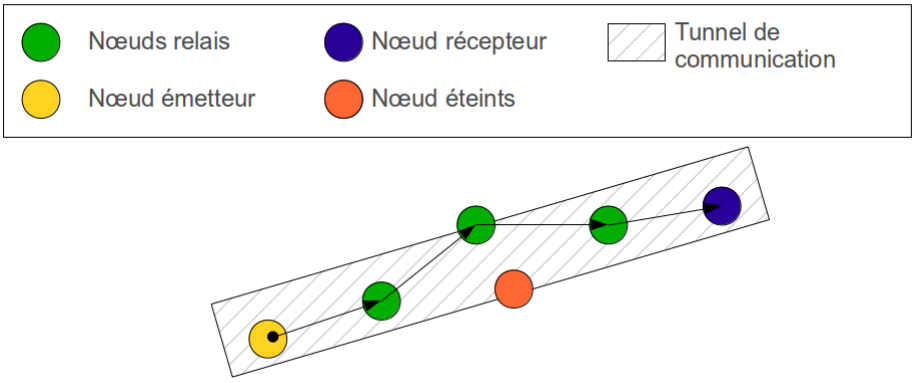


FIGURE 4 – Types de duplication de paquet rencontrée

### 4.2.3 Protocole de construction du tube de communication

Dans la partie qui suit nous allons expliquer le protocole que nous avons mis en place pour établir un tunnel de communication entre deux nœuds. Or comme il vous sera montré dans la suite du rapport, le protocole n'est pas encore finalisé, ce qui implique que plusieurs problèmes restent encore à résoudre. Pour une explication cohérente nous allons donc poser les conditions nécessaires, dans l'état actuel des choses, à la bonne exécution du protocole.

- Un seul des nœuds du réseau cherche à émettre à la fois.
- Afin de créer un tunnel entre deux noeuds, chaque nœud doit connaître la localisation dans l'espace, de lui même, du nœud émetteur et du nœud récepteur afin de pouvoir calculer son appartenance au tunnel

– La largeur du tunnel fait également partie des paramètres nécessaires au calcul d'appartenance au tunnel.

La construction de ce tunnel d'effectue sur un principe de réservation. C'est à dire que le nœud qui cherche à envoyer des données doit demander l'autorisation sur le réseau. Cette réservation se déroule en trois grandes étapes. L'émission d'un paquet de réservation, la confirmation de retransmission ou réception, et l'envoi de donnée. Nous allons voir de manière détaillée à l'aide des diverses figures qui suivent, le déroulement du protocole. Dans les schémas qui suivent nous allons conserver la même convention de couleurs afin de ne pas avoir à la répéter sur chacun d'eux.

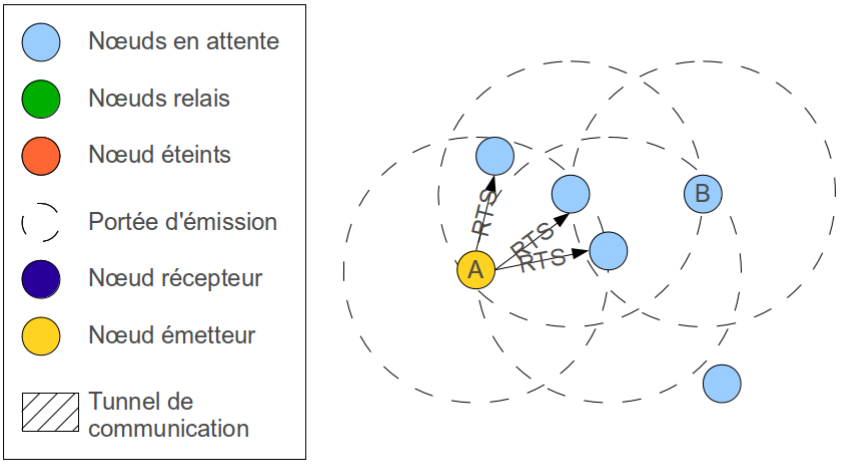


FIGURE 5 – Émission d'un paquet RTS

Lorsqu'un nœud veut émettre des données il doit émettre un paquet RTS (Ready To Send) qui contient l'id de la source de données,[2] l'id de la destination et l'id d'où provient le paquet, dans ce cas le premier et le troisième champs sont les mêmes.

Suite à cela chacun des nœuds qui recevra le paquet va calculer s'il appartient au tunnel. Si le nœud est un nœud relais et qu'il est dans le tunnel alors il émet à son tour un RTS identique mis à part le champ de l'émetteur du paquet qui lui change à chaque émission. Les nœuds qui ne font pas partie du tunnel s'éteignent pendant un certain temps comme expliqué précédemment.

2. Chaque nœud possède un id unique qui permet de le repérer.

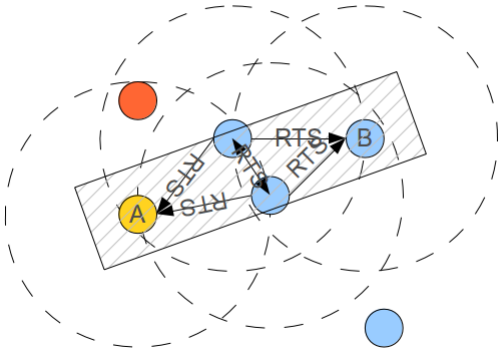


FIGURE 6 – Émission d'un paquet RTS de confirmation et de demande

Dans le schéma précédant on voit que le nœud initial reçoit le RTS ce qui le met en attente car cela lui indique qu'il y a bien un nœud qui peut assurer la retransmission de ses données. On se retrouve ici par contre avec le problème évoqué plus tôt où deux nœuds sont proches ainsi celui qui émettra en premier restera actif comme l'indiquait la méthode explicitée précédemment.

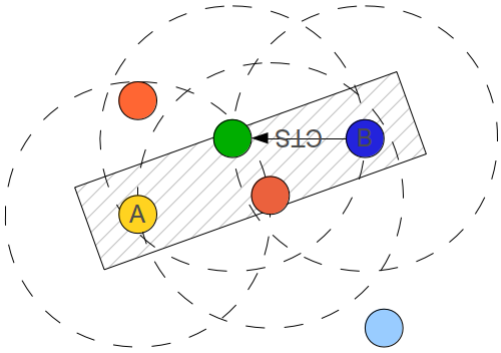


FIGURE 7 – Émission d'un paquet CTS de confirmation

Ici on voit que le nœud cible a été atteint ainsi celui-ci émet un paquet de confirmation CTS (Clear To Send) qui indique qu'il est prêt à recevoir des données. Dans le schéma cela n'est pas montré mais si un nœud reçoit un

CTS alors qu'il est en attente celui-ci s'éteint également. Suite à cela le nœud relais retransmet le CTS au nœud source.

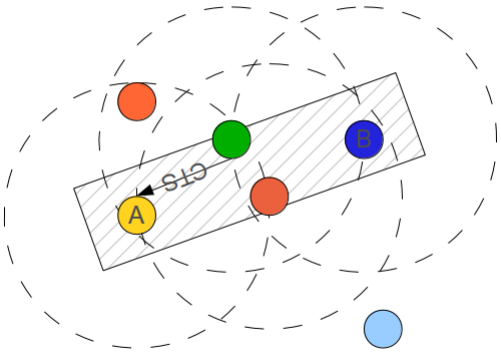


FIGURE 8 – Émission d'un paquet CTS de confirmation au noeud source

Puis vient l'envoi de données. Dans la figure suivante on voit que la source envoie son paquet de données au nœud relais. Puis on voit le nœud relais qui retransmet le paquet au nœud cible.

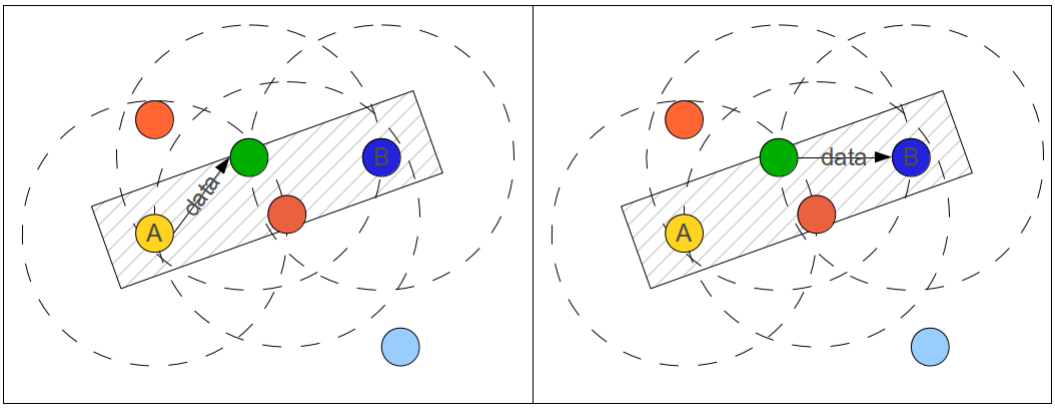


FIGURE 9 – Transfert de données entre le nœud source et le nœud cible par le biais d'un relais

Lorsque la source émet son dernier paquet elle le marque dans le paquet afin que les nœuds relais et cibles se remettent après réception en état d'attente.

Une autre option implémentée qui ne sera pas présentée ici afin de simplifier le rapport est l'utilisation des paquets d'acquittement. Si on le souhaite on peut utiliser cette option afin de sécuriser le transfert de données. Elle contraint les nœud à émettre une confirmation de réception de paquet à chaque fois que cela se produit.

#### 4.2.4 Problèmes rencontrés non résolus

A l'heure de la rédaction de ce rapport plusieurs problèmes restent encore à résoudre. Un d'eux est relatif à l'approche asynchrone. Lorsque plusieurs nœud cherchent à émettre en même temps, cela créé une situation de compétition pour le support. Or nous n'avons toujours pas implémenté une solution efficace à ce problème, cependant nous essayons de résoudre le problème à l'aide d'une technique d'attente à durée aléatoire, pour les nœuds voisins et des attentes à durée fixe pour les temps de confirmation de chemin.

Pour nos simulations nous nous sommes également basés sur le fait que chaque nœud connaît au moins la position des nœuds sources et cibles en plus de la sienne. Or cela peut être contraignant à mettre en œuvre, donc ce point doit être étudié tout particulièrement.

# 5 Gestion de projet

## 5.1 Cahier des charges

### 5.1.1 Description du système actuel

Aujourd'hui il existe plusieurs couches mac qui permettent de gérer les réseaux de capteurs sans-fil autonome ainsi que la gestion de leur consommation d'énergie. Cependant, ces couches MAC utilisent toutes une communication à temps partagé de type synchrone. Ce projet a pour but de développer une couche MAC avec un fonctionnement asynchrone.

### 5.1.2 Objectifs de l'appel d'offre

L'objectif de l'appel d'offre est de définir et d'implémenter une nouvelle couche MAC permettant la communication de capteurs sans-fils et plus précisément la propagation d'un message depuis un capteur source jusqu'à un capteur de destination. En effet lorsqu'un capteur souhaite transmettre une information à un destinataire, les autres capteurs se trouvant sur le "chemin" doivent stopper leur émission et servir de relais pour faire passer ce

message. Le "chemin" est défini dans un tunnel de communication, qui est la zone la plus directe pour accéder aux destinataire. Ce tunnel permet de sélectionner les nœuds susceptibles de pouvoir propager l'information dans la bonne direction, le plus rapidement possible.

Plusieurs problématiques se posent alors :

- les nœuds ne doivent pas réémettre une information déjà envoyée,
- les nœuds hors du tunnel ne doivent pas traiter l'information reçue et donc ne pas stopper leurs émissions, mais celles-ci ne doivent pas interférer le transit du message considéré, il est donc nécessaire de définir une priorité de messages.

Pour cela nous allons utiliser le simulateur de réseaux sans-fil Castalia.

### 5.1.3 Volumes d'informations

Les volumes d'informations seront de deux types. Nous fournirons le code source du module de Castalia ainsi que sa documentation, mais également une description protocole de communication de notre couche MAC.

## 5.2 Les besoins

Il est nécessaire d'avoir installé OMNeT++ et Castalia 3.2 pour pouvoir développer les modules ainsi qu'effectuer les tests nécessaires.

## 5.3 Diagrammes de Gantt

Voici le premier diagramme de Gantt établit en début de projet.

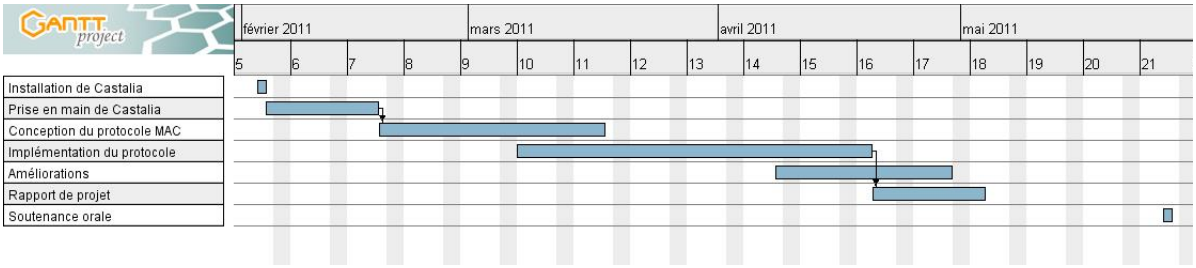


FIGURE 10 – Premier diagramme de Gantt

Premièrement nous avons installé le logiciel Castalia. Ensuite nous avons étudié sa documentation et pris en main son fonctionnement grâce aux exemples

fournis et modules déjà présents. La conception du protocole MAC et son implémentation a pris plus de temps que prévu, c'est pourquoi nous n'avons pas pu nous pencher sur les améliorations possibles à apporter au protocole pour notamment la gestion de l'énergie des capteurs.

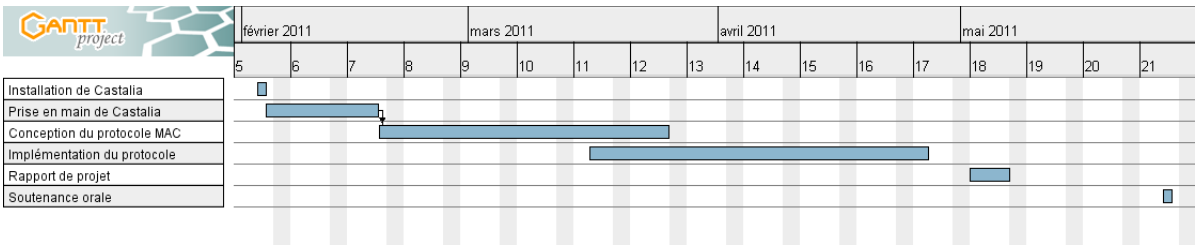


FIGURE 11 – Deuxième diagramme de Gantt

# 6 Conclusion

Durant ce projet nous avons développé une nouvelle couche MAC afin de permette les communications dans un réseaux de capteurs sans-fil et notamment le transfert de message d'un capteur à l'autre sans interférence des autres capteurs environnants. Une problématique importante des capteurs est la gestion de leur énergie et c'est à partir de ce point que l'on pourrait envisager d'améliorer la couche MAC que nous avons produite.

La gestion de projet est une bonne expérience en ce qui concerne la gestion du temps. Une bonne organisation est nécessaire pour remplir les objectifs prévus sans prendre de retard. C'est un point que nous avons eu du mal à appréhender. En effet travailler sur un projet depuis sa conception en définissant nous même les objectifs est une chose que nous n'avons pas beaucoup pratiqué dans notre scolarité.

Notre projet étant un projet de recherche, nous avons du nous débrouiller par nous même pour recueillir des informations et trouver nous même des solutions pour proposer une solution à la problématique posée.

# Références

[1] *Castalia A simulator for Wireless Sensor Networks and Body Area Networks, User's manuel.*

[2] *OMNeT++ User Manual.*