Modélisation de la consommation d'énergie En vue de la conception conjointe (matériel/logiciel) des applications embarquées. Application aux réseaux de capteurs sans fil (wsn).

Aina Randrianarisaina

Public Domain

Thèse de Doctorat

Aina Andriamampianina
RANDRIANARISAINA

Mémoire présenté en vue de l'obtention
du grade de Docteur de l'Université de Nantes
Sous le label de l'Université Nantes Angers Le Mans

Discipline : Electronique
Spécialité : Systèmes Embarqués
Laboratoire : IETR UMR 6164

Soutenue le 6 février 2015

École doctorale Sciences et Technologies de l'Information et Mathématiques (STIM)

MODÉLISATION DE LA CONSOMMATION D'ÉNERGIE EN VUE DE LA CONCEPTION CONJOINTE (MATÉRIEL/LOGICIEL) DES APPLICATIONS EMBARQUÉES. APPLICATION AUX RÉSEAUX DE CAPTEURS SANS FIL (WSN).

JURY

Présidente	**Mme Nathalie JULIEN**, Professeur, Université Bretagne Sud
Rapporteurs	**Mme Cécile BELLEUDY**, Maître de Conférences/HDR, Université de Nice Sophia Antipolis
	M. Thierry VAL, Professeur, Université de Toulouse 2 - IUT Blagnac
Directeur de Thèse	**M. Pascal CHARGÉ**, Professeur, Ecole polytechnique de l'université de Nantes
Encadrant	**M. Olivier PASQUIER**, Maître de Conférences Ecole polytechnique de l'université de Nantes

Remerciements

Tout d'abord, je tiens à remercier mon directeur de thèse Mr. Pascal CHARGE et mon co-encadrant Mr. Olivier PASQUIER pour leur disponibilité, leurs idées et leurs conseils durant ces trois années de thèse.

Ce travail a été réalisé au sein du laboratoire IETR (Institut d'Electronique et de Télécommunications de Rennes) UMR CNRS 6164, site Nantes. Ainsi, je tiens à remercier tous les personnels du laboratoire de m'avoir offert un excellent cadre de travail.

Un grand merci à ma famille qui m'a toujours soutenu et qui m'a donné la force et le courage nécessaires pour mener à bien cette thèse.

J'adresse aussi mes sincères remerciements à tous mes amis, qui m'ont soutenu de près ou de loin durant ces trois années de thèse.

Un grand merci à tous.

Table des matières

Introduction générale

La miniaturisation des équipements électroniques embarqués ne cesse d'augmenter alors qu'ils sont destinés à accomplir des tâches de plus en plus complexes. En raison de leur petite taille et de la faible consommation d'énergie qui leur est imposée, les constituants des réseaux de capteurs ne peuvent pas rivaliser avec les ordinateurs, smart phones, tablettes ou autres terminaux de poche en termes de capacité de traitement de données, de stockage et de communication. Il est donc important de pouvoir considérer la consommation d'énergie très tôt dans le cycle de conception et développement des systèmes embarqués.

Notre étude se concentre sur un cas particulier de ces types d'équipements miniaturisés : les noeuds capteurs sans fil. Ces derniers sont des dispositifs électroniques communicants entre eux par des liaisons sans fil, et dont les fonctions premières sont de collecter et de transmettre des données de manière totalement autonome en énergie. Ces systèmes communicants constituent ainsi le réseau et sont appelés noeuds capteurs. Ils sont nécessairement équipés chacun d'une unité de captage afin d'observer l'environnement dans lequel ils sont placés. L'association de ces noeuds forme le réseau de capteurs sans fil ou WSN (Wireless Sensor Network). Les noeuds capteurs peuvent être fixes ou mobiles. Leur position dans le réseau n'est pas obligatoirement prédéterminée car ils peuvent être dispersés aléatoirement dans une zone géographique (avec un algorithme et un protocole d'auto-organisation) ou bien leurs positions peuvent être spécifiées et organisées dans une zone ciblée (topologie pré-configurée). Un réseau de capteurs est connecté par un point à un système hôte (voir figure 1). Ce point de connexion est appelé "puits" (souvent "sink"). Toutes les données récoltées par chaque noeud sont routées jusqu'au puits ou point de collecte par l'intermédiaire d'une architecture multi-sauts ou seulement d'un saut selon la topologie du réseau.

Les noeuds capteurs utilisent des technologies sans fil pour accéder à l'ensemble du réseau, et il existe plusieurs catégories de ces technologies sans fil.

Un réseau sans fil, sans se limiter aux réseaux de capteurs ou WSN (Wireless Sensor Networks), est un réseau basé sur une communication par ondes électromagné-

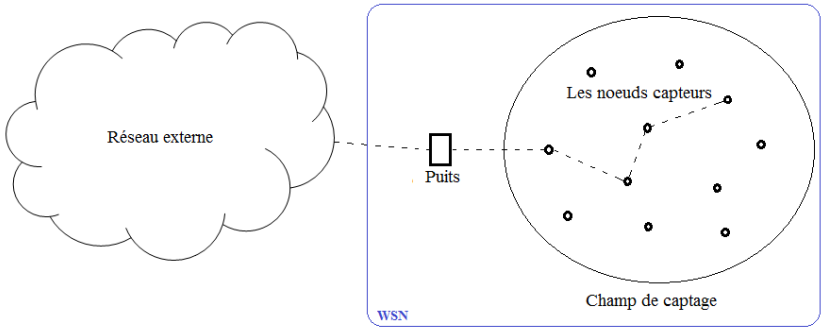

FIGURE 1 – Architecture globale

tiques (radio, ultra sons, etc). Ainsi, les terminaux peuvent se déplacer dans un périmètre géographique plus ou moins étendu, tout en restant connectés. Plusieurs technologies de communication sans fil ont été dévéloppées. Une technologie diffère des autres notamment selon la fréquence d'émission utilisée, le débit de transmission et la portée des transmissions.

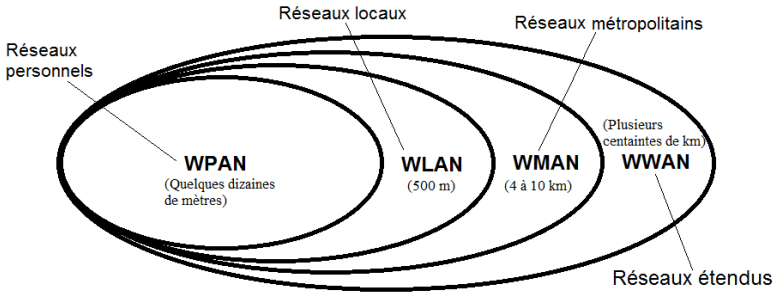

FIGURE 2 – Catégorie des réseaux sans fil

La figure 2 décrit les principales familles de réseau sans fil. En effet, il existe différentes catégories de réseaux sans fil ; les WWAN (Wireless Wide Area Networks), les WMAN (Wireless Metropolitan Area Network), les WLAN (Wireless Local Area Network) et les WPAN (Wireless Personal Area Network). Chaque technologie sans fil a ses propres normes, définie par ses portées, ses débits et ses fréquences de communications.

Les WWAN [1] sont des réseaux étendus, utilisés souvent dans les réseaux cellulaires mobiles. Cette catégorie de réseau est destinée aux opérateurs de télécommunication. Les principales technologies sont les suivantes : GSM (Global System for Mobile Communication), 3G, 4G. Ce type de réseau permet de relier plusieurs régions et interconnecte plusieurs villes. La portée d'un tel réseau peut atteindre jusqu'à plusieurs centaines de kilomètres.

Les réseaux WMAN [1] sont connus également sous le nom de boucle locale radio (BLR). La technologie WiMAX (Worldwide Interoperability for Microwave Access) et comme d'autres technologies appartenant à la famille WMAN sont basés sur la norme IEEE 802.16 et offrent un débit utile de 1 à 10 Mbps pour une portée de 4 à 40 km.

Les WLAN [1] sont des réseaux locaux sans fil permettant de couvrir des équipements informatiques d'une même entreprise, d'une habitation d'un particulier, ou d'une même université, avec une portée de plusieurs centaines de mètres et un débit de quelques Mbps à quelques Gbps. Il existe plusieurs technologies concurrentes, par exemple le WiFi (Wireless Fidelity) basé sur les normes IEEE 802.11 et le HiperLAN2 (High Performance Radio LAN 2.0).

La catégorie WPAN [1] est un réseau individuel sans fil ou encore un réseau domestique sans fil, qui a une faible portée (de l'ordre de quelques dizaines de mètres). Cette catégorie de réseau est souvent basée sur les normes IEEE 802.15.x et souvent utilisée pour relier des équipements personnels. Par exemple des noeuds capteurs reliés à un téléphone portable ou à un smartphone ou une tablette ou encore à un ordinateur via un support sans fil. Il existe plusieurs technologies utilisées dans cette catégorie de réseau : Bluetooth, Zigbee, etc.

Bien que les réseaux de capteurs sans fil puissent être de grande dimension, ils sont souvent classés dans la catégorie des réseaux personnels sans fil (WPAN) [2]. La technologie la plus utilisée dans les WSN est la technologie Zigbee basée sur la norme IEEE 802.15.4 et qui nous offre une portée typique de quelques dizaines de mètres.

La figure 1 montre un exemple d'acheminement des données entre les éléments du réseau. Ainsi, la figure 1 représente l'architecture globale du réseau. Ce dernier est donc composé de plusieurs noeuds capteurs, il peut y avoir un ou plusieurs points de collecte appelés puits et éventuellement un réseau externe (internet ou satellite). Les noeuds capteurs sont distribués dans une zone de capture. Chaque noeud peut faire l'acquisition de données et peut en même temps jouer le rôle de relai selon le type du réseau. Les données acquises par chacun des noeuds seront routées jusqu'au puits. Ce dernier est un point d'accès vers un éventuel réseau extérieur puisqu'il peut communiquer les données collectées, par exemple via internet, jusqu'à un moniteur de contrôle.

L'architecture globale du réseau peut être moins élargie, c'est-à-dire que les données recoltées par les noeuds peuvent être traitées ou stockées directement au niveau du puits (pas de réseau externe).

La consommation d'énergie est l'un des principaux problèmes rencontrés dans les WSN puisque les noeuds capteurs sont bien souvent autonomes en énergie sauf souvent

le puits. Ainsi, cette contrainte en énergie est relative à notre motivation.

Depuis l'émergence des réseaux capteurs sans fil, l'objectif n'est plus le même que dans les réseaux filaires ou dans les réseaux WIFI, on ne cherche plus seulement à maximiser le débit de communication mais plutôt bien souvent à maximiser la durée de vie des noeuds capteurs ainsi que celle du réseau. Ceci passe notamment par le développement d'applications et systèmes optimisés en terme de consommation et de compromis entre complexité et fiabilité.

En outre, les noeuds capteurs doivent être résistants pour pouvoir subsister dans les conditions les plus extrêmes (chaleur, pluie, feu, etc). Mais la contrainte à laquelle nous nous intéressons est liée à l'énergie. Effectivement, la perte importante de noeuds peut engendrer une fin de cycle de vie d'un réseau puisque ceci peut causer des pertes de communications dues à un isolement de noeuds ou une trop grande distance entre les noeuds. Par conséquent, il est primordial que les batteries durent le plus longtemps possible car il n'est pas toujours possible de changer la batterie du noeud une fois qu'il est placé dans le réseau.

Cependant, dans le domaine des noeuds capteurs sans fil, les applications embarquées sont de plus en plus sensibles à la consommation d'énergie. Cette dernière dépend non seulement de l'autonomie des éléments du réseau, mais aussi des applications. Par ailleurs, selon sa taille, un noeud capteur sans fil est limité en énergie. Dans la plupart des cas, remplacer la batterie est impossible. Ce qui signifie que la durée de vie d'un noeud capteur dépend principalement de la durée de vie de la batterie et plus précisement de la manière dont on l'utilise. Malgré le nombre des travaux concernant la récupération d'énergie (energy harvesting), notre étude n'est pas orientée sur cet aspect, mais plutôt sur la façon dont le noeud capteur consomme l'énergie.

Dans un réseau de capteurs, chaque noeud collecte des données. Le dysfonctionnement de quelques noeuds implique un changement de la topologie du réseau et un re-routage des paquets pour les réseaux à multi-sauts. Toutes ces opérations sont gourmandes en énergie. C'est la raison pour laquelle, les recherches actuelles se concentrent principalement sur les moyens de réduire cette consommation. Il est donc important d'estimer le coût en énergie.

Ainsi, l'objectif de notre étude est de modéliser la consommation d'énergie des noeuds capteurs sans fil dans le but de fournir des informations aux concepteurs dès les premières étapes de la conception du système. Pour bien considérer la durée de vie des noeuds et du réseau, les modèles de noeuds doivent tenir compte de l'impact de la consommation d'énergie sur leurs différentes activités dans le réseau. Nous adopterons donc la modélisation de la consommation d'énergie des noeuds capteurs avec une ap-

proche par fonctions. Cette approche permet d'obtenir un modèle du noeud facilement manipulable et surtout elle donne la possibilité d'avoir un impact plus conséquent en optimisation. De plus, cette approche par fonctions permet d'identifier facilement le lien entre la consommation d'énergie et les activités du noeud dans le réseau. Ainsi, elle permet de fournir des informations aux concepteurs dès les premières étapes de conception du noeud puisque la conception et l'utilisation des différentes fonctions dans les noeuds sont liées aux contraintes applicatives.

Pour atteindre cet objectif, nous allons planifier la suite de ce document en quatre grande partie. Le premier chapitre est consacré aux généralités des noeuds dans les réseaux de capteurs sans fil. Dans le deuxième chapitre, nous parlons des modèles et outils de simulation pour les WSN. Nous définissons les différents niveaux d'abstraction de modélisation d'un système, qui permet d'avoir des vues différentes sur un même modèle. Ensuite, nous décrivons quelques modèles et outils de simulations qui existent dans la littérature et qui traitent la même problématique. Et finalement, nous évoquons certaines limites de ces simulateurs et l'interêt de notre étude. Le troisième chapitre est consacré à notre contribution sur la modélisation de la consommation d'énergie des noeuds capteurs sans fil avec une approche par fonctions. Nous commencons par décrire quelques fonctions les plus utilisées par les noeuds capteurs. Puis, nous décrivons la modélisation d'une fonction et la modélisation d'un noeud par cette approche. Ensuite, nous établissons quelques scénarios pour simuler les modèles de noeuds et du réseau. Dans le chapitre quatre, différentes simulations sont proposées pour illustrer l'interêt de l'approche par fonctions. Nous discutons ensuite des résultats de simulations qui permettent de valoriser nos modèles et notre approche. Enfin, des relevés de mesures réelles de consommation d'énergie des noeuds capteurs sont fournis. Ces résultats de mesures sont ensuite comparés avec des résultats de simulations pour valider notre approche. Ce document s'achève par une conclusion générale et quelques perspectives.

PUBLICATIONS

- **Conférences Internationales :**
  - A. Randrianarisaina, O. Pasquier, P. Chargé. Energy Consumption Modeling of Smart Nodes with a Function Approach. Conference on Design & Architectures for Signal & Image Processing (DASIP), Madrid (Spain) 2014.
  - A. Randrianarisaina, O. Pasquier, P. Chargé. A function Approach for Simple

Wireless Sensor Node Energy Consumption Modeling. Forum on Specification & Design Languages (FDL), Paris (France) 2013.

- **Colloque National :**
  - A. Randrianarisaina, O. Pasquier, P. Chargé. Modélisation fonctionnelle de la consommation d'énergie dans un noeud capteur sans fil, GDR SoC-SiP CNRS, Lyon : France (2013).
- **Workshop :**
  - A. A. Randrianarisaina, O. Pasquier, P. Chargé. A Functional Approach for Wireless Sensor Node Energy Consumption Modeling. Accepted in 2nd Sino-French Workshop on Research Collaborations in Information and Communication Technologies (SIFWICT 2013), Guangzhou, China, Jun. 2013.

1

Généralités sur les noeuds des WSN

Un noeud capteur est un ensemble de dispositifs électroniques, très souvent autonome en terme d'énergie, équipé d'un dispositif d'acquisition. Cette dernière unité lui permet de récolter des données. L'interconnexion entre ces noeuds de capteurs forme le réseau de capteurs sans fil ou WSN. Selon le type du réseau, un noeud peut faire seulement capture ou capture et relais en même temps. Dans ce dernier cas, un noeud peut recevoir des données venant des autres noeuds et peut les retransmettre vers d'autres noeuds, avec ses propres données.

Les progrès récents dans la technologie des systèmes micro-électromécaniques (Micro Electro-Mechanical Systems MEMS) utilisés pour les capteurs, les communications sans fil, et l'électronique numérique ont permis le développement de petits dispositifs peu coûteux, de faible puissance, et qui peuvent communiquer entre eux. Ces avancées technologiques ont largement favorisé l'essor des réseaux de capteurs sans fil pour de nombreuses applications et des contextes très variés. Ainsi les noeuds qui constituent ces réseaux sont des dispositifs électroniques conçus et dimensionnés pour répondre à des besoins de surveillance, collecte et transport de données, communication, etc. Ainsi, ces dispositifs intègrent une unité d'acquisition de données environnementales (température, humidité, vibrations, luminosité, ...) pouvant être transformées en grandeurs numériques, une unité de traitement qui permet d'agréger les données collectées, une unité de stockage, un module de transmission radio, et une source d'alimentation (pile ou batterie). Ces noeuds coopèrent entre eux pour former

une infrastructure de communication appelée réseau de capteurs sans fil ou WSN.

Voici quelques exemples concrets sur l'application des WSN.

Dans le domaine environnemental, on peut citer deux projets, ce qui est loin d'être exhaustif. En premier exemple, un projet présenté dans [3], étudie, en se basant sur un WSN, les oiseaux de mer dans une réserve naturelle au Royaume-Uni. Le projet WildCENSE [4] consiste lui à suivre les déplacements des antilopes Nilgaud en Inde.

Dans le domaine militaire, un WSN a été déployé dans la base militaire de MacDill (Air Force Base) à Tampa (Floride) pour détecter et suivre les mouvements d'objets mobiles intrus [5]. Le projet WATS (Wide Area Tracking System) [6] consiste à utiliser un WSN qui a pour objectif de détecter les rayons gamma et les neutrons dans le but de dépister des dispositifs nucléaires. Il existe plusieurs projets similaires à JBREWS (Joint Biological Remote Early Warning System) [7] qui ont pour objectif de détecter et d'avertir les troupes sur l'utilisation d'armes biologiques.

Dans le domaine industriel, des travaux sont axés sur l'intégration d'une commande intelligente basée sur la technologie des capteurs/actionneurs sans fil [8] [9]. D'autres applications de WSN ont pour objectif de surveiller l'état d'un pont [10], de surveiller des installations oléoducs et gazoducs [11] [12] (vérification de la pression ou le débit circulant dans les tuyaux).

Un grand nombre d'applications de WSN est développé dans le domaine médical pour faire une télésurveillance des patients à domicile [13], équiper des patients par des capteurs relevant des informations médicales telles que le niveau d'oxygénation et les pulsations cardiaques (projet CodeBlue [14]). De même le projet Mercury [15] vise à surveiller les patients atteints de la maladie de Parkinson ou ceux qui souffrent d'épilepsie.

Cette brève présentation est loin d'être exhaustive car le nombre de domaines où les WSN sont utilisés est très grand ainsi que les utilisations des WSN qui sont faites dans chacun de ces domaines.

Ce chapitre a pour objectif de décrire les caractéristiques générales des noeuds et des réseaux de capteurs sans fil. Dans la première partie, nous aborderons les notions de topologies et de durée de vie d'un réseau de capteurs car elles sont essentielles pour la suite de nos travaux. Puis, nous décrivons les caractéristiques et le fonctionnement d'un noeud capteur sans fil. L'objectif est de montrer quelques variantes de noeuds avec les différents modules matériels et logiciels qui le composent. Ensuite, une discussion sur l'aspect surconsommation d'énergie des noeuds capteurs sans fil est présentée. Et finalement, ce chapitre s'achève par une conclusion.

1.1 Notions utiles sur les réseaux de capteurs

1.1.1 Les différentes topologies dans les WSN

La topologie détermine l'organisation des noeuds capteurs dans le réseau. Il existe en général 3 types de topologies du réseau : la topologie étoile, la topologie maillée, la topologie arbre la topologie cluster-tree. Une comparaison de la performance de ces topologies est illustrée dans [16].

- La topologie étoile

La topologie étoile est composée d'un noeud central appelé coordinateur ou sink du réseau de capteurs et d'une pluralité de noeuds capteurs sans fil (voir figure 1.1). Dans cette topologie, tous les noeuds capteurs transmettent leurs données directement vers le coordinateur. Ainsi, l'architecture d'une telle topologie est caractérisée par une transmission à un saut. Dans cette topologie, le noeud central a la responsabilité de contrôler et de coordonner les noeuds capteurs qui communiquent exclusivement avec lui. Il est en charge de relayer les données vers d'autres systèmes. L'intérêt de cette architecture est qu'elle limite au maximum le transport des données et est donc très performante, en plus d'être relativement simple.

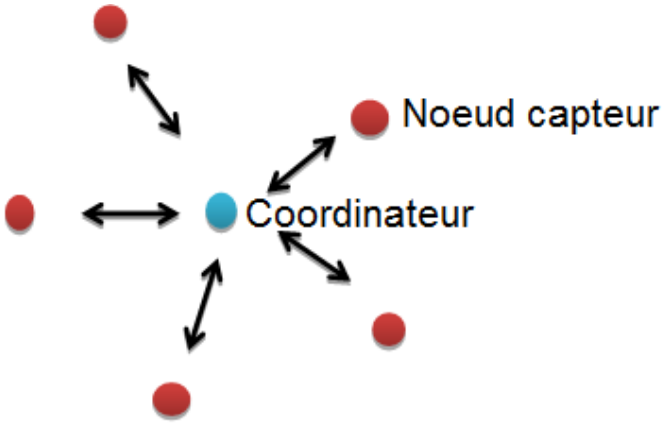

FIGURE 1.1 – Topologie étoile

La topologie étoile est particulièrement adaptée à deux types d'applications :

- Transmission radiofréquence en champs libre, où très peu d'obstacles sont présents dans la zone de couverture du réseau

- Transmission radiofréquence en dynamique : l'absence de routeur permet de réduire fortement le temps de latence dû aux mécanismes de re-routage des données.

Elle est surtout appliquée en médecine dans le cas de réseau à l'échelle humaine (Wireless Body Area Networks, les noeuds capteurs sont disséminés sur la peau du

patient et le coordinateur récolte les données provenant de ces noeuds capteurs.

- La topologie maillée

Dans les réseaux de capteurs mobiles classiques, la topologie maillée est la topologie standard. "MANET" ou Mobile Ad hoc NETwork est le type de topologie maillée le plus utilisé. C'est un réseau distribué sans infrastructure, les noeuds peuvent communiquer entre eux sans contrôle centralisé ni point d'accès.

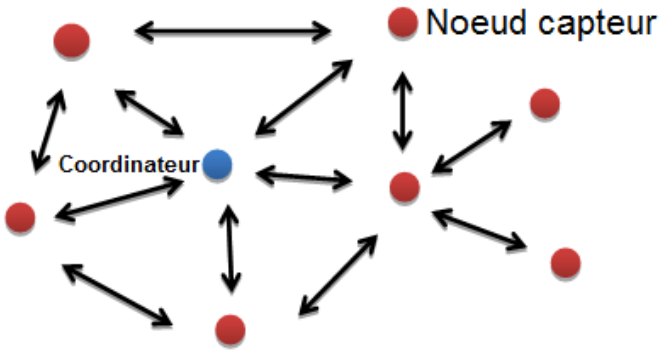

FIGURE 1.2 – Topologie maillée

La topologie maillée de la figure 1.2 est constituée d'un ensemble de noeuds reliés les uns aux autres et agissant comme des routeurs. D'ailleurs, tous les noeuds possèdent le même rôle. Chacun des noeuds transmet ses données à son voisin après avoir établi le meilleur chemin pour véhiculer les données à travers le réseau jusqu'au puits (coordinateur ou sink). Cette caractéristique permet de maintenir le fonctionnement du réseau en cas de panne d'un noeud. Cette topologie est alors plus robuste et flexible par rapport à la variation du nombre de noeuds. Quand il n'y a pas de transfert de données, les noeuds se mettent en mode veille pour économiser de l'énergie.

- La topologie arbre et cluster-tree

La topologie cluster-tree (voir 1.3) est partitionnée en groupes appelés "clusters". Un cluster est consituté d'un noeud particulier appelé "cluster-head" ou "tête de cluster" et d'autres noeuds. Ces derniers ne communiquent qu'avec leur "tête de cluster". Ce dernier est ensuite en charge de faire suivre les messages reçus vers le puits du réseau.

Cette topologie est alors hiérarchisée selon le rôle des éléments du réseau (coordinateur, têtes de cluster, noeuds). Il peut y avoir un changement de hiérachie entre les noeuds et les têtes de cluster en fonction de l'énergie disponible dans chaque noeud. Cela permet d'équilibrer la différence des niveaux d'énergie de tous les noeuds afin d'éviter la disparition ou l'isolement d'un noeud et de mieux prolonger la durée de vie du réseau présentée par la suite. Dans les réseaux où l'on tient surtout compte de

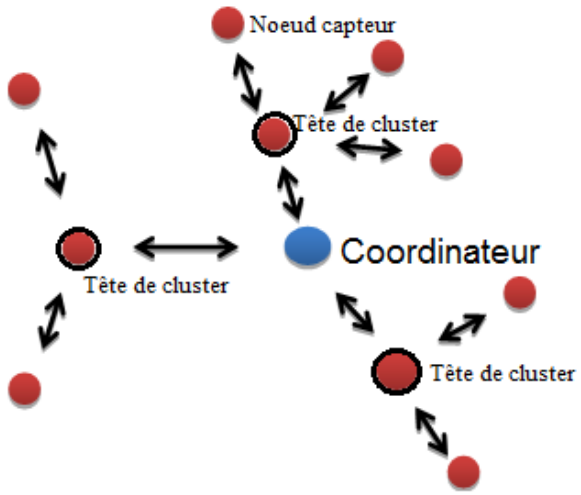

FIGURE 1.3 – Topologie cluster-tree

l'énergie, plusieurs travaux et recherches comme dans [17] [18] [19] ont affirmé que cette topologie cluster-tree est la mieux adaptée pour équilibrer le niveau d'énergie des noeuds dans le réseau.

En adoptant la topologie cluster-tree, chaque noeud dans le réseau est souvent muni d'un dispositif pour estimer le niveau de batterie du noeud. Ensuite, un protocole spécial doit être installé pour élire la prochaine tête de cluster suivant les niveaux de batterie de tous les noeuds dans le réseau.

La topologie arbre est parfois similaire à la topologie cluster-tree, seulement il n'y a pas de tête de cluster mais ce sont les noeuds simples qui jouent le rôle de relai dans le réseau.

1.1.2 La durée de vie d'un réseau de capteurs

La durée de vie d'un réseau de capteurs est généralement définie par le temps durant lequel le réseau, selon les cas suivants, est capable de maintenir assez de connectivité, couvrir la zone de captage ou garder son taux de perte de noeud inférieur à un certain niveau. Il peut y avoir d'autres définitions de la durée de vie du réseau liées à d'autres paramètres du réseau. La durée de vie d'un réseau de capteurs est donc liée à la durée de vie des noeuds. Et la durée de vie d'un noeud dépend principalement de la durée de vie de sa batterie. Cette dernière dépend des énergies consommées par les différents modules qui composent le noeud, par la technologie de la batterie et la façon de s'en servir [20].

Plusieurs définitions de durée de vie d'un réseau de capteurs sans fil ont été propo-

sées dans la littérature. Parmi les définitions possibles, on peut citer :

- la durée de vie d'un réseau de capteurs est équivalente à la durée jusqu'à ce qu'un certain nombre de noeuds meurt. Par exemple elle est équivalente à :

  - la durée jusqu'à ce que le premier noeud épuise toute son énergie [21] [22] [23] [24] [25],
  - la durée jusqu'à ce qu'il reste au plus une certaine fraction β de noeuds survivants dans le réseau [26] [27] [28],
  - la durée jusqu'à ce que tous les noeuds capteurs épuisent leur énergie, c'est-à-dire $\beta = 0$ [29],
  - la durée jusqu'à ce que 50% des noeuds épuisent leurs batteries et s'arrêtent de fonctionner, c'est-à-dire $\beta = 50\%$ [21],
  - la durée jusqu'à ce que le réseau soit partitionné : apparition de la première division du réseau en deux (ou plus). Cela peut correspondre aussi à la mort du premier noeud (si celui-ci tient une position centrale) [21],
  - la durée pendant laquelle un pourcentage donné de noeuds possède un chemin vers le puits ou coordinateur [30].

- la durée de vie d'un réseau de capteurs peut être aussi liée à la zone de couverture, elle peut être aussi équivalente à la durée jusqu'à ce qu'une partie de la zone reste couverte par un certain nombre de noeud :

  - durée pendant laquelle la zone d'intérêt est couverte par au moins k noeuds [31],
  - espérance de l'intervalle complet pendant lequel la probabilité de garantir simultanément une connectivité et une k-couverture est au moins α [31],
  - durée pendant laquelle chaque cible est couverte par au moins un noeud [32] [33],
  - durée pendant laquelle l'ensemble de la zone est couverte par au moins un noeud [34] [35],
  - durée pour laquelle au moins une portion α de la zone est couverte par au moins un noeud [36] [37] [38],

- durée pendant laquelle la couverture tombe en-dessous d'un seuil prédéfini φ [39],
- durée de fonctionnement continu du système avant que la couverture ou la proportion de paquets reçus (PDR pour Packet Delivery Ratio) tombent en-dessous d'un seuil prédéfini [40] [30],
- durée jusqu'à la perte de la connectivité ou de la couverture totale [41] [42] [43].

- D'autres définitions de la durée de vie d'un réseau de capteurs, sont la :

- durée jusqu'à ce que la première tête de cluster épuise toute son énergie [44] [45],
- durée jusqu'à ce que le réseau ne fournisse plus un taux acceptable de détection d'événements [29],
- durée pendant laquelle le réseau satisfait continuellement les besoins de l'application [46] [47] [48].

Parmis ces différentes définitions, on remarque que certaines ne sont que des relaxations d'autres. Cependant, la plupart suggère qu'une durée de vie de réseau de capteurs est relative à la consommation d'énergie de ses noeuds capteurs.

1.2 Les noeuds des réseaux de capteurs sans fil

Les définitions précédentes nous ont permis d'indiquer que la durée de vie d'un WSN dépend fortement de la durée de vie de chacun des noeuds dans le réseau. Par conséquent, il est très important de considérer la consommation d'énergie dans les noeuds capteurs sans fil. Dans notre étude, afin de bien considérer cette consommation, nous devons tenir compte des différentes activités des noeuds dans le réseau. Ainsi, nous allons étudier, dans la suite de ce chapitre, les caractéristiques générales du matériel et du logiciel dans les noeuds capteurs et nous évoquerons les surconsommations souvent rencontrées dans les WSN.

La prochaine section est dédiée à l'aspect matériel du noeud capteur sans fil. Nous décrivons quelques modules physiques qui ont un impact direct sur la consommation d'énergie du noeud. Puis, nous évoquons les différentes couches protocolaires du noeud capteur sans fil. Nous consacrons la suite à l'aspect logiciel qui commande les modules matériels dans le noeud capteur.

1.2.1 Aspect matériel

A la demande des utilisateurs ou suivant des contraintes liées à l'application, les dimensions des noeuds capteurs sont souvent réduites. Ceci constitue une contrainte technologique importante au regard des éléments matériels plus ou moins nombreux ou imposants à intégrer. L'architecture générale d'un noeud capteur est illustrée sur la figure 1.4 ci-dessous.

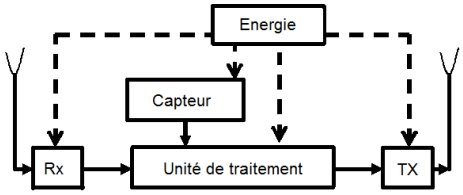

FIGURE 1.4 – Architecture d'un exemple de noeud capteur

Un noeud capteur est autonome en énergie, il possède sa propre source d'énergie (pile ou batterie) [49]. Il est muni d'un dispositif d'acquisition (capteur), une unité de traitement, une unité de communication et éventuellement un système de déplacement. L'unité de traitement est souvent rattachée à une mémoire flash pour pouvoir stocker les données et les traiter avec une consommation d'énergie minimale [50]. Pour s'interconnecter avec les autres noeuds, il dispose d'un module émetteur-récepteur (Tx et Rx sur la figure 1.4 souvent appelé unité de communication.

1.2.1.1 Le module capteur ou dispositif d'acquisition

Un noeud peut inclure plusieurs dispositifs d'acquisition selon l'application. Par exemple, les MEMS (MicroElectroMechanical Systems) sont les plus utilisés pour une grande variété de phénomènes physiques (accélération, concentration chimique, etc). Ces modules d'acquisition procèdent à la capture des données en transformant les grandeurs physiques observées (température, humidité, mouvement, etc.) en données numériques compréhensibles par l'unité de traitement.

Certains types de dispositifs d'acquisition ont des exigences en termes de délai et de débit. Comme exemple, les équipements médicaux de type ECG (électrocardiographie) ou le SpO_2 (Oxymétrie de pouls) ou le thermomètre ont leurs exigences en termes de débit [51].

Le type d'application, la complexité de détection (acquisition ou collecte de données), le type de données à transmettre (vidéos, images, etc) et le mode de délivrance de paquets qu'impose l'application (périodique ou sporadique, continue ou à la de-

mande de l'utilisateur qui peut le commander à distance) ont un impact sur la durée de vie du noeud capteur.

1.2.1.2 L'unité de traitement

L'unité de traitement est le contrôleur principal du noeud capteur sans fil. Le type d'unité de traitement dans un noeud capteur est choisi en fonction de la complexité de ses tâches. Du point de vue matériel, l'unité de traitement peut être de type microcontrôleur, de type microprocesseur, ou encore de type FPGA ou les deux à la fois.

Voici quelques exemples d'unité de traitement parmi celles utilisées ces dernières années dans les WSN.

- MSP430 de Texas Intrument est un microcontrôleur 16 bits réputé pour sa faible consommation. Il est déjà utilisé par de nombreuses plateformes de noeuds capteurs comme Telos, BSN, Wavenis et eyesIFX. Sa consommation varie de 1 à 5 mW [52],
- Atmel ATmega est un microcontrôleur RISC 32 bits et consomme deux fois plus d'énergie que la famillle MSP430x2xx sur tous les modes de fonctionnement,
- Mica est un microcontrôleur CISC 8 bits utilisé dans les études universitaires. Sa consommation dépend de son mode de fonctionnement, elle varie de 0.8 mW sous 1V jusqu'à 77 mW sous 3.5 V [53],
- Intel StrongARM est un processeur RISC 32 bits qui peut atteindre une fréquence de fonctionnement jusqu'à 206 MHz. Ainsi, il est très utilisé pour gérer les applications complexes qui manipulent des flux importants de données [54]. Sa consommation varie de 200 mW à 2W [52].

L'unité de traitement dans les noeuds capteurs est généralement associée à des unités de stockage de type RAM (Random Access Memory), ROM (Read Only Memory), EEPROM (Electricaly-Erasable Programmable Read Only Memory) ou flash. Le choix de ces mémoires (taille et type) varie selon la plateforme utilisée et le type d'application du noeud capteur [53]. Du point de vue logiciel, l'unité de traitement intègre toutes les applications spécialement conçues pour le noeud capteur pour piloter les différents modules matériels.

Cependant, l'unité de traitement est le centre de commande du noeud capteur. Elle a pour rôle de gérer les périphériques, de programmer l'ordonnancement les tâches, de traiter et d'analyser les données, de gérer l'énergie (mise en veille et réveil du noeud

selon la situation). Ainsi, les décisions sur la réception ou l'envoi des messages et des données sont contrôlées au niveau de l'unité de traitement.

1.2.1.3 L'unité de communication

Cette unité est responsable de la connection du noeud capteur au réseau afin qu'il puisse communiquer avec les autres noeuds via un médium sans fil. Elle peut être de type optique, de type ultra sons ou de type radio-fréquence. Le type radio-fréquence comprend des circuits de modulation, démodulation, filtrage, multiplexage, démultiplexage et autres dispositifs électroniques.

Le médium sans fil le plus utilisé dans les WSN est l'onde radio. Voici quelques exemples des normes les plus utilisées :

- Les normes WiFi offre une large bande passante, une portée d'environ 300 mètres et un débit élevé (11 Mbps à Quelques Gbps) sur une bande de fréquence 2.4 GHz ou 5 GHz. Par contre, cette technologie consomme beaucoup d'énergie et la durée de vie des noeuds capteurs sans fil qui l'utilisent est limitée à quelques heures. Cependant, le WiFi n'apparait plus actuellement comme une solution viable pour les projets de WSN ayant de fortes contraintes en énergie sauf en version WiFi ultralow power.

- Le Bluetooth offre aux noeuds capteurs une durée de vie plus longue (quelques jours) que celle du WiFi. Il a une portée d'environ 100 mètres et offre un débit jusqu'à 3 Mbps. Par ailleurs, il est handicapé par la taille limité (8 noeuds, y compris le coordinateur) du réseau qu'il peut former. Néamoins, des travaux sont en cours pour résoudre ce problème du nombre de noeuds limité dans les réseau Bluetooth [55] [56]. Il existe aussi un autre standard dérivé de Bluetooth appelé BLE (Bluetooth Low Energy). Ce dernier permet un débit du même ordre de grandeur pour une consommation dix fois moindre que Bluetooth.

- La 802.15.4 est la solution la plus utilisée dans les WSN puisqu'il a un faible coût énergétique et un faible coût de production des composants. L'utilisation du médium tel que Zigbee prolonge théoriquement la durée de vie d'un noeud capteur sur plusieurs années. De plus, il permet une taille de réseau plus dense (jusqu'à 65536 noeuds) et une portée moyenne de 100 mètres avec un débit maximal de 250 Kbps. Il peut travailler, suivant les pays, sur les bandes de fréquences 868/915 MHz et 2.4 GHz.

Pour économiser de l'énergie, l'unité de communication peut éteindre certains de ses dispositifs selon le degré de mise en veille qu'il souhaite établir [57] [58]. La mise hors tension de dispositifs détermine l'état en cours de l'unité, appelé état interne.

En effet, cette unité a pour rôle de récevoir et de transmettre les données. Ainsi, nous considérons qu'elle peut être divisée en deux modules : l'émetteur (Tx) et le récepteur (Rx) comme le montre la figure 1.4. De manière générale, ces deux modules possèdent seulement quelques états internes. Chaque état correspond à une puissance de consommation. Ainsi, l'état en cours de chaque module caractérise sa puissance de consommation.

Le module récepteur comporte généralement trois états internes : veille, écoute et réception. Pour économiser de l'énergie, le module peut souvent être mis en mode veille, il est alors complètement éteint. Il peut être mis en mode écoute quand il attend des données et/ou écoute le canal de communication. Et quand les données sont en train d'arriver, il est en mode réception.

Le module émetteur comprend également trois états internes : veille, idle et transmission. Il peut prendre le mode veille pour économiser de l'énergie. La plupart de ses dispositifs sont éteints (ne sont pas alimentés) durant celui-ci. L'état idle est un état de transition entre l'état veille et l'état transmission, l'émetteur ne fait rien pendant ce mode idle. Puis, il est mis en mode transmission pour envoyer les données.

La portée d'une transmission d'un noeud capteur sans fil peut aller de quelques mètres à plusieurs centaines de mètres. Elle est contrôlée par plusieurs facteurs clés. Le facteur le plus intuitif est la puissance d'émission. Plus on met d'énergie dans un signal, plus sa portée augmente. D'autres facteurs dans la détermination de la portée incluent la sensibilité au niveau récepteur, le gain, l'efficacité des antennes (émetteur et récepteur), la localisation, les fréquences, la modulation, les conditions météorologiques et le codage de canal, les obstacles et la nature du milieu traversé, le mouvement des noeuds, les interférences, etc.

Bien souvent, dans les réseaux de capteurs sans fil, il n'est pas nécessaire d'avoir des débits élevés, un débit de 10-250 Kbps pour de nombreuses applications est suffisant. Cela permet aussi de préserver l'énergie car le débit a un impact des plus significatifs sur la consommation d'énergie et la durée de vie des noeuds de capteurs.

1.2.2 La source d'énergie

C'est une unité importante dans le noeud capteur. Elle peut avoir deux aspects, essentiellement de :

- Stocker l'énergie (exemple : pile AA normale d'environ 2.2-2.5 Ah fonctionnant à 1.5 V) et la fournir sous la forme requise. Chaque composant dans le noeud sans fil est alimenté par cette source, et sa capacité limitée [59] exige un fonctionnement au rendement optimum pour les tâches effectuées par chaque composant,

- Reconstitution de l'énergie consommée par un réapprovisionnement (grâce à une source externe : cellules solaires, temperature, vibration, etc). C'est une technique de récupération d'énergie par l'énergie thermique, cinétique, etc. Cette technique est utilisée pour des applications où une plus longue vie de réseau est essentielle.

1.2.3 Le système de localisation

Pour faciliter le routage avec les autres noeuds dans le réseau, un noeud a besoin d'un système de localisation. La localisation peut aussi être directement nécessaire à l'application et aux fonctions assurées par le réseau.

Le noeud capteur peut être équipé d'un système de localisation spécifique (GPS pour les réseaux extérieurs). Il peut dans ce cas se localiser de manière autonome. Un noeud peut aussi se localiser au sein de son réseau par rapport à des balises. Dans ce cas, des mesures de distances (cf la partie 3.1) par rapport à ces balises peuvent être opérées (TOA ou Time Of Arrival, TDOA ou Time Difference Of Arrival, ToF ou Time of Flight, etc) et la position est obtenue par des algorithmes de multilatération [60] [61] [62] [63]. Il est aussi possible d'obtenir une estimation de la distance entre le noeud et les balises en effectuant des mesures de puissance (RSSI ou Received Signal Strength Indication) ou de fingerprinting.

1.2.4 La mobilité des noeuds

Pour qu'un noeud puisse se déplacer dans le réseau, si ce dernier n'est pas rattaché à un appareil mobile, il lui faut un système de déplacement. Ce dernier permet de déplacer les noeuds pour accomplir ses tâches. Le système de déplacement peut également opérer en l'intéraction avec l'unité d'acquisition et l'unité de traitement pour contrôler les mouvements du noeud. Selon le degré de cette mobilité, la topologie du réseau change d'une manière très fréquente ; ainsi, connaître ses voisins et reconfigurer le réseau nécessitent un nombre de messages important, donc une dépense plus importante

d'énergie. D'autres types de réseaux de capteurs considèrent à la fois la mobilité des noeuds et du puits. Dans ce cas, la mobilité devient le principal problème.

1.2.5 Les différentes couches protocolaires

On peut aussi définir un noeud capteur en plusieurs couches et en différents plans, comme le montre la figure 1.5. Le but de cette représentation en différentes couches est de pouvoir séparer le problème en différentes parties. D'après le concept historique OSI, chaque couche communique avec ses couches adjacentes (celle du dessus ou celle du dessous). Et chaque couche fournit des services à celle de niveau supérieur et exploite celle de niveau inférieur.

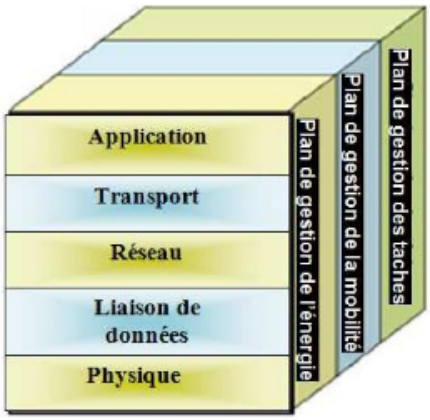

FIGURE 1.5 – Exemple de couches protocolaire d'un noeud capteur [49]

La couche application représente les différentes applications et pilotes : codage, collecte, agrégation, compression, etc. Elle assure l'interface avec les applications. C'est le niveau le plus proche des utilisateurs, géré directement par les logiciels.

La couche transport est chargée du transport des données, contribue au maintien du flux si c'est nécessaire (contrôle de flux, gestion des éventuelles erreurs de transmission non corrigée au niveau 2, etc), au découpage des données en paquets, ainsi que la conservation de l'ordre des paquets.

La couche réseau s'occupe de l'acheminement des données fournies par la couche transport : routage, relayage, etc. Le protocole de routage est le principale acteur dans cette couche réseau, il permet de trouver une route et une transmission performante des données. Plusieurs nouveaux algorithmes de routage ont été conçus pour les problèmes de routage dans les réseaux capteurs (optimisation de l'utilisation de l'énergie des capteurs, gestion des trafics, etc).

La couche liaison de données spécifie comment les données sont expédiées entre deux noeuds dans une distance d'un saut. Cette couche s'occupe du multiplexage des données, elle s'occupe aussi de la méthode d'accès au canal, du contrôle de puissance, de la retransmission, du contrôle d'erreurs. Cependant, elle assure la liaison point à point et multi-point dans un réseau de communication. Le protocole MAC dans cette couche liaison doit tenir compte de la consommation d'énergie et doit être en mesure de réduire les collisions.

La couche physique est relative à la technologie ou l'architecture utilisée, la modulation, la technique de transmission et de réception, le contôle de puissance, le codage, le filtrage, etc.

On trouve aussi parfois des plans verticaux illustrés sur la figure 1.5.

Les plans de gestion d'énergie, de mobilité et des tâches permettent de contrôler la consommation d'énergie, les mouvements, la répartition des tâches entre les noeuds capteurs. Ces plans permettent aux noeuds capteurs de coordonner les tâches et de réduire l'ensemble de la consommation d'énergie. Ces plans ont un rôle important pour la gestion des noeuds, pour que les noeuds puissent collaborer, pour l'acheminement des données à travers un réseau mobile et pour le partage des ressources.

Le plan de gestion d'énergie permet la gestion de l'utilisation efficace de l'énergie au niveau du noeud. Par exemple, il permet la mise en veille du module récepteur dès que le noeud a fini de recevoir un message (économiser l'énergie restante). Quand un noeud possède un niveau d'énergie faible, il permet de diffuser un message aux autres capteurs pour ne pas participer aux tâches de routage, et conserver l'énergie restante aux fonctionnalités d'acquisition.

Le plan de gestion de mobilité a pour rôle de connaître et d'enregistrer les mouvements du noeud capteurs de manière à lui permettre de garder continuellement une route vers l'utilisateur final. Cela permet de faciliter le choix du chemin optimal de donnée puisqu'il a la responsabilité de maintenir une image récente sur la position des noeuds voisins [64]. Cette image récente des noeuds voisins est également utile pour le plan de gestion de tâche.

Le plan de gestion de tâche applicatives collabore avec le plan de gestion d'énergie. C'est lui qui permet l'équilibre et l'ordonnancement des différentes tâches d'acquisition faites par les noeuds dans la zone de captage ciblée. Il n'est pas nécessaire que tous les noeuds dans cette région effectuent la tâche d'acquisition simultanement, certains noeuds peuvent la faire plus que d'autres suivant le niveau de batterie.

1.3 Logiciels et protocoles réseau

En plus des plateformes matérielles et des standards, plusieurs plateformes logicielles ont été, également, développées spécifiquement pour les réseaux de capteurs sans fil.

1.3.1 Les systèmes d'exploitation

Il existe plusieurs systèmes d'exploitation conçus pour les noeuds capteurs sans fil. A cause du nombre élevé d'applications sur les WSN et les différentes plateformes matérielles disponibles, un système d'exploitation semble nécessaire pour des types d'applications complexes sur les WSN. Il a pour mission de jouer l'intermédiaire entre l'utilisateur et les périphériques matériels.

Les rôles du système d'exploitation sont d'assurer les services basiques suivant : l'abstraction de la partie matérielle, la gestion des tâches, la gestion de la mémoire, la gestion de puissance et la gestion des périphériques. Plusieurs systèmes d'exploitation sont dévéloppés pour les réseaux de capteurs sans fil. Les plus utilisés depuis une petite dizaine d'années sont TinyOS et Contiki.

TinyOS [65] est un système d'exploitation open-source spécialement développé pour les applications embarquées fonctionnant en réseaux et, en particulier, pour les réseaux de capteurs sans fil. En effet, on peut trouver celui-ci dans certains noeuds disponibles actuellement et de nombreux groupes de recherche ainsi que des industriels s'en servent afin de développer et tester des protocoles, différents algorithmes, des scénarios de déploiement.

Cette popularité est due au fait que TinyOS fournit non seulement des outils de développement et de simulation aux développeurs mais également une solution pour développer rapidement des applications qui correspondent aux caractéristiques d'un réseau, telles que le besoin de fiabilité dans les informations récoltées, la qualité de service du réseau ou encore aux capteurs mis en jeu. De plus, le domaine de l'embarqué impose de sévères contraintes concernant l'espace de stockage et l'espace mémoire alloués au système d'exploitation et aux applications. TinyOS résout ce problème en générant une très petite empreinte mémoire. TinyOS incorpore une architecture à base de composants, qui réduit au minimum le nombre d'instructions et fournit une plateforme flexible pour mettre en application de nouveaux protocoles de communication. Sa bibliothèque de composants inclut des protocoles réseau, des services distribués, des pilotes logiciels pour les capteurs et des outils pour l'acquisition de données ; qui peuvent être modifiés ou améliorés sur la base des conditions spécifiques d'application.

Enfin, en ce qui concerne plus particulièrement le domaine des réseaux de capteurs, les capteurs sont souvent à l'origine de la détection d'un phénomène et renseignent le réseau de son existence. TinyOS suggère ainsi un modèle de programmation orienté évènement.

D'autres plateformes logiciel et systèmes d'exploitation ont été dévéloppés pour les réseaux de capteurs sans fil. LiteOS [66] est un système d'exploitation multi-thread et assure la gestion de mémoire dynamique et un interpreteur de commandes interactifs. Contiki [67] est un autre système d'exploitation open source multitâche développé pour une série de plates-formes comprenant des microcontrôleurs tels que le TI MSP430 et Atmel AVR, qui sont utilisés dans les familles de Telos, de Tmote et de Mica. Contiki a été construit autour d'un noyau "event-driven" mais il est possible d'utiliser le multithread pour certains programmes aussi bien que le chargement et le remplacement dynamiques de différents programmes et services. Comparé à TinyOS, qui est statiquement lié à la version compilée, Contiki permet de substituer des programmes et des pilotes (drivers) en temps réel et sans recompilation.

1.3.2 Les protocoles de routage

Dans les réseaux de capteurs, l'infrastructure peut ne pas être fixe. Dans certaines topologies (maillée, arbre, etc), la participation au routage concerne chaque noeud pour transmettre les informations de la source à un noeud de destination bien précis, par des sauts multiples. Les protocoles de routages sont des applications implémentées dans les noeuds capteurs pour définir le comportement de ces derniers lors des calculs du chemin optimal, du routage de données.

Il existe plusieurs techniques de routage, selon la topologie du réseau et le modèle de trafic.

1.3.2.1 Les protocoles de routage non hiérarchiques

- Les protocoles de routages proactifs :
Ils sont globalement basés sur le même principe de routage que dans les réseaux filaires. Ces types de protocoles sont indépendants du modèle de trafic. Dans ce type de routage, les routes sont définies avant la transmission de données, suivant l'état des liens (distances, obstacles, nombre de sauts, énergie restante des noeuds destinataires, etc) entre le noeud émetteur et les autres noeuds. En cas de problème de pertes de données ou erreurs de transmission de bout en bout, on a recourt à la technique de retransmission. Chaque noeud met à jour sa table de routage en échangeant des paquets

de contrôle avec les noeuds voisins. En effet, si un noeud veut transmettre des données, il a la possibilité de consulter localement la table de routage et de créer le chemin dont il a besoin. La nécessité de conserver et de contrôler la validité des tables de routages en permanence (comprenant en outre des informations qui ne seront sans doute pas utilisées) est l'un des principaux inconvénients des protocoles proactifs. De plus, si la topologie change, il faut mettre à jour la table de routage toute entière. Par contre, ils présentent l'important avantage de n'introduire aucun délai avant de transmettre un paquet puisque la route est déjà définie.

Exemple : OLSR (Optimized Link State Routing) [68] [69]. Comme son nom l'indique, OLSR est un protocole à état de lien optimisé. C'est un protocole proactif spécialement adapté aux réseaux grande échelle. Tous les noeuds du réseaux peuvent jouer le rôle de relais et OLSR maintient sur chaque noeud une table de routage complète (comprenant une entrée pour tous les autres noeuds du réseau). Chaque noeud choisit la route la plus adaptée en fonction des informations qu'il a reçues. Dans un protocole à état de lien, chaque noeud déclare ses liens directs avec ses noeuds voisins à tout le réseau. En utilisant OLSR, les noeuds ne déclarent qu'une sous partie de leur voisinage. Les routes sont construites à base de relais multipoints qui sont l'ensemble des noeuds voisins (MPRs). Par ailleurs, l'interêt des relais multipoints est de minimiser le trafic dû à la diffusion des paquets de contrôle dans le réseau.

Pour maintenir à jour toutes les informations nécessaires au choix des relais multipoints et au calcul de la table de routage, l'échange périodique des informations est nécessaire entre les noeuds utilisant OLSR. Chaque noeud diffuse périodiquement un message HELLO contenant des informations sur son voisinage et l'état des liens. Le paquet TC (Topology Control) est le deuxième paquet d'OLSR qui lui aussi est envoyé périodiquement pour compléter la table de routage. Un paquet TC contient la liste des voisins du noeud l'ayant choisi comme MPR. Ce message TC est diffusé sur tout le réseau mais seuls les voisins MPR rediffusent un paquet TC pour éviter la redondance.

- Les protocoles de routage réactifs :

Contrairement aux protocoles proactifs, les protocoles réactifs ne calculent la route que sur demande. Ils tiennent compte de l'évolution des trafics et beaucoup d'autres paramètres dans le réseau afin de déduire le chemin optimal pour transmettre l'information d'un noeud à un autre. Ces protocoles sont surtout utilisés dans le cas des noeuds mobiles et quand les liens entre les noeuds varient progressivement. Si un noeud source a besoin d'envoyer un message à un noeud destination, alors il envoie une requête à tous les membres de réseau. Après la réception de la requête, le noeud destination envoie un message réponse qui remonte vers la source (méthode Backward Learning [70]). Ce-

pendant, le routage à la demande génère une lenteur à cause de la recherche des routes. Cela peut entraîner une dégradation des performances des applications. Ce type de protocole présente l'inconvénient d'être très coûteux en transmission de paquets lors de la détermination des routes. Par conséquent, il n'y a pas une route prédéfinie, on ne maintient la route que si nécessaire. Ces protocoles ont alors l'avantage de ne pas avoir à maintenir des informations inutilisées dans les tables de routage. Exemple : AODV (Ad Hoc On-Demand Distance Vector Routing), DSR (Dynamic Source Routing), etc.

AODV [70] est un protocole à vecteur de distance. L'objectif du protocole AODV est de fournir un service complètement orienté sur le principe de la route à la demande, puisque qu'il ne demande une route que lorsqu'il en a besoin. Les noeuds ne maintiennent pas d'information de routage et n'échangent pas périodiquement leur table de routage. Par ailleurs, un noeud doit d'abord établir une demande de connexion avant de découvrir et de maintenir les routes. Les paquets de découverte ne sont transmis seulement que lorsque cela est nécessaire. AODV permet de différencier la gestion des connexions locales et la gestion de la topologie générale. Ce protocole permet aussi de diffuser les changements des connexions locales aux noeuds mobiles qui ont probablement besoin de cette information. L'avantage de ce protocole reside sur son mécanisme de découverte de route basé sur un établissement dynamique des routes par les noeuds intermédiaires. Ce système est efficace pour des réseaux à grande échelle. Ce protocole assure alors une utilisation efficace de la bande passante, une réactivité aux changements de topologie et évite les boucles dans le réseau.

1.3.2.2 Les protocoles de routages hiérarchiques

La structuration d'un réseau est une technique pour sauvegarder l'énergie dans chaque noeud du réseau. Et cette sauvegarde de l'énergie aboutit au prolongement de la vie du système. Une des structures les plus connues est la hiérarchie. La technique de hiérarchisation sert à partitionner le réseau en sous ensembles afin de faciliter la gestion du réseau surtout le routage, qui se réalise à plusieurs niveaux. Dans ce type de protocoles, la vue du réseau devient locale ; des noeuds spéciaux peuvent avoir des rôles supplémentaires.

Après avoir défini la route ou le prochain noeud de destination vers lequel on doit véhiculer l'information, le rôle de la couche MAC est de définir la technique d'accès au canal de communication afin de connecter le noeud émetteur et le noeud récepteur.

1.3.3 Les techniques d'accès au canal de communication

La plupart des travaux sur l'optimisation de la consommation d'énergie sont convaincus que le module radio semble être le plus consommateur en énergie dans les noeuds capteurs sans fil. La grande partie de l'activité radio est commandée par la couche MAC. La perte d'énergie peut avoir plusieurs causes. Parmi les principales, on peut citer les collisions de données, la réception de données destinées à un autre noeud, la transmission de données vers un noeud récepteur non prêt à recevoir, hors de portée radio, etc.

Donc, le protocole MAC joue un rôle important pour économiser l'énergie dans la partie radio. Il faut alors que le protocole MAC gère le transmetteur, ce dernier doit être éteint le plus tôt et le plus longtemps possible avant et après une réception ou une transmission de données. Ce mécanisme doit être en phase avec la synchronisation et la répartition des périodes de réveil entre les noeuds capteurs.

Le protocole MAC doit alors avoir certaines caractéristiques : une efficacité en termes d'énergie, une stabilité sur l'évolution du réseau (topologie, taille), une stabilité sur la dynamicité du réseau (apparition et disparition d'un ou plusieurs noeuds).

Le protocole MAC spécifie la technique d'accès au canal. Il existe principalement deux modes de fonctionnement de la couche MAC : le mode "non beacon" utilisant souvent le CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) et le mode "beacon" utilisant une balise pour synchroniser les dispositifs.

Les protocoles MAC synchronisés : la synchronisation est utilisée par certains protocoles MAC pour coordonner les phases de réveil des noeuds afin qu'ils se réveillent simultanément et ainsi pour qu'ils puissent faire la transmission de données sans problème de connectivité tout en restant économiques en consommation d'énergie. Plusieurs protocoles utilisant cette synchronisation ont été dévéloppés.

Le protocole S-MAC (Sensor MAC) [71] est l'un de ces protocoles synchronisés spécialement conçu pour les réseaux de capteurs. Tous les noeuds dans le réseau partagent le même calendrier de réveil afin d'éviter le problème de surdité (transmission vers un noeud non prêt à recevoir). Le S-MAC gère la période de réveil de tous les noeuds avec deux modes possibles.

- Mode active (radio allumée) : les noeuds sont réveillés pour se synchroniser d'abord en échangeant des paquets de synchronisation "SYNC", puis la transmission des trames de données.

- Mode inactive (Radio en veille) : Les noeuds mettent en veille leur module radio pour économiser de l'énergie.

Le protocole S-MAC utilise le mécanisme CSMA/CA et les paquets de contrôle RTS/CTS pour éviter le problème de stations cachées et pour éviter les collisions.

Plusieurs autres variétés du protocole MAC ont été dévéloppées. Tel que le D-SMAC (Dynamic S-MAC) [72] qui est aussi une extension du protocole S-MAC permettant aux noeuds de rajouter des périodes d'activités supplémentaires et ainsi d'augmenter leur cycle de réveil. Plusieurs autres extensions de S-MAC ont été dévéloppées, par exemple dans [73], afin de prendre en charge la mobilité des noeuds dans le réseau. Les périodes d'activité d'un noeud sont adaptées à la mobilité de ses noeuds voisins en se basant sur la variation de la puissance du signal des paquets SYNC reçus. Le protocole T-MAC (Timeout MAC) [74] est une variante de S-MAC.

Le protocole D-MAC [75] a été dévéloppé afin de pallier le problème de latence engendré par l'utilisation des protocoles S-MAC et T-MAC, tout en ayant toujours comme objectif la diminution de la consommation énergétique. Ce protocole a principalement été conçu pour des applications avec un trafic de type "convergecast", c'est-à-dire un trafic qui converge vers une destination bien précise (le puits). Il est sutout utilisé dans les réseaux de topologie arbre ou cluster-tree.

Les protocoles MAC à préambule : B-MAC (Berkeley MAC) [76], WiseMAC [77], etc. Avec ce type de protocole, les noeuds choisissent eux même leur temps de réveil indépendament de leur voisinage, mais avec des périodes inter-réveil constantes. Les noeuds capteurs sont en mode veille durant la majeur partie de leur cycle et se réveillent périodiquement pour un petit intervalle de temps pour vérifier l'état du canal de commmunication. Une décision sera prise, pour rester éveillé ou aller se mettre en veille, selon l'état du canal. Les noeuds ont alors des périodes de réveil asynchrones, cependant pour assurer la communication, chaque noeud doit émettre avant la trame de données un préambule d'une longueur assez grande pour couvrir au moins un cycle complet (réveil/sommeil) et ceci dans le but de s'assurer que le récepteur puisse se réveiller à temps, entendre le préambule et ainsi recevoir la trame envoyée.

Ce genre de protocoles présente certains inconvénients. D'une part un surcoût énergétique dû à l'émission du préambule et d'autre part l'obligation des noeuds, en attendant un préambule, de rester éveillé jusqu'à la fin de ce dernier pour vérifier si la trame allant être émise leur est adressée. Ainsi, plusieurs protocoles ont été suggérés pour remédier à ce problème de surcoût énergétique imposé par le préambule, tout en optimisant la taille de ce dernier pour réduire l'attente des noeuds avant la réception des paquets.

D'autres protocoles tels que dans [78] et X-MAC [79] ont été proposés afin de réduire la consommation énergétique générée par l'envoi et la réception du préambule.

Une technique a été utilisée pour que les noeuds n'étant pas concernés par la transmission puissent rapidement éteindre leur module radio sans attendre la fin du préambule. En utilisant cette technique, la latence sera également réduite grâce à la possibilité pour les noeuds d'interrompre l'émission du préambule en envoyant un acquittement pour signaler à l'initiateur qu'ils sont prêts à recevoir les données.

1.4 Discussion

On rencontre souvent une consommation inutile, un gaspillage d'énergie au niveau du module radio. Les sources de cette surconsommation sont nombreuses surtout au niveau MAC où se déroule le contrôle d'accès au médium sans fil.

- L'écoute passive : dans les techniques d'accès au support sans fil classique (exemple : 802.11 DCF), le module radio est prêt à recevoir en permanence. Ceci est coûteux et inutile dans le cas des réseaux à faible charge de trafic. Ce mode prêt à recevoir consomme beaucoup d'énergie alors qu'il n'y a pratiquement pas beaucoup d'informations qui circulent dans le canal de communication. Ce problème est fréquent dans les réseaux ad hoc de capteurs où le canal est libre la plupart du temps. D'où le gaspillage d'énergie et que la solution serait d'éteindre la radio. Il faut cependant tenir compte des phases de transitions, l'énergie consommée pendant ces transitions et la fréquence de ces transitions doit rester raisonnable.

- L'écoute abusive (ou overhearing) : cette situation se présente quand un noeud reçoit des paquets inutiles.

- Réception et décodage d'une trame destinée à un autre noeud

- Réception et décodage de trames rédondantes dont le contenu est sans interêt pour le routage ou pour l'application car il a déjà été reçu par le noeud considéré.

Le coût de l'écoute abusive peut être un facteur dominant de la perte d'énergie quand la charge de trafic est élevée et en particulier quand la densité des noeuds est grande.

- Les collisions : elles sont la première source de perte d'énergie. Quand deux ou plusieurs trames sont émises et se heurtent, elles deviennent inexploitables et doivent être abandonnées. Les retransmettre par la suite, consomme de l'énergie. C'est le rôle des protocoles MAC de les éviter, en particulier les protocoles MAC avec contention.

- Le surcoût des paquets de contrôle : quelques techniques d'accès au support utilisent des paquets de contrôles (exemple : RTS, CTS) pour l'évitement des collisions. L'envoi, la réception et l'écoute de ces paquets de contrôle consomment de l'énergie.

Comme les paquets de contrôles ne transportent pas directement de l'information pour les applications, ils réduisent également le débit utile effectif.

L'élaboration d'une couche "liaison de donnée" efficace adaptée aux réseaux de capteurs est donc primordiale pour réduire la consommation d'énergie et augmenter la durée de vie du réseau puisque le taux d'erreur ou BER (Bit Error Rate) est aussi un facteur important de l'augmentation de la consommation.

Afin de traiter ces différents aspects de surconsommation, de nombreux outils de modélisation et de simulation (NS2 ou Network Simulator 2, OMNET ou Objective Modular Network Tesbed, etc) sont apparus ces dernières années.

1.5 Conclusions

En résumé, ce chapitre nous a permis de voir le fonctionnement général d'un noeud capteur sans fil. Ce dernier est composé de différents modules matériels (unité de traitement, unité de captage, unité de transmission et l'énergie) et des logiciels qui les commandent. Un noeud capteur sans fil peut être aussi representé par des couches protocolaires adjacentes. Chaque couche a ses propres rôles et collabore avec ses couches adjacentes. Ainsi, l'optimisation que l'on veut apporter au niveau de chaque couche a une influence très importante sur le choix des mécanismes et des protocoles que l'on veut implémenter dans le noeud capteur. Ce choix doit prendre en compte la consommation d'énergie et dépend également du type de topologie du réseau et de l'application du noeud capteur sans fil.

2

Simulation de la consommation d'énergie des WSN

2.1 Introduction

Les modèles sont les premières étapes de la concrétisation d'une nouvelle idée ou d'une nouvelle approche. Les modèles créés sont souvent simulés afin d'estimer leur validité en les comparant avec d'autres modèles similaires. Les simulateurs sont alors d'une grande utilité dans la recherche scientifique et dans le développement d'applications. Ils permettent de simuler des modèles dans un environnement virtuel dans le but d'anticiper les résultats non satisfaisants et de faciliter la modification du modèle pour corriger les problèmes avancés par les résultats de simulation. Ils permettent aussi l'aide à la décision avant de choisir ou de définir les matériels qui seront utilisés. Ils permettent également d'analyser les situations dangereuses ou difficilement reproductibles.

L'objectif de ce chapitre est de présenter quelques modèles et outils de simulation qui ont été développés pour les WSN. Nous décrivons leurs caractéristiques et nous établissons un classement en fonction du niveau d'abstraction auquel ils permettent de travailler.

La première section de ce chapitre est consacrée à la description des différents niveaux d'abstraction pour modéliser un système. Dans la deuxième partie, nous dé-

crivons quelques travaux sur la modélisation des noeuds et des réseaux de capteurs sans fil. Ensuite, nous évoquons les caractéristiques de quelques simulateurs de WSN. Une étude comparative de ces simulateurs, ainsi que d'autres simulateurs, est alors effectuée. Cette analyse est aussi présentée comme une cartographie des simulateurs suivant le niveau d'abstraction et le stade de conception du noeud capteur auquel ils sont dédiés. Enfin nous indiquerons, en fin de chapitre, que l'état de l'art actuel sur les modèles de noeuds et les simulateurs ne permet pas de répondre à tous les besoins en terme de méthodologie de conception.

2.2 Introduction sur la modélisation des noeuds capteurs

Les travaux de modélisations qui nous intéressent sont alors liées à la consommation d'énergie du noeud et à la consommation d'énergie générale du réseau de capteurs. Le dysfonctionnement de quelques noeuds nécessite une modification de la topologie du réseau et un changement de routage des paquets, etc. Autrement dit, tout cela a un impact important sur toutes les couches. Toutes ces opérations de modifications sont gourmandes en énergie, c'est la raison pour laquelle la plupart des travaux se concentrent principalement sur l'optimisation de la consommation d'énergie au niveau des noeuds et du réseau tout entier.

La modélisation d'un noeud s'avère quasi-incontournable pour optimiser sa consommation d'énergie afin de prolonger le plus longtemps possible sa durée de vie. En effet, un réseau de capteurs doit prendre en compte la consommation d'énergie de chaque noeud pour éviter l'extinction et l'isolement des noeuds et pour prolonger la durée de vie du réseau entier.

2.2.1 Les niveaux d'abstraction

La modélisation d'un même noeud ou d'un même réseau peut être différente selon le niveau d'abstraction considéré. Tout d'abord, une abstraction implique souvent une simplification et un remplacement d'une architecture complexe et détaillée du dispositif par un modèle compréhensible grâce auquel nous pouvons résoudre un problème. Ainsi, le niveau d'abstraction est un critère de classification de concepts. En effet, un des objectifs de l'abstraction est d'éliminer les détails dont l'impact sur la solution d'un problème n'est pas importante ou inexistante, nous conduisant à un modèle qui

nous permet de nous consacrer à l'essence du problème. Ainsi, il peut y avoir plusieurs niveaux d'abstraction et le niveau d'abstraction le plus élevé est le concept d'un modèle le plus facilement compréhensible par l'utilisateur. Ainsi, plus on monte en abstraction, plus la vision du modèle est générale et ne tient pas compte de certains détails. Dans ce cas, le modèle est plus facilement manipulable. En contre partie, les résultats produits sont parfois moins précis.

2.2.2 Les niveaux d'abstraction et la conception d'un système

Généralement, le cycle de conception d'un système est divisé en plusieurs étapes. Le concepteur utilise en général différentes variétés de niveaux d'abstraction pour modéliser son système en fonction de la précision des informations qu'il souhaite valider. Ensuite, le concepteur descend d'un niveau d'abstraction vers un niveau plus détaillé en rendant le modèle plus subtile et plus sophistiqué. La figure 2.1 montre les différents niveaux d'abstraction d'un modèle de système (du système embarqué tout entier) [80].

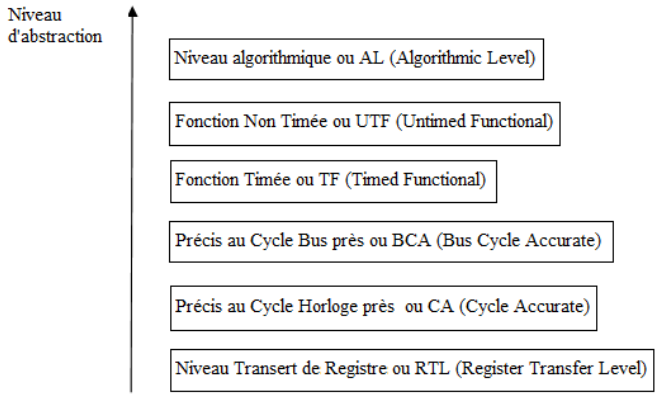

FIGURE 2.1 – Niveau d'abstraction

Le niveau AL (Algorithmic Level) est le niveau d'abstraction le plus élevé. C'est une représentation fonctionnelle du système qui peut être "mappée" sur une architecture. Il existe un certain nombre de langages utilisables à ce niveau d'abstraction, qui dépendent généralement du domaine d'application, tel que UML, le langage utilisé pour l'outil Matlab, etc.

Le niveau UTF (Untimed Functional) est employé pour décrire la vue fonctionnelle d'un système. A ce niveau d'abstraction, un système est décomposé en un ensemble de modules possédant un ou plusieurs processus, et communiquant entre eux par l'intermédiaire de mécanismes de communication de haut niveau tels que les FIFO (First In, First Out). A ce niveau de description, les modèles sont non-timés, c'est-à-dire que

les algorithmes des fonctions et les opérations de transport de données s'exécutent en temps nul.

Le niveau TF (Timed Functional) est une extension du niveau UTF mais dans laquelle la modélisation du temps a été rajoutée. A ce niveau de détail, les durées d'exécution des algorithmes des fonctions sont définies de manière approximative. De même, les durées d'exécution des opérations de transport de données s'avèrent être fixes ou variables. Celles-ci peuvent, par exemple, dépendre de la taille des messages.

Le niveau BCA (Bus Cycle Accurate) est destiné à décrire la communication entre les systèmes au cycle d'horloge prêt. Le passage du niveau TF au niveau BCA est marqué par la synthèse des communications. A ce niveau, la cohabitation des modules de niveau UTF et TF est possible.

Le niveau CA (Cycle Accurate) est obtenu par synthèse comportementale de chaque module matériel du modèle BCA. Ce niveau élabore tous les détails de la microarchitecture et représente typiquement des modèles d'interfaces au niveau du bit. Toute l'infrastructure liée au protocole de communication est modélisée. Les modèles de niveau CA sont précis au niveau du cycle d'horloge et font abstraction de l'activité à l'intérieur du cycle d'horloge. Ceci va permettre d'obtenir des résultats de simulation plus rapidement qu'au niveau RTL. Le passage au niveau RTL est nécessaire lorsque l'on souhaite simuler le détail de tous les signaux à l'intérieur d'un cycle d'horloge.

Enfin, le niveau d'abstraction RTL (Register Transfer Level), utilisé pour la conception de plates-formes, constitue le niveau d'implantation du matériel. Le niveau RTL est le premier à avoir été défini en tant que niveau industriel standard pour la conception des ASIC (Application Specific Integrated Circuit), pour l'échange d'IP (Intellectual Propriety) matérielles, et en tant que plus haut niveau d'interface pour la conception des puces. Les langages utilisés à ce niveau sont typiquement le VHDL et le Verilog.

Plusieurs simulateurs sont spécialement conçus pour simuler différents types de modèles, que ce soit un modèle de noeud capteur ou un modèle de réseau de capteurs. Chaque simulateur de réseau ou de noeud capteur a son propre objectif, mais tous les simulateurs visent à respecter les même normes (mediums radio, protocoles, etc).

2.3 La modélisation dans le domaine des WSN

La modélisation d'un même noeud capteur peut être différente suivant l'objectif assigné à la simulation. Par exemple, si on veut optimiser la consommation d'énergie dans un noeud capteur ou dans un réseau global, il peut y avoir un modèle de noeud capteur ou un modèle de réseau de capteurs. Par conséquent, un noeud peut être modélisé à

plusieurs niveaux d'abstraction suivant l'objectif de la modélisation mais aussi suivant les informations disponibles pendant l'étape de conception pour lequel le modèle sera utilisé.

La plupart des travaux de modélisation sont souvent orientés vers l'optimisation d'une fonction bien précise d'un noeud capteur ou sur un aspect spécifique, comme la stratégie de communication [57] [81] [82], les techniques de routages [83] [84] [64], les différentes topologies des réseaux [85] [18] [17] ou encore les dispositifs matériels [86] [87].

Beaucoup de travaux considèrent le noeud capteur à un niveau d'abstraction très bas, en examinant la partie matérielle du noeud. Dans ce cas, les travaux se concentrent sur la plateforme matérielle et étudient des matériels spécifiques qui composent le noeud capteur. Ils explorent plus précisément l'architecture et le fonctionnement de chaque bloc du noeud capteur [88] [89]. La plupart de ces travaux [58] [90] de bas niveau d'abstraction concernent surtout le module radio qui semble être le plus consommateur en énergie dans le noeud capteur.

Travailler à un bas niveau d'abstraction permet d'avoir une estimation de la consommation d'énergie plus précise et plus proche de la réalité. Mais, on risque de ne pas pouvoir séparer l'estimation d'énergie sur les différentes opérations supportées par la même plateforme matérielle et la simulation peut prendre plus de temps suivant la complexité de l'application.

Le niveau d'abstraction le plus bas trouvé, parmi les travaux de modélisation déjà existants, est la modélisation du noeud capteur au niveau composant électronique.

2.3.1 Modélisation au niveau bloc matériel

Modéliser à un très bas niveau d'abstraction comme dans [91] consiste à modéliser les différents dispositifs électroniques qui composent le noeud capteur. L'objectif est d'avoir une estimation plus détaillée et précise, proche de la réalité de la consommation du noeud capteur.

Le module le plus souvent modélisé est le module radio puisqu'il semble être le plus consommateur en énergie dans le noeud. La modélisation à bas niveau d'abstraction considère les différents composants électroniques afin d'estimer le plus fidèlement possible la consommation d'énergie ou de trouver une éventuelle optimisation sur la consommation d'énergie.

Voici l'exemple d'un module radio dans [91], avec le modèle de transmission présenté dans la figure 2.2.

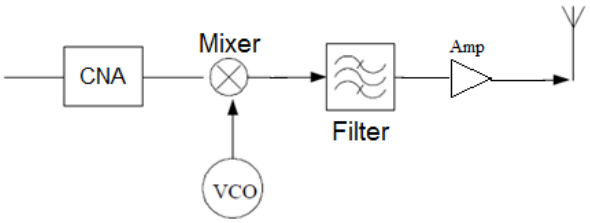

FIGURE 2.2 – Exemple de module d'émission

Le modèle de puissance d'émission est :

$$P_{tx} = P_{CNA} + P_{vco} + P_{mixer} + P_{filt} + P_{Amp} \tag{2.1}$$

avec,

P_{CNA} : Puissance consommée par le convertisseur numérique analogique

P_{filt} : Puissance consommée au niveau du filtre

P_{vco} : Puissance consommée par l'oscillateur commandé en tension

P_{mixer} : Puissance consommée par le mélangeur

P_{Amp} : Puissance d'amplification

Exemple de modèle de réception présenté par la figure 2.3.

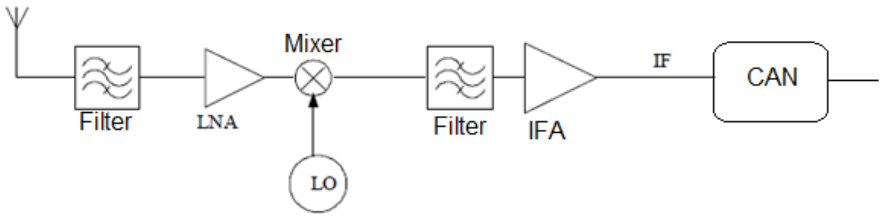

FIGURE 2.3 – Exemple de module de réception

Et le modèle de puissance de réception est :

$$P_{rx} = P_{LO} + P_{mixer} + 2P_{filt} + P_{lna} + P_{IFA} + P_{CAN} \tag{2.2}$$

avec,

P_{lna} : Puissance consommée par l'amplificateur à faible bruit

P_{IFA} : Puissance consommée par l'amplificateur à fréquence intermédiaire

P_{CAN} : Puissance consommée par le convertisseur analogique numérique

Dans ce cas, la modélisation concerne les composants électroniques qui caractérisent la consommation d'énergie dans le bloc émetteur-récepteur.

Il existe aussi une autre façon de modéliser un noeud capteur à ce même niveau d'abstraction. La modélisation suivante ne s'intéresse pas aux détails technologiques dans le noeud mais a pour objectif d'avoir un modèle de système plus globale. Dans ce cas, on considère la quantité d'information (taille de données), les transitions entre les états, la fréquence de transition, etc.

2.3.1.1 Dans le module radio

Au niveau d'abstraction bloc matériel, l'énergie consommée dans le bloc radio dépend des types de dispositifs implémentés dans le module radio, la taille de données à transmettre et la qualité du canal de commumnication. C'est alors une modélisation de l'énergie du noeud liée à une estimation de la quantité d'information.

Par exemple, dans [57] [81] [92] [93], l'énergie consommée pendant la réception des données est caractérisée par cette équation :

$$E_{rx}(k) = E_{elec}k \tag{2.3}$$

E_{elec} : dissipation d'énergie par bit de l'émetteur en J/bit (typiquement $50nJ/bit$)
k : taille de données en bit

On peut noter par cette formule que l'énergie consommée au niveau de la réception dépend de la taille des données reçues, mais aussi du type de dispositif utilisé par le module radio.

L'énergie consommée pendant la transmission des données [57] [81] [92] [93] est illustrée par cette équation :

$$\begin{aligned} E_{tx}(k,d) &= E_{elec}k + E_{amp}k \\ &= E_{elec}k + \xi_{fs}d^2k \quad d_o > d \\ &= E_{elec}k + \xi_{amp}d^4k \quad d_o \leq d \end{aligned} \tag{2.4}$$

ξ_{amp} : dissipation de l'antenne en "two-ray" propagation (typiquement $0.0015pJ/bit/m^4$)
ξ_{fs} : dissipation de l'antenne en "Friss free space" (typiquement $10pJ/bit/m^2$)
d_o : la distance seuil pour choisir entre le modèle "Friss free space" ou le modèle "two-

ray" propagation

Si la distance entre le noeud émetteur et le noeud récepteur est plus petite qu'une certaine distance d_o (valeur de référence fixe), alors on utilise le modèle "Friss free space" [57] [81] (avec une attenuation en d^2). Dans le cas contraire, on utilise le modèle "two-ray ground propagation" [92] [93] (attenuation en d^4).

L'énergie de transmission nécessaire dépend alors de certains paramètres, comme la distance entre l'émetteur et le récepteur et la taille des données à transmettre. Cependant, les atténuations et les réflexions ne sont pas pris en compte dans ces modèles.

2.3.1.2 Dans le module acquisition ou capteur

La consommation d'énergie dans le dispositif d'acquisition est divisée en plusieurs étapes [94] : activation de la sonde de capture, échantillonnage, conversion analogique/numérique et enregistrement des données dans la mémoire.

$$E_{capture}(b) = bV_{alim}I_{capt}T_{capt} \tag{2.5}$$

$$E_{enregistrement}(b) = E_{criture} + E_{lecture} = \frac{bV_{alim}}{8}(I_{criture}T_{criture} + I_{lecture}T_{lecture}) \tag{2.6}$$

avec,
V_{alim} : tension d'alimentation
I_{capt} : intensité du courant nécessaire pour la capture de données
T_{capt} : durée d'acquisition
$I_{criture}$: intensité nécessaire pour l'écriture d'un octet dans la mémoire
$T_{criture}$: durée d'écriture
$I_{lecture}$: intensité nécessaire pour la lecture d'un octet dans la mémoire
$T_{lecture}$: durée de lecture de données dans la mémoire
b : taille du paquet en bit

La sonde fournit des signaux électriques, basés sur le phénomène observé, au convertisseur analogique-numérique. Ce dernier convertit les signaux électriques en données numériques et les enregistre dans la mémoire. L'énergie consommée dans cette dernière action est équivalente à la somme d'énergie pendant l'écriture et la lecture des données dans la mémoire.

L'énergie totale consommée pendant la phase d'acquisition est alors la somme

d'énergie consommée durant la capture et l'enregistrement de données dans la mémoire.

$$E_{acquisition}(b) = E_{capture}(b) + E_{enregistrement}(b) \tag{2.7}$$

En générale, l'énergie consommée dans ce module acquisition représente un faible pourcentage de l'énergie totale consommée par le noeud.

2.3.1.3 Dans le module processeur

L'énergie consommée dans le module traitement de données se divise en deux [95] : l'énergie consommée pendant le mode actif et l'énergie consommée pendant le mode en veille.

Quand le processeur est en mode actif, sa consommation d'énergie est modélisée comme suit [95] :

$$E_{Processeur,actif} = V_{dc} I_{actif} T_{actif} \tag{2.8}$$

avec,
V_{dc} : tension d'alimentation
I_{actif} : courant nécessaire pendant le mode actif
T_{actif} : durée pendant lequel le processeur est resté en mode actif
En mode veille, l'énergie est alors [95] :

$$E_{Processeur,veille} = V_{dc} I_{veille} T_{veille} \tag{2.9}$$

avec,
I_{veille} : courant nécessaire pendant le mode veille du processeur
T_{veille} : durée pendant lequel le processeur est resté en mode veille

On peut conclure que la modélisation au niveau bloc matériel ne permet pas d'identifier les différentes activités du noeuds dans le réseau. Le modèle d'énergie concerne uniquement les matériels, sans se soucier des applications qui les commandent.

Par ailleurs, la section suivante étudie le comportement du noeud capteur dans le réseau.

2.3.2 Modélisation au niveau couche protocolaire

D'autres travaux considèrent le noeud capteur à un niveau d'abstraction où l'on distingue les différentes couches protocolaires du noeud : couche application, couche transport, couche réseau, couche liaison de donnée, couche physique. A ce niveau d'abstraction, on n'entre plus en détails sur les différents blocs et modules dans le noeud capteur. L'étude se concentre plus sur les différentes applications qui commandent ces blocs matériels. Par exemple, les différents mécanismes, les protocoles utilisés par le noeud, les différentes topologies du réseau et les différentes applications sur le traitement de données.

Dans cette section, on présente les différents travaux qui proposent d'améliorer les mécanismes dans chacune des couches. Par exemple, dans la couche physique il y a le protocole MAC qui gère l'unité de communication. La grande partie des travaux sur l'optimisation d'énergie dans les noeuds capteurs se concentre sur ce protocole MAC puisque ce dernier a un impact important sur la consommation d'énergie du noeud. Il existe comme on l'a déjà dit, plusieurs variantes de ce protocole MAC (O-MAC, S-MAC, T-MAC, X-MAC, etc) qui seront détaillés dans la section suivante. Le choix d'un protocole MAC dépend aussi du type de réseau (topologie, mobilité, nombre de noeud, type de canal de communication, etc).

Dans la couche liaison de donnée, il y a le mécanisme sur la méthode d'accès au canal de communication comme le CSMA/CA avec "beacon" ou sans "beacon" (balise).

Dans la couche réseau, la plupart des travaux se focalisent sur le protocole de routage. Ce dernier a un impact important sur la consommation d'énergie.

La modélisation du protocole de routage a souvent pour objectif de proposer ou d'améliorer un algorithme pour chercher un chemin optimal à travers le réseau. Mais la finalité revient toujours à optimiser entre autres la consommation d'énergie des noeuds et du réseau.

Malgré le nombre important de variantes de protocoles de routage qui existent, on trouve encore beaucoup de travaux qui proposent des solutions de protocole de routage plus compétitifs et plus efficaces.

Voici quelques travaux récents sur la modélisation des différents protocoles [81] [83] [84], des travaux sur les différentes topologies des réseaux [85] [18] [17], les calendriers de réveil (ou duty cycle) [96] [97], différentes applications comme les techniques de compression [98] [99] [100] [101].

2.3.3 Modélisation au niveau état

Il existe aussi des études portant sur un autre niveau d'abstraction où l'on met en valeur les différents modes ou états du noeud liés aux états de l'unité de communication [102] et l'unité de traitement [103].

A ce niveau d'abstraction, le modèle de consommation d'énergie dépend de la probabilité p_i de chaque état i, la puissance P_i correspondant à chaque état en cours i et à la durée totale $Time$ de simulation. L'énergie consommée dans le module CPU est illustée dans (2.10) [103]. Dans [103], le noeud capteur est modélisé par un réseau de Pétri stochastique.

$$TotalEnergy(cpu) = (\sum_{i} P_i p_i)Time \tag{2.10}$$

avec, $i \in \{standby, powerup, idle, active\}$

Cette même approche peut aussi s'appliquer à l'échelle du noeud.

Le modèle d'énergie dans un noeud par le réseau de Pétri est illusté par la formule suivante :

$$TotalEnergy(noeud) = (\sum_{j} P_j p_j)Time \tag{2.11}$$

avec $j \in \{wait, receiving, computation, transmitting\}$
$wait$: attente d'un état
$Time$: durée totale de simulation
p_j : probabilité de l'état j
P_j : puissance consommée par l'état j

Dans le module acquisition, la consommation d'énergie dépend des différents états et des transitions d'états dans le module. Par exemple, l'énergie d'un capteur est :

$$E_{sensor} = E_{on\_off} + E_{off\_on} + E_{sensor\_run} \tag{2.12}$$

$$= e_{on\_off} + e_{off\_on} + V_s I_s T_s \tag{2.13}$$

e_{on_off} : l'énergie consommée pendant l'extinction du capteur
e_{off_on} : l'énergie consommée pendant l'activation du capteur
E_{sensor_run} : l'énergie consommée pour l'opération d'acquisition des données
V_s et I_s sont le voltage et l'intensité pour l'acquisition
T_s : l'intervalle de temps pour effectuer l'opération d'acquisition

La consommation d'énergie pour n'importe quel type de noeud capteurs peut être également généralisée comme suit [104] :

$$E_{noeud} = \sum_{i=1}^{n} E_{(etat,noeud,i)} + \sum_{j=1}^{m} E_{(trst,noeud,j)} \tag{2.14}$$

E_{noeud} est l'énergie consommée par le noeud pendant une durée de simulation.

$$E_{noeud} = \sum_{i=1}^{n} P_{(etat,noeud,i)} T_{(etat,noeud,i)} p_{(etat,noeud,i)} + \sum_{j=1}^{m} P_{(trst,noeud,j)} T_{(trst,noeud,j)} p_{(trst,noeud,j)} \tag{2.15}$$

avec,
n : nombre d'états internes du noeud
m : nombre d'états de transition du noeud
$E_{(etat,noeud,i)}(t)$: énergie consommée par l'état en cours i
$E_{(trst,noeud,j)}$: énergie consommée par l'état de transition j
$P_{(etat,noeud,i)}$: consommation de puissance de l'état i
$P_{(trst,noeud,j)}$: consommation de puissance de l'état de transition j
$T_{(etat,noeud,i)}$: durée de l'état i
$T_{(trst,noeud,j)}$: durée de l'état de transition j
$p_{(etat,noeud,i)}$: probabilité de l'état i
$p_{(trst,noeud,j)}$: probabilité de l'état de transition j

Le modèle d'énergie au niveau état est alors relatif aux différents états de la plateforme matérielle et à la probabilité de chaque état. Dans ce cas, les différentes activités du noeud dans le réseau ne sont pas considérées dans le modèle.

Pour résoudre ce problème, nous introduisons, dans la section suivante, la modélisation de la consommation d'énergie par l'approche par fonctions. Ensuite, nous développons dans le troisième chapitre tous les détails de cette approche.

2.3.4 Modélisation au niveau fonction : formalisme proposé

Tout d'abord, nous définissons une fonction comme étant une activité accomplie par le noeud dans le réseau. Dans les différents travaux de modélisations cités précédement, la plupart sont orientés vers la modélisation d'une seule fonction jugée prépondérante. Dans notre étude, nous nous intéressons en revanche plutôt à l'ensemble de ces différentes fonctions pour modéliser la consommation d'énergie du noeud capteur et du

réseau de capteurs tout entier.

Dans cette section, nous proposons donc de formaliser l'approche générale de la modélisation au niveau fonction. A ce niveau d'abstraction, un noeud est considéré comme un ensemble de fonctions. Ainsi, la modélisation de la consommation d'énergie se concentre particulièrement sur l'évolution au fil du temps de la variation de la consommation de puissance, ce que nous appelons par la suite le **profil de puissance**. Les profils de puissance de toutes les fonctions permettent d'estimer la consommation d'énergie du noeud tout entier.

Soit une fonction f comportant n **états internes**, chaque état interne i ($i \in \{\text{état}_1, \text{état}_2, ..., \text{état}_n\}$) correspond uniquement à un **niveau de consommation de puissance** P_i associé à la fonction f. Soit $P_f(t)$ le profil de puissance de la fonction f, qui correspond à l'évolution au fil du temps de la consommation de puissance. Le profil de puissance d'une fonction f est alors défini par la formule suivante :

$$\forall t, P_f(t) = P_i \tag{2.16}$$

avec, $i \in \{\text{état}_1, \text{état}_2, ..., \text{état}_n\}$
Sur une durée d'observation T, l'énergie consommée par la fonction f est décrite par $E_f = \int_T P_f(t)dt$. Et finalement, l'énergie consommée par un noeud tout entier est la somme des énergies consommées par toutes les m fonctions dans le noeud.

$$E = \sum_f \int_T P_f(t)dt \tag{2.17}$$

avec, $f \in \{fonction_1, fonction_2, ..., fonction_m\}$

Voici une synthèse des termes clés de notre approche.

f	Fonction ou activité accomplie par un noeud
$P_f(t)$	Profil de puissance de la fonction f qui est défini comme l'évolution de sa puissance consommée au cours du temps
état i	Etat interne d'une fonction f traduisant le niveau ou le stade de l'activité de f
P_i	Niveau de consommation de puissance d'un état interne i

TABLE 2.1 – Les termes clés de notre approche

La section suivante cite quelques simulateurs les plus utilisés et les plus connus

dans le domaine des réseaux de capteurs.

2.4 Les différents simulateurs des WSN

Plusieurs simulateurs sont développés pour les WSN. Ces simulateurs permettent de simuler des modèles de noeuds ou de réseaux dans un environnement virtuel.

Les simulateurs gratuits de réseaux de capteurs les plus connus sont NS2 [105] et OMNET [106]. Ils sont employés pour simuler les caractéristiques des réseaux avec des noeuds capteurs spécifiques. De plus, ils sont principalement utilisés pour comparer l'efficacité des algorithmes utilisés dans le réseau (MAC, routage, etc).

Dans ce paragraphe sont présentés quelques simulateurs parmi les plus connus, le classement et les limites de ces simulateurs et finalement une discussion et une conclusion.

2.4.1 Network Simulator NS2

NS2 est un simulateur à évènements discrets destiné à la recherche. Il intervient souvent dans l'étude des algorithmes de routage multipoint ou unipoint, des protocoles de transport, de session. Il permet de modéliser aussi très bien la couche physique du modèle OSI avec différents systèmes de transmission, filaires ou non. Cependant, il est principalement adapté aux petits réseaux. NS2 ne possède nativement aucune interface graphique. Ainsi, toutes les simulations sont réalisées en ligne de commande. Par ailleurs, grâce à l'extension ".nam" (Network Animator), on peut visualiser les résultats d'une simulation une fois achevée.

Celle-ci considère surtout l'énergie consommée dans l'unité de communication. Cette énergie est relative aux différents états internes de son module radio (émetteur-récepteur). L'un des modèles du noeud utilisé par le simulateur NS2 est mentionné dans [107] et est représenté par la figure 2.4 :

Le simulateur NS2 est "open source", il est entièrement développé en C++ et son utilisation requiert une bonne connaissance de TCL (Tool Command Language). L'utilisateur peut proposer des améliorations sur chacun des modules qui composent le modèle de noeud dans la figure 2.4. Par exemple, l'utilisateur peut proposer des nouveaux algorithmes dans les modules MAC, routage, etc.

Les différentes configurations que l'on peut faire sur NS2 sont illustrées sur la figure 2.5.

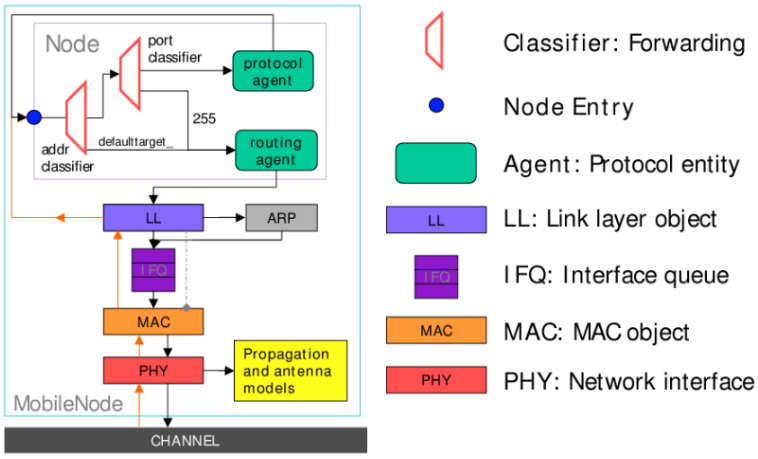

FIGURE 2.4 – Modèle de noeud utilisé dans NS2 [107]

Application	Web, ftp, telnet, générateur de trafic (CBR, ...)
Transport	TCP, UDP, RTP, SRM
Routage (couche Réseau)	Statique, dynamique (vecteur distance) et routage multipoint (DVMRP, PIM)
Gestion de file d'attente	RED, DropTail, Token bucket
Discipline de service	CBQ, SFQ, DRR, Fair queueing
Système de transmission (couche Liaison)	CSMA/CD, CSMA/CA, lien point à point

FIGURE 2.5 – Les différents choix de configurations sur NS2

Une troisième génération du simulateur Network Simulator, nommée NS3, a été développée depuis 2014.

2.4.2 OMNET++

OMNET ++ est un environnement de simulation open source. Ce simulateur offre une interface graphique solide, et un noyau de simulation intégré. Il a principalement pour but de simuler des communications réseaux mais aussi des systèmes des technologies de l'information. En effet, grâce à son architecture de base flexible, il est capable de simuler des architectures matérielles. C'est ainsi que cette plateforme est devenue connue non seulement au sein de la communauté scientifique mais aussi dans le monde industriel. Et c'est grâce à cette architecture modulaire qu'il est plus facile d'y implémenter de nouveaux protocoles.

Le simulateur OMNET fournit à la fois l'estimation de l'énergie consommée dans l'unité de communication et dans l'unité de traitement. La configuration de chaque noeud se fait par une interface graphique comme la montre la figure 2.6.

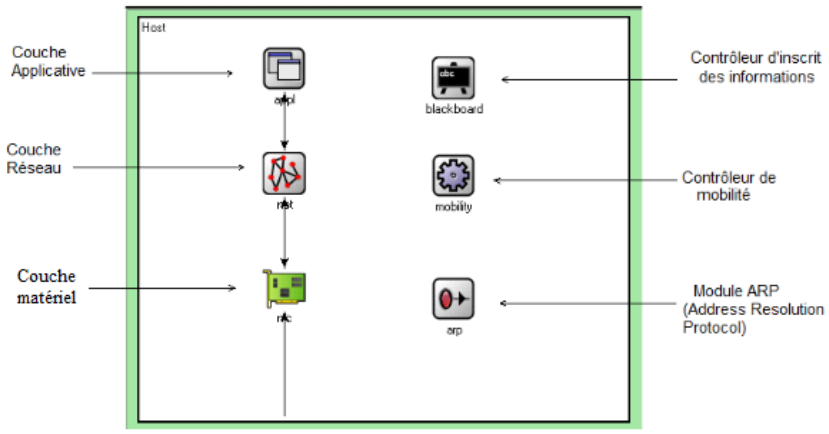

FIGURE 2.6 – Interface de configuration du modèle de noeud dans OMNET++

L'architecture du modèle de noeud [108] dans OMNET est montrée par la figure 2.7 :

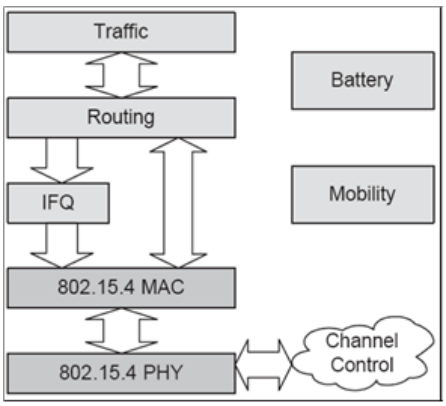

FIGURE 2.7 – Architecture d'un noeud mobile sur OMNET++ [108]

Dans OMNET++, l'architecture d'un modèle est hiérarchique, et composée de modules. Il peut y avoir un module simple ou un module composé. Un module simple correspond à un fichier .cc et un fichier .h. Un module composé comprend de simples modules ou d'autres modules composés connectés entre eux. Les paramètres, les sous modules et les ports de chaque module sont spécifiés dans un fichier .ned.

Ainsi, l'utilisateur peut intervenir sur chacun des modules montrés dans la figure 2.7.

2.4.3 OPNET

Le simulateur OPNET [109] est développé avec le langage C++. C'est un simulateur de réseau basé sur une interface graphique intuitive. Sa prise en main et son utilisation est relativement aisée. Le simulateur OPNET dispose de trois niveaux de domaine hiérarchique, respectivement de niveau plus haut au niveau plus bas : network domain, node domain et process domain. La simulation sous OPNET fournit en standard une liste d'implémentations de routeurs, de stations de travail, de switchs, etc.

L'utilisateur peut construire un modèle de réseau en utilisant les modèles de noeuds prédéfinis et fournit par la librairie d'OPNET. Autrement, l'utilisateur peut définir lui même le modèle des liens, le modèle de noeud en tant que routeur ou hôte, etc.

Ce simualateur n'est gratuit que pour les Universités.

2.4.4 GloMoSim

GloMoSim [110] permet la simulation d'environnement à grande échelle pour des réseaux sans fil et filaires. Il a été développé selon les capacités du langage Parsec. Il est capable de simuler un réseau purement sans fil, avec tous les protocoles de routage que cela implique (AODV, DSR, algorithme de Bellman-Ford (routage par vecteur de distance), ODMRP, WRP, FSR, ...). A l'avenir, des nouvelles versions pourront simuler à la fois un réseau filaire et un réseau hybride (filaire et sans fil). La plupart des systèmes réseaux de GloMoSim sont construits en utilisant une approche basée sur l'architecture à sept couches du modèle OSI.

De plus, l'intégration de modules supplémentaires ne nécessite pas la compréhension du fonctionnement du noyau. L'utilisation précise de Parsec s'avère suffisante. GloMoSim est un simulateur caractérisé par une grande portabilité (Sun Solaris, Linux, Windows) et sa licence est gratuite pour les universitaires. Cependant, le simulateur GloMoSim ne modélise pas la consommation d'énergie au niveau des noeuds.

2.4.5 J-Sim

Le simulateur J-Sim [111] permet de simuler des réseaux de l'ordre de 1000 noeuds. Le simulateur utilise deux langages, Java et TCL, quasi indifféremment.

L'architecture et le code sont suffisamment bien structurés pour une prise en main relativement rapide. De plus il permet d'utiliser n'importe quelle application Java en tant que générateur de trafic. Par ailleurs, 802.11 est la seule norme modélisée dans J-Sim, notamment au niveau couche MAC. De plus, il est difficilement maniable.

2.4.6 SENS

Le simulateur SENS [112] considère la consommation d'énergie liée à l'acquisition des données environnementales. Le modèle d'environnement, avec lequel les noeuds capteurs interagissent, est fortement considéré dans ce simulateur. Le modèle de noeud dans SENS est montré par la figure 2.8.

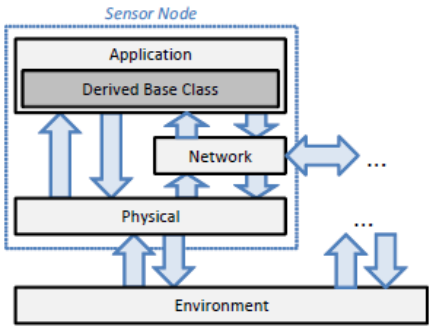

FIGURE 2.8 – Modèle d'un noeud sur SENS [112]

Les protocoles MAC et routage ne sont pas modifiables dans le simulateur SENS. De plus, les applications que l'on peut simuler sont limitées.

2.4.7 PAWiS

PAWiS [113] est un environnement de modélisation pour le simulateur OMNET++. Il modélise la puissance consommée par les unités de traitement sur chaque noeud. Le canal de communication est également modélisé de façon à pouvoir prendre en compte les bruits, les interférences et la perte de puissance de signal. Plusieurs méthodes d'accès au canal sont également fournies avec PAWiS. En outre, PAWiS n'est qu'un environnement de modélisation et son utilisation doit être maintenue dans le simulateur OMNET++.

2.4.8 SENSE

Le simulateur SENSE [114] a été développé pour simuler des réseaux de grande taille. Il permet de simuler la consommation des différents composants du noeud. Il intègre deux modèles de la couche MAC. Par ailleurs, ce simulateur est difficile à manipuler. De plus, les modèles de propagation radio et de capture de données environnementales ne sont pas implémentés dans ce simulateur.

2.4.9 Les émulateurs basés des matériels ou logiciels spécifiques

Tout d'abord, un émulateur est un sous ensemble de simulateur qui permet d'analyser les codes destinés à être embarqués dans la plateforme ciblée. Il permet d'émuler de manière réaliste le comportement du logiciel embarqué.

Il existe plusieurs types d'émulateurs et ils sont souvent dédiés à des plateformes ou logiciels spécifiques. Par exemple ceux qui permettent de simuler des noeuds qui ont leur propre système d'exploitation, telque TOSSIM [115] pour les noeuds qui utilisent TINYOS, ou COOJA [116] pour les noeuds qui utilisent CONTIKI, etc. Le principe général de ces émulateurs est d'exécuter sur ordinateur les applications destinées à être implémentées dans le noeud. Ainsi, certain émulateurs permettent d'estimer la consommation d'énergie qui correspond aux détails du comportement du noeud. Mais ils sont limités aux systèmes d'exploitation utilisés. Par exemple, Power-TOSSIM permet d'estimer la consommation d'énergie des applications simulées.

Bien que COOJA soit initialement destiné à simuler des applications sur des noeuds utilisant CONTIKI, il est également capable d'émuler TinyOS. Il permet également d'émuler le module radio sur lequel transitent les données.

ATEMU est un émulateur dédié à la plateforme MICA2. Il est réputé pour sa précision de la simulation du jeu d'instruction dédié au microcontrôleur ATMega128. Cependant, il ne considère pas la consommation d'énergie.

Avrora est un émulateur dédié à l'architecture AVR de ATMEL. Il permet d'analyser l'application et de simuler un ensemble de noeud embarquant cette application en tenant compte des communications sans fil. Cependant, il ne considère pas la consommation d'énergie des plateformes simulées.

D'autres simulateurs sont destinés à des technologies bien spécifiques. Par exemple, Packet Tracer [117] qui est un outil de simulation pour l'équipement CISCO.

2.4.10 La fusion des simulateurs

Il existe aussi des simulateurs qui peuvent intégrer d'autres simulateurs et qui se chargent de les fusionner. C'est le cas du simulateur WorldSens [118]. Il associe deux autres simulateurs (WSim et WSNet). Le simulateur WSim est utilisé pour simuler le fonctionnement de chaque noeud capteur tandis que WSNet est utilisé pour simuler le système radio dans le réseau. WSim est un simulateur de plateforme matériel pour les noeuds capteurs. Il permet d'aider le concepteur sur le choix de la couche physique.

La figure 2.9 montre un exemple de modèle de noeud simulé avec WSim. L'utilisateur peut simuler la configuration des différents modules matériels qui composent le

modèle de noeud. Et l'estimation d'énergie dépend de ces configurations et des différentes applications qui pourraient commander ces modules.

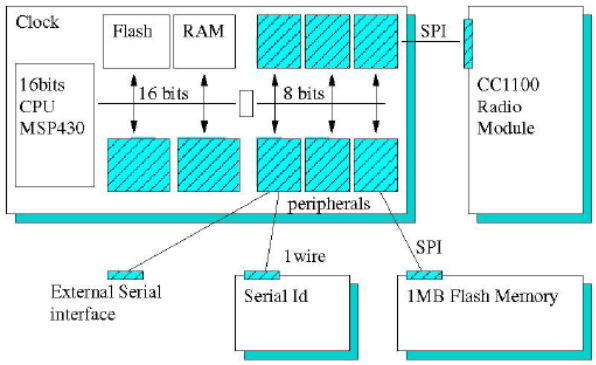

FIGURE 2.9 – Exemple d'un modèle physique de noeud capteur sur WSim [118]

Selon la figure 2.10, WSNet simule tout ce qui concerne la partie radio et la connexion au réseau. Il permet d'aider les concepteurs sur le choix du protocole, le modèle de trafic, le réglage des paramètres sur le module radio et aussi sur le choix des matériels. Il travaille conjointement avec WSim pour simuler le réseau complet. La figure 2.10 montre le bloc architecture du WSNet. On remarque dans ce bloc architecture qu'il n'y a pas de module pour le dispositif d'acquisition. Par conséquent ce simulateur ne tient pas compte de l'énergie consommée par celui-ci. L'utilisateur peut intervenir sur chacun des modules dans le modèle de la figure 2.10.

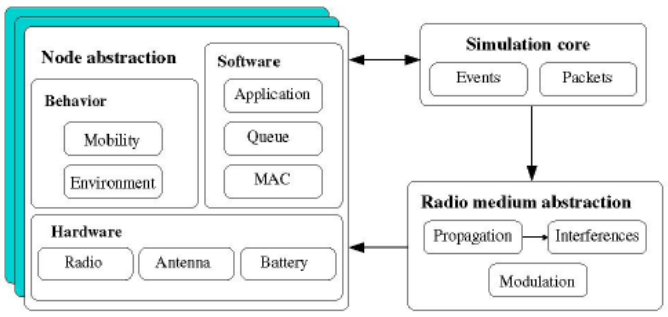

FIGURE 2.10 – Bloc architecture du WSNet [118]

Ce simulateur est utile pour simuler les couches physiques et radio du noeud capteurs. Il peut aider le concepteur sur le choix de la modulation, le mode de propagation, la fréquence d'émission. En outre, ce simulateur utilise des noeuds capteurs modélisés à un bas niveau d'abstraction.

2.4.11 CAPNET

CAPNET [119] est un simulateur pour l'utilisation industrielle et peut aider les concepteurs sur les choix des matériels utilisés avant la conception du noeud. Ce simulateur permet par exemple le choix du processeur, la mémoire et le type de communication radio utilisés par les noeuds suivant sa consommation d'énergie. Par contre, le simulateur CAPNET n'est pas open source. L'utilisateur ne peut changer, proposer ou enlever des applications et des mécanismes dans le modèle de noeud. Il ne peut que choisir les différentes options de configuration dans le simulateur.

2.4.12 IDEA1

Un autre simulateur "IDEA1" est proposé dans [88], il permet l'exploration de l'espace de conception. L'architecture des noeuds dans IDEA1 est définie avec un bas niveau d'abstraction comme celle dans NS3 qui est une extension de NS2.

D'après l'architecture du modèle du noeud dans IDEA1 montrée par la figure 2.11, l'estimation d'énergie est basée sur les différents modules matériels implémentés dans le modèle de noeud.

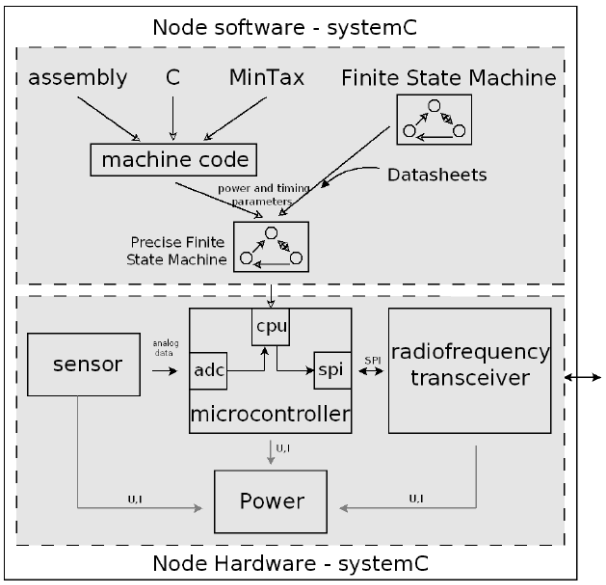

FIGURE 2.11 – Modèle du noeud dans IDEA1 [88]

2.4.13 WISENES

WISENES [120] est un simulateur qui permet l'aide à la conception, la simulation, l'évaluation et l'implémentation des noeuds capteurs dans le réseau. Il est ciblé pour la conception et le déploiement des réseaux de capteurs réels. Il opère à un haut niveau d'abstraction pour la conception de protocole et pourtant produit des résultats précis sur les performances du noeud et du réseau. La différence par rapport aux autres simulateurs est que WISENES permet de modéliser à haut niveau d'abstraction et génère automatiquement des codes pour la mise en oeuvre effective. Cependant, il ne considère pas les différentes activités du noeud dans le réseau.

2.5 Classification et limites de quelques simulateurs

Les modèles de noeud dans NS2 et OMNET sont constitués de différents modules, tels que le module PHY, le module MAC, etc. Ces deux simulateurs fournissent la consommation d'énergie suivant la configuration des noeuds. Mais dans l'architecture du modèle des noeuds montrés dans les figures 2.4 et 2.7 respectivement dans NS2 [107] et OMNET [108], l'utilisateur ne peut proposer des algorithmes qui communiquent simultanément dans différents modules. En effet, dans les modèles de noeud, bien que les deux simulateurs soient open source, et que l'utilisateur puisse proposer des applications dans chaque module du modèle, l'utilisateur ne peut pas implémenter des applications communicantes, dépendantes et qui doivent s'exécuter simultanément dans différents modules.

Aucun module ne prend en charge le traitement de données comme la compression de données, la prise de décision, etc. De plus, les deux modèles de noeud qui existent dans NS2 et OMNET ne tiennent pas compte de la consommation d'énergie dans le dispositif d'acquisition.

On remarque que certains simulateurs ne dépendent pas des architectures ou des technologies déjà existantes, ils offrent aux utilisateurs la possibilité d'imaginer et de choisir les caractéristiques des matériels adéquats à la simulation. Dans ce cas, les résultats de simulations peuvent fournir aux concepteurs des informations avant la phase de conception des noeuds.

La liste des simulateurs que l'on a décrit ici est loin d'être exhaustive. Il existe encore plusieurs simulateurs (par exemple : Matsnl [121], SensorSim [122], Sidh [123], etc) qui considèrent les modèles de consommation d'énergie au niveau des noeuds. Pourtant, ces simualteurs ne répondent pas à nos besoins en terme de disponibilité du

logiciels qui ne sont pas "open source". De plus, ils utilisent des modèles de noeuds spécifiques.

Dans la figure 2.12, nous proposons une classification des différents simulateurs mentionnés précédement ainsi que d'autres simulateurs suivant le niveau d'abstraction et l'interêt dans le processus de conception du noeud capteur.

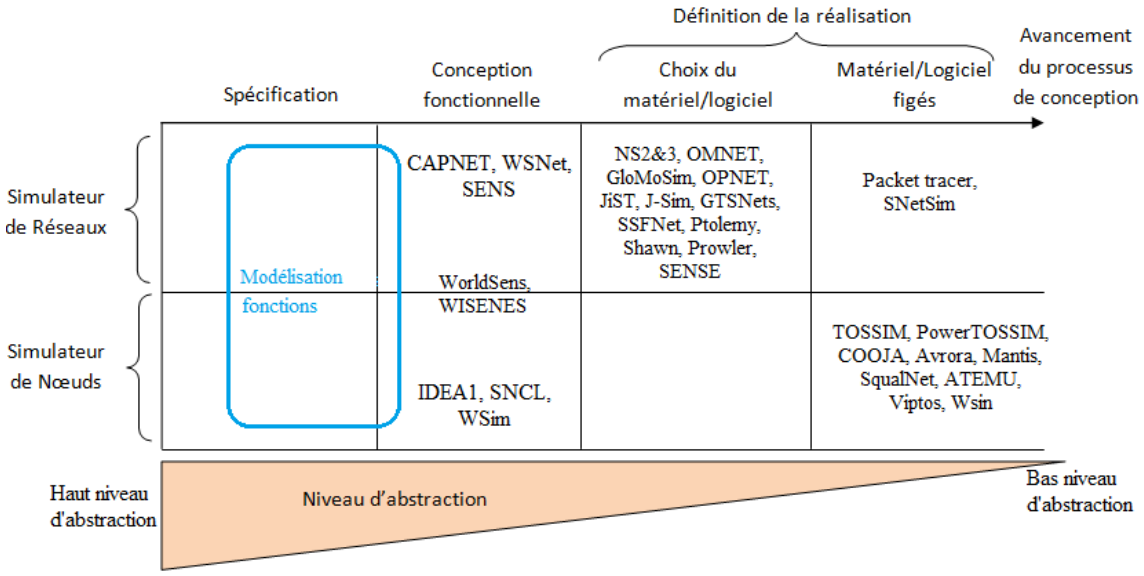

FIGURE 2.12 – Classement de quelques simulateurs de WSN

La conception d'un système est bien souvent trop complexe pour être réalisée en une seule étape. En effet, le processus de conception est divisé en plusieurs étapes (voir figure 2.12) permettant ainsi d'aborder le problème avec un degré de détail progressif. Ainsi, la modélisation du noeud doit être faite dans un premier temps à un haut niveau d'abstraction afin de simuler les différentes activités du noeud dans le réseau, et en tenant compte de la consommation d'énergie. Dans ce cas, la consommation d'énergie et les différentes activités du noeud sont considérées très tôt dans le processus de conception, notamment pendant les phases de spécification et conception fonctionnelle. Bien évidement, la modélisation concerne les noeuds capteurs et le réseau tout entier.

On peut remarquer sur la figure 2.12 que plusieurs simulateurs de réseaux de capteurs sans fil sont basés sur des noeuds spécifiques. Ils utilisent souvent des modèles matériels/logiciels prédéfinis. La plupart des modèles de noeuds dans ces simulateurs concernent un bas niveau d'abstration. Dans ce cas, l'estimation d'énergie est faite au niveau matériel, elle peut être détaillée et précise. Ce niveau d'abstraction ne permet donc pas de considérer les fonctions qui commandent ces matériels surtout dans les

cas où plusieurs fonctions sont supportées par une même plateforme. Ainsi, la modélisation à bas niveau d'abstraction est souvent destinée à simuler les systèmes de noeud après les étapes de conception, et elle n'est pas adaptée à la description des différentes activitées des noeuds dans le réseau. Afin de bien considérer l'impact en énergie des différentes fonctions à implémenter dans le modèle de noeud, il est nécessaire de modéliser le lien direct entre chacune des fonctions et leurs consommations respectives. Cette modélisation par fonctions doit pouvoir se faire indépendament du fait que les fonctions pourront être implémentées sur des circuits éventuellement partagés et sans nécessairement considérer leur partitionnement (matériel/logiciel). La figure 2.12 illustre que pour considérer les différentes activités du noeud, il faut modéliser les noeuds capteurs et le réseau tout entier. La modélisation doit commencer dans un premier temps à un haut niveau d'abstraction. Ceci permet de prendre en compte les phases de spécification et de conception fonctionnelle très tôt dans l'avancement du processus de conception.

2.6 Discussion et conclusion

Tout d'abord, nos besoins et nos attentes pour un modèle de noeud sont :

- Prendre en compte les différentes activités du noeud dans le réseau. Le modèle devra être capable de distinguer l'impact en énergie lorsque l'on ajoute, remplace ou enlève une fonction dans le noeud,
- Ne pas dépendre dans un premier temps des technologies puisque les résultats de simulation doivent donner des informations aux concepteurs dès les premières étapes de conception du noeud.

Le simulateur qu'il nous faut doit avoir les propriétés suivantes :

- le parallélisme permettant l'exécution de plusieurs tâches simultanément,
- le déclenchement par évènement permettant de synchroniser plusieurs fonctions ou de déclencher un changement d'état d'une fonction,
- la gestion du temps permettant de générer une durée de période en boucle, en guise d'horloge interne d'une fonction,
- la notion de pause permettant de stopper un processus en cours et de le remettre en marche au moment voulu.

Dans ce chapitre, nous avons pu voir plusieurs travaux de modélisation de noeuds capteurs et de réseaux de capteurs sur plusieurs niveaux d'abstraction. Nous constatons qu'à notre connaissance, qu'aucune étude n'a été effectuée concernant la modélisation de la consommation d'énergie liée aux différentes activités des noeuds dans le réseau. Ensuite, nous avons cité quelques simulateurs correspondant aux différents modèles définis par leurs niveaux d'abstraction. Nous remarquons alors que ces simulateurs ne sont pas adaptés à la prévision de la consommation en tenant compte des activités (fonctions) des noeuds dans le réseau.

Ces constatations nous amènent à proposer une modélisation avec une approche par fonctions. Cette approche par fonctions permettra de considérer toute la consommation d'énergie dans un noeud relative à ses activités dans le réseau. Dans un processus de définition et de conception, cette approche doit permettre d'estimer la consommation d'énergie pour différentes configurations fonctionnelles, c'est-à-dire, en ajoutant, modifiant ou retirant des fonctions du modèle du noeud. La modélisation utilisant cette approche par fonctions permet également de ne pas dépendre des différentes technologies existantes. Les résultats de simulation doivent fournir l'évolution au fil du temps de la variation de la consommation de puissance. Ces résultats permettraient d'aider le concepteur dans le choix du matériel et du logiciel qui seront utilisés dès les premières étapes de conception du noeud. Et finalement, cette approche par fonctions doit principalement apporter une aide importante pour l'optimisation énergétique des WSN.

3

Modélisation par fonctions

Tout d'abord, nous considérons qu'une fonction est définie comme une opération ou une activité que le noeud capteur doit accomplir dans le réseau. Les différentes fonctions sont normalement identifiées dans le cahier de charges avant la conception du noeud, par exemple : une fonction d'acquisition, une fonction d'émission, etc.

Le nombre et le type de ces fonctions dépendent alors de l'application, des caractéristiques du réseau et des noeuds. Par exemple pour un noeud mobile, la localisation est une fonction nécessaire pour le noeud. Pour une topologie cluster, une fonction d'estimation de niveau de batterie peut être nécessaire pour permettre de choisir la tête de cluster parmi les noeuds, ceci afin d'équilibrer le niveau d'énergie de tous les noeuds dans le réseau. La fonction routage est une fonction dont l'algorithme est choisi suivant la topologie, le type du réseau et les caractéristiques du noeud. Ces fonctions sont alors implémentées dans le noeud et il existe plusieurs algorithmes et variantes pour ces fonctions. Le concepteur choisit uniquement les fonctions et les types d'algorithmes adaptés aux besoins du système choisi (topologie, dynamicité du réseau, type de noeud, etc).

Dans notre étude, on s'interesse surtout à la modélisation de la consommation d'énergie liée à chaque fonction du noeud capteur. Il est donc nécessaire d'identifier d'abord toutes les fonctions supportées par le noeud. Ces fonctions sont alors identifiées grâce au comportement attendu du noeud dans le réseau. Parmi les différentes opérations que le noeud doit accomplir, on identifie toutes les fonctions qui ont un

impact sur la consommation d'énergie du noeud capteur.

3.1 Quelques fonctions les plus utilisées

Dans cette section, quelques fonctions les plus utiles ou les plus fréquentes sont présentées. Cette présentation est loin d'être exhaustive car, selon les besoins de l'application, de nouvelles fonctions peuvent être nécessaires.

- La fonction réception

Cette fonction est utilisée pour recevoir tous les types d'informations : trames de contrôles, trames de données, etc. Elle peut collaborer avec d'autres fonctions (émission, compression, etc) en modifiant leur état ou en leur transmettant des données. De plus, elle peut contrôler le bloc récepteur à se mettre en veille ou à se reveiller selon le scénario.

- La fonction émission

A l'opposé de la fonction réception, la fonction émission permet d'envoyer vers le réseau des trames de contrôles et des trames de données, etc. Dans notre exemple, l'énergie d'émission dépend de certains paramètres, comme la distance entre les noeuds émetteur et récepteur ainsi que la taille des données à transmettre. La fonction émission gère l'état du bloc physique émetteur et elle peut collaborer avec d'autres fonctions du noeud.

- La fonction acquisition

La fonction acquisition a pour objet d'obtenir des informations de l'environnement (élément observé) afin que celles-ci soit transmises vers le puits du réseau.

Il peut y avoir 2 modes de fonctionnement pour cette fonction : le mode évènementiel et le mode périodique. Le mode périodique repose sur l'idée selon laquelle le dispositif d'acquisition est actif ou inactif périodiquement. Ce type d'opération est approprié pour les applications qui nécessitent un prélevement périodique des données. Par exemple, cela est utile dans des applications de supervision (température, météo, humidité, etc) afin d'établir des rapports périodiques.

A l'inverse, le mode évènementiel repose sur l'apparition d'évènements bien précis et le dispositif est actif selon l'apparition ou non de cet événement (exemple : un détecteur de mouvement). Ce type d'opération est souvent utilisé dans des applications qui doivent réagir à des apparitions soudaines d'évènements.

Indépendament du mode, les données acquises seront ensuite combinées avec les données reçues par la fonction réception pour être compressées ou pas, afin de les transmettre via la fonction émission.

- La fonction de localisation

Pour se localiser dans un réseau, un noeud doit d'abord mesurer la distance entre lui et ses points de repère en utilisant des techniques de mesure de distance telles que l'AoA, le ToA, le TDoA par exemple. L'AoA [124] consiste à déterminer l'angle d'arrivée de tous les signaux reçus de chaque noeud du voisinage, le ToA et ToF (Time of Flight) [125] sont utilisés pour connaître le temps de propagation et pour estimer la distance entre les noeuds, le TDoA [126] permet d'évaluer la différence des temps d'arrivée de deux ou plusieurs différents signaux. Ces signaux peuvent provenir de deux noeuds de référence distincts ou peuvent être de natures différentes, comme les ultrasons et les signaux radio qui peuvent être émis par une même source. Ensuite, on peut obtenir la position du noeud par des algorithmes de multilatération, de triangulation ou de trilatération.

- La fonction de routage

Les protocoles de routage définissent le comportement des noeuds capteurs lors du calcul du chemin optimal à travers le réseau. Il existe plusieurs variantes de ces protocoles de routage. Les travaux dans [83] comparent quelques uns de ces protocoles de routage. Pour les noeuds capteurs mobiles, le protocole de routage utilisé est parfois associé au routage géographique [84] [64], d'où la nécessité de la fonction localisation, des mécanismes de découverte de voisins et d'échange des coordonnées des noeuds entre voisins.

- La fonction compression

Cette fonction permet de compresser les données en utilisant un algorithme relatif au taux de compression voulu [99]. Ce taux de compression aura un impact sur la taille de données transmises, donc le temps d'émission de données et la consommation de la fonction associée. Il existe néamoins plusieurs algorithmes de compression adaptées au taux de compression voulu. Le choix du taux de compression et du type d'algorithme de compression dépendent alors de certains paramètres, par exemple [98] : le type de donnée à capturer, l'état du lien entre le noeud émetteur et le noeud récepteur, la puissance d'émission.

3.2 Modélisation avec l'approche par fonctions

Dans cette étude, on s'intéresse surtout à la consommation d'énergie des fonctions implémentées dans le noeud capteur. Pour pouvoir bien définir les caractéristiques de chaque fonction, on propose dans cette section une classification des différentes fonctions possibles selon leur mode d'activation et la manière dont elles gèrent leurs

données.

3.2.1 Classification des fonctions

Il existe plusieurs façons de classifier une fonction. Parmi les critères caractérisant une fonction et influant l'évolution de la consommation associée à la fonction, nous en avons retenu deux qui sont le mode d'activation de la fonction, et la gestion des données reçues par la fonction. La classification selon ces deux critères est présentée par la suite.

L'activation d'une fonction est liée à la nature de l'évènement déclenchant pour la fonction. Quatres sources de déclenchement sont à considérer :

- un évènement aléatoire,
- un évènement déterministe,
- l'état d'entrée (activation par l'arrivée des données à l'entrée de la fonction),
- ou une fonction peut être tout simplement de type permanent.

L'activation par évènement aléatoire est basée sur l'apparition d'un évènement spécifique externe du noeud, comme par exemple la détection d'une alarme du noeud. L'instant d'apparition de ce genre d'évènement n'est pas connu par avance lors de la conception de l'application.

Une fonction activée par un évènement déterministe est une fonction basée sur un évènement périodique ou un évènement dont on connaît à l'avance sa date d'apparition. Ici, on considère qu'un évènement est toujours périodique même si la période contient une partie aléatoire, c'est-à-dire que la période n'est pas très précise et qu'on l'estime avec une petite marge d'erreur.

Une fonction peut être aussi activée par l'arrivée des données à l'entrée de celle-ci. Dans ce cas, on dit que la fonction est activée par son état d'entrée puisque les données (trames) reçues à l'entrée changent l'état de la fonction.

Une fonction de type permanente est toujours active qu'il y ait une apparition d'évènement ou pas.

Dans notre étude, on s'intéresse aussi à la façon dont les fonctions gèrent les données qu'elles reçoivent en entrée, pour produire des données (sortie de la fonction). En effet, outre le temps mis par la fonction pour effectuer son traitement, cette information influe sur la consommation de la fonction. Trois modes sont considérés ici :

- Fonction de transfert de données, si la fonction retransmet les données sans les modifier et sans ajout ou suppression des champs de contrôles. La quantité et la taille de données à l'entrée de la fonction ne changent pas à la sortie de celle-ci,

- Fonction de traitement de données, si la fonction fait des traitements sur les données reçues ou d'envoyer des résultats issus du traitement ou de prendre des décisions suivant les résultats obtenus. La fonction peut donc décider d'augmenter ou de réduire le nombre de données en sortie de la fonction par rapport au nombre de données en entrée de celle-ci. De plus, la fonction peut modifier la quantité de ses données en sortie par rapport à la quantité des données qu'elle a eu en entrée,

- Fonction de contrôle/décision de données qui peut modifier l'état d'une fonction dans le noeud capteur. Ce type de fonction a des données en entrée, mais n'a pas de données en sortie. Elle prend des décisions selon les données reçues, et peut contrôler le changement d'état interne d'une fonction dans le modèle de noeud.

La table 3.1 ci-dessous résume la classification des fonctions suivant le mode d'activation et le mode de gestion de données.

Mode d'activation	Mode de gestion de données
Evènement aléatoire	Traitement de donnée
Evènement déterministe	Transfert de données
Etat d'entrée	Contrôle/Décision sur les données
Active en permanence	

TABLE 3.1 – Classification des fonctions

Toutes les combinaisons entre ces deux critères sont possibles, le mode d'activation n'ayant a priori pas d'influence sur la gestion des données.

3.2.2 Modélisation des fonctions

Dans cette sous section, la modélisation d'une fonction générique d'un noeud est présentée.

L'énergie consommée par une fonction dépend de son comportement ainsi que deux critères : le vecteur d'entrée de la fonction et le mode d'activation de la fonction. Le vecteur d'entrée définit les données qui sont reçues par la fonction. La taille de ces données définit la durée pendant laquelle la fonction traite les données. L'état de

la fonction dépend d'une part du mode d'activation de la fonction et d'autre part des changements de mode imposés par les autres fonctions du noeud capteur. Par exemple, lors d'une émission de données, la fonction émission peut imposer une mise en veille de la fonction réception. En résumé, l'état en cours définit la puissance consommée par la fonction et la taille des données traitées peut influencer la durée pendant laquelle la fonction est restée sur cet état.

Dans notre étude, on s'est limité à deux types de résultats. Le premier est lié à l'objectif de cette étude, à savoir les profils de puissances. Le deuxième type de résultat concerne les données (vecteur de sortie) et le contrôle (mode de sortie) en sortie de la fonction. Ces sorties permettent de relier une fonction aux autres fonctions du noeud pour construire le modèle du noeud tout entier.

La figure 3.1 représente une vue externe de la modélisation proposée pour une fonction. Le lien "Vecteur d'entrée" et le lien "Vecteur de sortie" représentent plusieurs types de données (coordonnées, données acquises, niveau de batterie, etc). Le lien "Mode d'entrée" permet de demander le changement d'état interne de la fonction et le lien "Mode de sortie" permet à une fonction d'imposer un changement d'état aux autres fonctions du modèle. Selon le comportement de la fonction, le lien "profil de puissance" fournit l'évolution au cours du temps de la puissance consommée par la fonction.

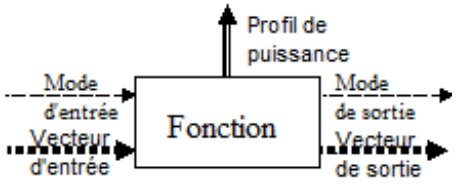

FIGURE 3.1 – Bloc représentatif d'une fonction

La figure 3.2 illustre les caractéristiques internes du modèle d'une fonction. Le flot "comportement de la fonction" gère les différents états internes en fonction des conditions liées à l'entrée mode et au vecteur d'entrée. Ces deux derniers peuvent tous les deux déclencher le changement d'états internes d'une fonction. Le flot consommation de puissance intègre les niveaux de puissance correspondant aux différents états internes positionnés par le flot comportement, il comprend donc un algorithme qui permet d'indiquer la consommation de puissance correspondant à l'état en cours. Les deux flots sont donc liés par un lien interne à la fonction : "Etat en cours". Ainsi, à chaque état interne correspond un niveau de puissance. Ce dernier peut être obtenu par mesure réelle ; ou fourni par le "Datasheet components", ou par calcul ou par l'utili-

sation d'une méthodologie comme FLPA [127]. Cette dernière permet de modéliser de façon simple des architectures complexes et de fournir des estimations de bonnes précisions.

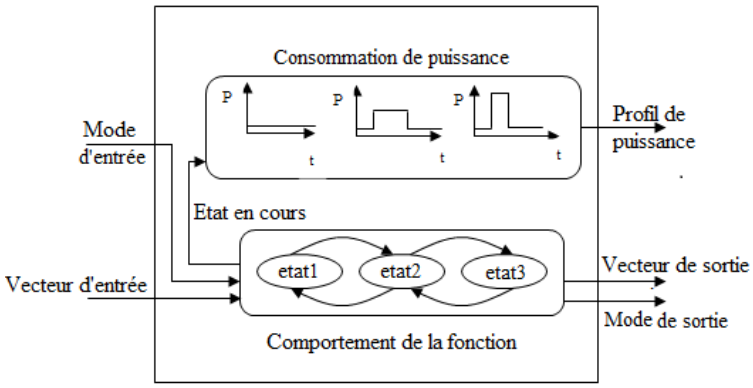

FIGURE 3.2 – Caractéristique interne d'une fonction

La description du comportement d'une fonction revient à décrire les conditions de changement d'état interne de la fonction suivant l'entrée mode et le vecteur d'entrée, ainsi que l'évolution du vecteur de sortie. Le profil de puissance fournit quant à lui l'évolution au cours du temps de la puissance consommée, suivant l'état interne adopté par la fonction (comportement de la fonction) et suivant l'entrée mode.

De manière générale, une fonction peut avoir plusieurs vecteurs d'entrées provenant de plusieurs fonctions. Le vecteur de sortie représente les productions de données issues de la fonction qui sont également une source de dissipation d'énergie.

Pour obtenir le résultat de l'évaluation de l'énergie consommée par la fonction, on intègre les profils de puissances de consommation suivant l'axe de temps.

Pour pouvoir simuler les modèles mentionnés précédemments, on a besoin d'un simulateur qui considère à la fois les noeuds capteurs et le réseau complet. Dans la suite de cette section, nous allons mentionner les caractéristiques du simulateur qu'il nous faut pour prendre en considération le type de modèle proposé. Le simulateur doit supporter :

- le parallélisme pour pouvoir simuler l'exécution simultanée de plusieurs tâches,
- le déclenchement par évènement (en utilisant la notion de pause et continue) permettant de synchroniser plusieurs tâches ou plusieurs fonctions ou de déclencher un changement d'état d'une fonction,
- la gestion de temps permettant de générer une durée de période en boucle.

3.3 Modélisation en utilisant SystemC

La création d'un modèle implique l'utilisation d'un langage permettant sa simulation. Parmi les différentes possibilités offertes, nous avons retenu une implantation du modèle utilisant SystemC. Ce dernier et l'implantation des modèles décrits sont présentés dans ce chapitre.

3.3.1 Généralités sur SystemC

SystemC n'est pas vraiment un langage à part entière mais se présente comme un surensemble du langage C++.

Il est utilisé pour compléter les manques de VHDL pour modéliser les hauts niveaux d'abstraction et de langage C pour modéliser les aspects matériel. Il répond aux besoins de conception de systèmes associant des ressources matérielles et logicielles. Il est utilisé dans le processus de conception de systèmes embarqués afin de :

- favoriser les phases d'exploration architecturale,
- valider le logiciel embarqué.

SystemC est adapté pour la modélisation des systèmes embarqués, nécessitant :

- la gestion du temps,
- la gestion du parallélisme,
- la gestion des communications,
- la structuration hiérarchique,
- la spécification fine des types de données.

SystemC peut modéliser à plusieurs niveaux d'abstraction. On peut l'utiliser pour modéliser le noeud capteur ou le réseau de capteurs entier à plusieurs niveaux d'abstraction.

On a choisi d'utiliser SystemC pour simuler les modèles graphiques précédents car les aspects de SystemC qui nous intéressent sont les suivants :

- SystemC permet de modéliser à un haut niveau d'abstraction, par exemple TLM (Transaction Level Modeling). Ce dernier est parmi les niveaux dits transactionnels utilisés pour le prototypage du logiciel,

- SystemC permet de modéliser des évènements avec un temps de simulation rapide,

- SystemC permet de considérer le parallélisme,

- SystemC permet de modéliser la communication et la synchronisation entre les éléments en parallèles.

Ces aspects répondent totalement aux besoin exprimés en fin du paragraphe précédent.

3.3.2 Modélisation d'une fonction avec SystemC

Dans le but de simuler l'application, les modèles graphiques présentés précédement sont transformés en modèle SystemC.

Le principe retenu consiste à utiliser l'objet module de SystemC pour représenter chaque fonction. A l'intérieur de ce module, au moins deux threads sont nécessaires, chacun étant lié à un flot de la fonction (voir figure 3.2).

La figure 3.3 représente la structure interne du modèle SystemC d'une fonction sans considérer les threads additionnels pour l'activation, c'est-à-dire une fonction pour laquelle la condition d'activation est un évènement aléatoire ou par état d'entrée.

Le mode d'activation doit être considéré dans le modèle selon la classification des fonctions présenté par la table 3.1. Dans le cas d'une fonction activée par l'état d'entrée ou d'une fonction activée par un évènement aléatoire, l'activation est définie par un code SystemC à l'extérieur à la fonction. Par contre, pour une fonction activée par un évènement déterministe, un troisième thread sera introduit dans le modèle pour activer le thread du comportement de la fonction.

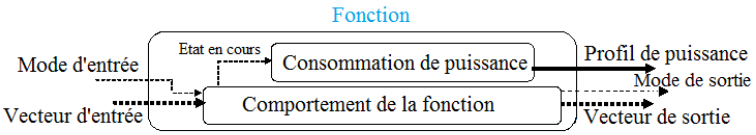

FIGURE 3.3 – Modèle de fonction activé par un évènement extérieur

La figure 3.4 représente la structure interne du modèle SystemC d'une fonction activée par une horloge interne, par exemple un évènement déterministe. Comme il a été dit auparavant, la période de cet évènement peut contenir une part aléatoire.

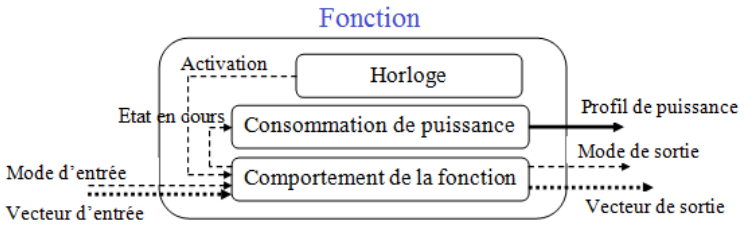

FIGURE 3.4 – Modèle de fonction activé par une horloge interne

3.3.3 Modélisation de la consommation d'énergie totale d'un noeud capteur

Rappelons que l'objectif du modèle est d'obtenir une image de l'évolution de la consommation du noeud au cours du temps. Pour cela, un noeud est représenté par un ensemble de fonctions qui sont liées entre elles par les vecteurs d'entrée/sortie et peuvent être contrôlées par l'entrée et la sortie mode.

La modélisation de la consommation d'énergie se concentre sur l'évolution du profil de puissance dans chaque fonction pour estimer la consommation d'énergie totale d'un noeud suivant le temps de simulation.

Il est donc nécessaire, pour modéliser totalement le noeud, de prendre en compte la consommation instantannée de chaque fonction pour obtenir la consommation globale du noeud.

Pour cela, un thread supplémentaire est ajouté au modèle globale de chaque noeud. Ce thread a en entrée toutes les consommations des fonctions et calcule la consommation totale du noeud. Il est activé à chaque modification de l'état en cours d'une fonction.

3.4 Exemple de modélisation d'un noeud capteur

Afin d'expérimenter le modèle de base sur des situations concrètes, trois situations ont été envisagées et sont présentées dans cette sous section.

Dans cette étude, un noeud capteur est désormais considéré comme un ensemble de fonctions. Pour créer un modèle, il s'agit dans un premier temps d'identifier les fonctions supportées par le noeud, donc les fonctions à modéliser.

3.4.1 Modélisation d'un noeud capteur simple

Pour bien illustrer le modèle, nous allons prendre l'exemple simple d'un noeud composé de trois fonctions de bases : la fonction réception, la fonction acquisition et la fonction émission.

Nous considérons que les fonctions réception et émission comprennent trois états, respectivement l'état veille, l'état écoute, l'état réception et l'état veille, l'état "idle" ou en attente, l'état émission. Nous supposons que la fonction acquisition comprend deux états, l'état éteint et l'état acquisition.

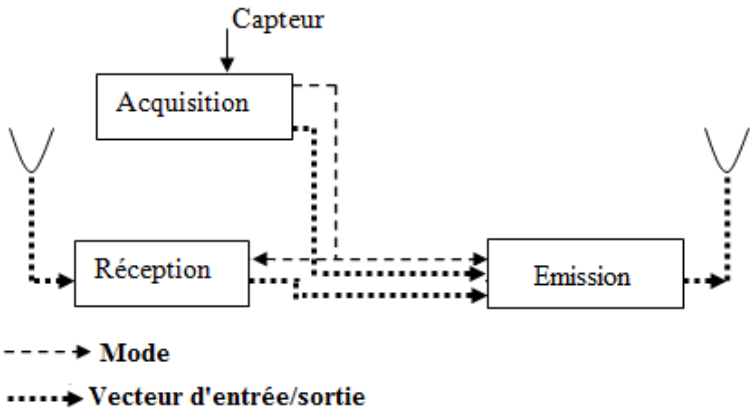

FIGURE 3.5 – Déscription d'un exemple de modèle simple d'un noeud

Le tableau 3.2 montre la classification des différentes fonctions utilisées par le modèle 3.5.

Fonction	Mode d'activation	Mode de gestion de données
Acquisition	Evènement déterministe	Traitement de donnée
Réception	Evènement déterministe/Etat d'entrée	Transfert de données
Emission	Etat d'entrée	Transfert de données

TABLE 3.2 – Classification des fonctions utilisées

Si on considère plusieurs noeuds capteurs dans le modèle, le vecteur de sortie de la fonction émission du noeud émetteur sera connecté au vecteur d'entrée de la fonction réception de l'autre noeud récepteur par l'intermédiaire du modèle de canal de communication. Ainsi, la fonction réception comprend deux modes d'activation. Le premier mode d'activation est l'activation par un évènement déterministe permettant de changer l'état veille la fonction réception en état écoute selon son calendrier de réveil. Le deuxième mode d'activation est l'activation par état d'entrée. Ce deuxième

mode d'activation permet de changer l'état écoute de la fonction en état réception lors de l'arrivée de donnée.

Concernant la fonction acquisition, nous avons considéré comme un exemple que le changement de l'état éteint en état acquisition est activé par un évènement déterministe selon la période d'acquisition.

Cependant, le mode d'activation de la fonction émission est l'activation par l'état d'entrée. Tout d'abord, il faut savoir que selon leur calendrier de réveil, les fonctions réception et acquisition peuvent déclencher une demande de changement d'état de la fonction émission, notament sur le changement de l'état veille en état idle. Ensuite, l'état idle de la fonction émission change en état émission lorsque les données sont arrivées à l'entrée de celle-ci.

Le code SystemC correspondant à ce modèle peut donc être structuré selon l'organisation présentée par la figure 3.6. On y retrouve les modules qui correspondent aux trois fonctions, et le thread nécessaire au calcul de la puissance globale consommée par le noeud.

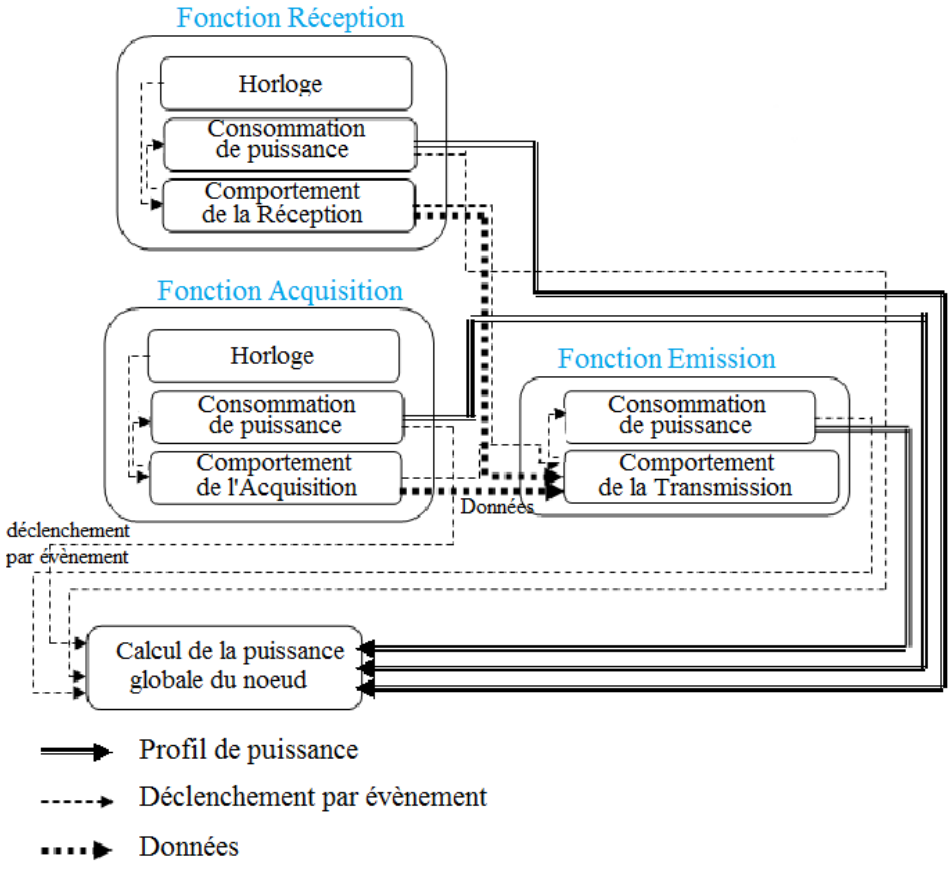

FIGURE 3.6 – Exemple de modèle d'un noeud simple avec la description SystemC

Toutes les fonctions sont interconnectées via les vecteurs d'entrée/sortie et sont synchronisées par les déclencheurs d'évènements ou les modes d'entrée/sortie pour former le modèle de noeud capteur.

D'après ce qui est mentionné précédement, chaque fonction possède au moins deux threads relatifs aux deux types de résultats à la sortie (les profils de puissance et les vecteurs de sorties). Le thread comportement caractérise le comportement de la fonction suivant l'entrée mode et le vecteur d'entrée. Il comporte également les différents états internes de la fonction et les conditions de changement d'un état interne à un autre. Chaque état interne correspond à un niveau de puissance. Le thread consommation de puissance fournit en sortie l'évolution de la puissance consommée par la fonction au cours du temps, ce qui donne le profil de puissance. Ainsi, ce dernier dépend de l'état interne en cours et de l'entrée mode. Le thread horloge permet de réveiller le thread comportement, c'est une activation interne de la fonction.

3.4.2 Modélisation d'un noeud capteur mobile

Un second exemple considéré ici concerne la mobilité du noeud. Pour cela, la fonction localisation est ajoutée dans le noeud pour permettre au noeud mobile de se localiser dans le réseau et de découvrir ses noeuds voisins (à un saut). Cette fonction intègre le mécanisme de routage géographique pour router les données jusqu'au puits. La fonction localisation reçoit les coordonnées des noeuds voisins et leur transmet les siennes. Une décision sera prise pour choisir le noeud voisin qui fera le prochain relais. Cette fonction localisation permet aussi de synchroniser tous les noeuds dans le réseau puisque cette fonction est périodique. La figure 3.7 ci-dessous représente un exemple de modèle d'un cas simple de noeud mobile avec ses différentes fonctions.

Le modèle SystèmC correspondant à cette figure 3.7 est montré par la figure 3.8. On y retrouve quatre modules correspondant aux quatre fonctions du modèle : réception, émission, acquisition et localisation. Ce qui différencie ce noeud mobile par rapport à un noeud simple est cette fonction localisation.

L'exemple de modèle de noeud mobile montré sur la figure 3.8 a une fonction localisation, elle même composée de trois threads. Le thread horloge lui permet d'activer périodiquement certaines tâches dans le thread comportement. Les différents traitements dans ce thread comportement sont : la découverte des noeuds voisins, la localisation dans le réseau et le routage de données. Ainsi, cette fonction localisation sollicite les fonctions réception et émission pour l'échange de calendrier de réveil et les coordonnées de positionnement.

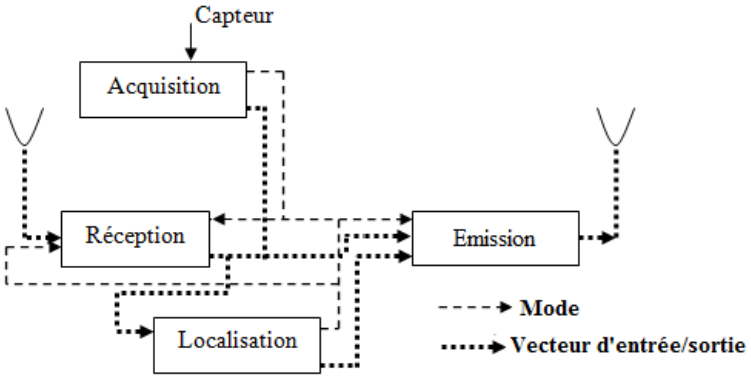

FIGURE 3.7 – Exemple d'un modèle de noeud mobile

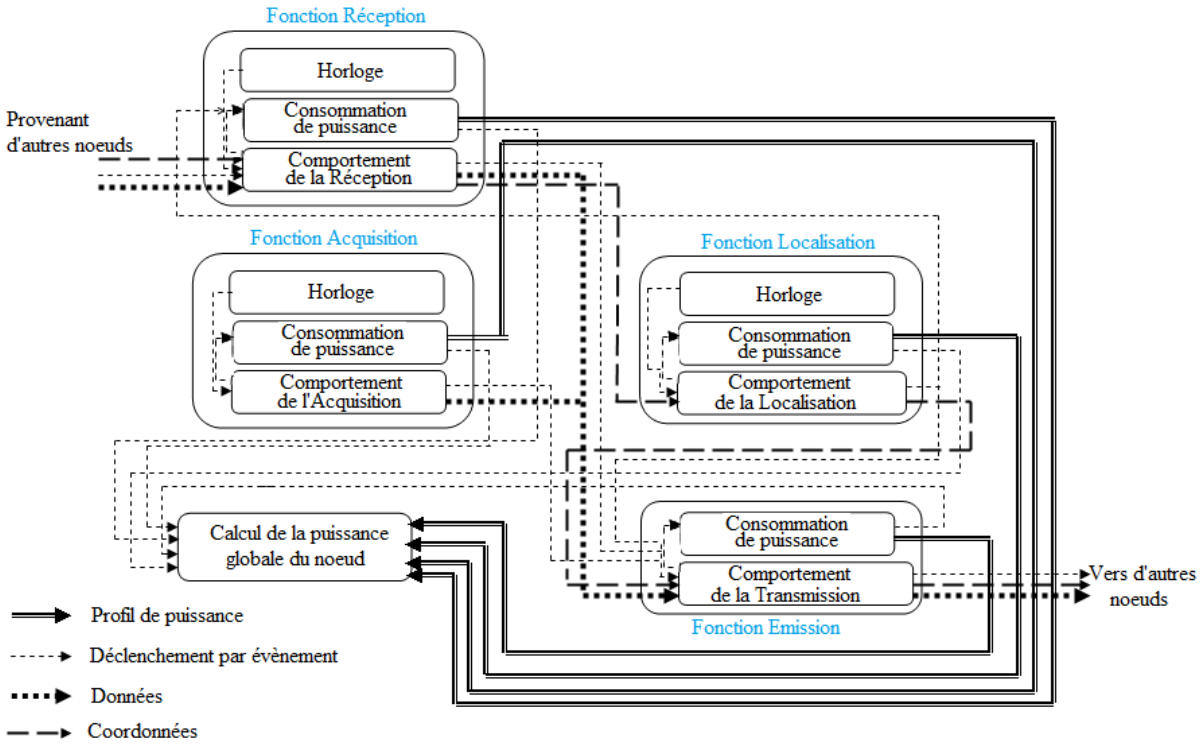

FIGURE 3.8 – Exemple de modèle d'un noeud mobile avec la description SystemC

3.4.3 Modélisation d'un noeud capteur intelligent

Une fonction de supervision est ajoutée dans le noeud pour le rendre intelligent en consommation d'énergie. L'objectif de cette fonction est de gérer en continue les autres fonctions qui composent le modèle de noeud. La gestion de ces autres fonctions dépend

de quelques paramètres d'entrée et l'objectif principal est d'optimiser la consommation d'énergie.

La fonction de supervision peut être classifiée comme une fonction active en permanence puisqu'elle gère en continue toutes les autres fontions.

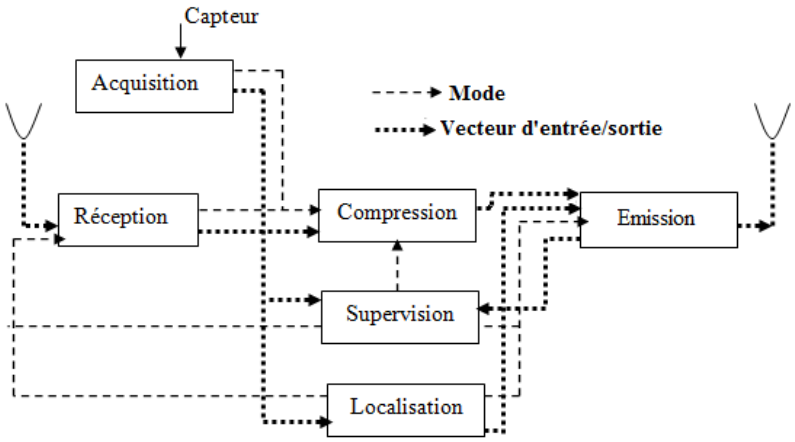

FIGURE 3.9 – Exemple d'un modèle de noeud intelligent

La décision prise par cette fonction de supervision peut être régie par les équations implementées à l'intérieur de celle-ci. Suivant les flux d'information inclus dans le vecteur d'entrée, la fonction de supervision prend des décisions dans le but de contrôler les autres fonctions en utilisant la sortie mode.

Dans l'exemple présenté par la figure 3.9, les fonctions réception et émission fournissent des informations à la fonction de supervision. Cette dernière peut alors prendre des décisions (locales) différentes suivant ses paramètres d'entrée et la décision prise aura un impact conséquent sur le comportement des autres fonctions. La fonction de supervision peut donc contrôler le changement des états internes de la fonction réception, émission et compression suivant les vecteurs d'entrée venant des fonctions réception et émission.

Le modèle SystemC illustré par la figure 3.10 suivante est basé sur le modèle de noeud mobile intelligent présenté par la figure 3.9. La figure 3.10 montre que le modèle de noeud mobile intelligent est composé de cinq modules correpondant aux cinq fonctions. Par rapport à l'exemple de noeud simple, la fonction localisation est très utile pour la mobilité d'un noeud et la fonction supervision lui permet d'être intelligente en gestion de consommation d'énergie. La fonction la plus intéressante dans ce modèle est alors la fonction supervision.

Le rôle de cette fonction supervision est de gérer les autres fonctions dans le mo-

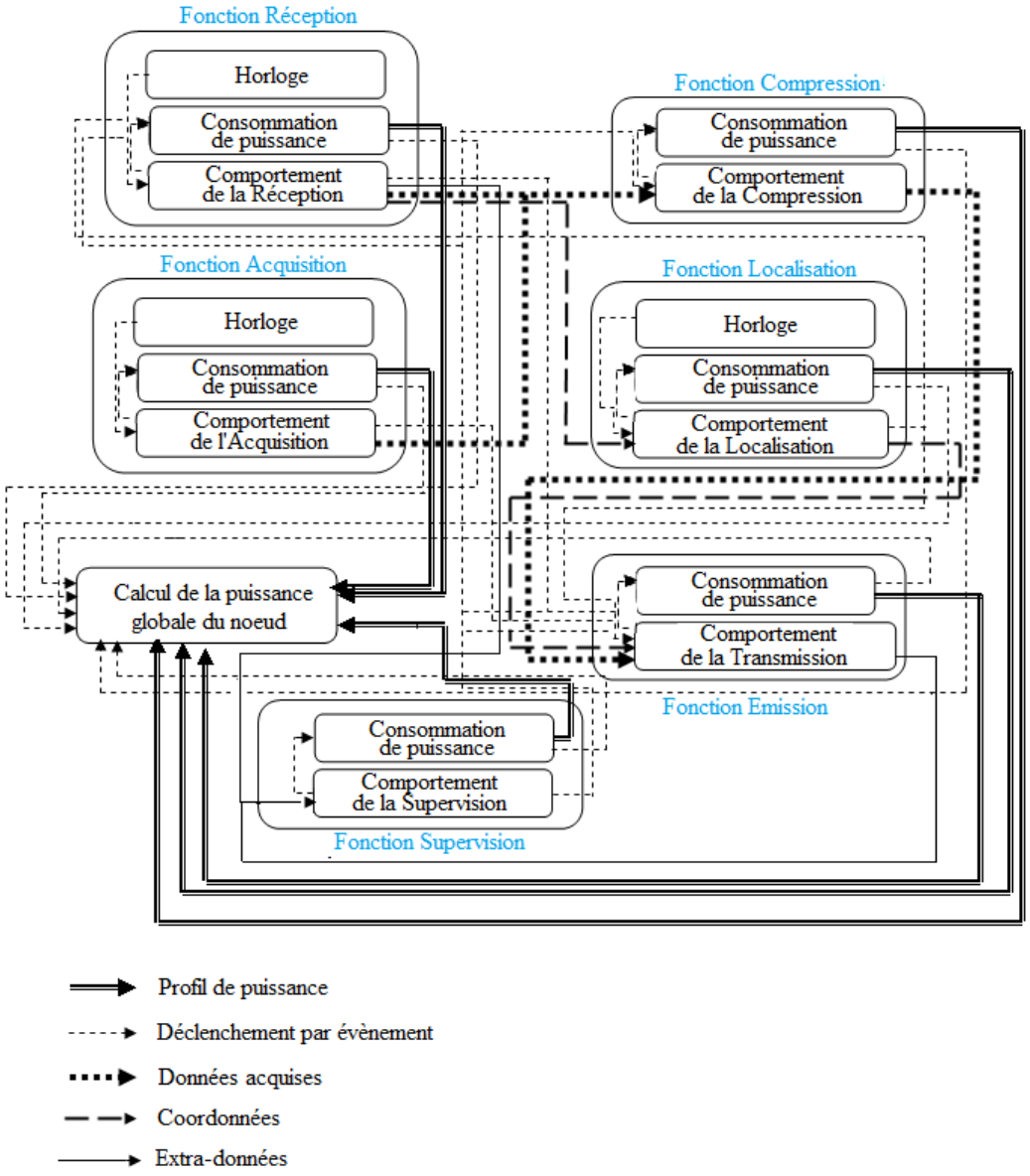

FIGURE 3.10 – Structure SystemC du modèle d'un noeud mobile intelligent

dèle de noeud, mais dans l'exemple que nous avons pris et illustré sur la figure 3.10, la fonction supervision gère surtout les fonctions réception, émission et compression. Comme on le voit sur la figure 3.10, la fonction supervision est composée de deux threads, elle n'est pas activée par une horloge interne ni par une activation externe puisqu'elle est classifée comme une fonction active en permanence. Son thread comportement permet d'intégrer les différentes conditions afin de calculer les compromis selon la situation pour économiser l'énergie.

Dans l'exemple de scénario que nous avons décrit, la fonction supervision reçoit des informations venant des fonctions réception et émission sur l'évolution de la situation dans le réseau (positionnement, topologie, calendrier de réveil). En tenant compte de ces variations de situation, la fonction supervision met à jour ses décisions (locales) à chaque changement de situation. Comme nous l'avons mentionné auparavant, la fonction supervision peut prendre deux types de décisions : la décision de faire la compression et la décision de faire basculer les fonctions réception et émission en mode veille. Comme le montre la figure 3.10, la fonction supervision peut contrôler les fonctions réception, émission et compression via son lien mode de sortie.

Dans notre cas, on considère que la fonction acquisition pourrait être déclenchée par un évènement déterministe (activée périodiquement).

3.4.4 Création d'un modèle de noeud

De ces quelques exemples de modèles SystemC mentionnés précédemment apparaissent clairement les règles à suivre pour construire un modèle suivant les principes utilisés dans notre étude.

Dans un premier temps, il faut déterminer les fonctions nécessaires pour l'application. A partir de cette information, il est possible de construire un premier niveau du modèle SystemC puisqu'à chaque fonction il s'agit d'associer une classe héritant de la classe sc_module de SystemC. Ensuite, il faut déterminer toutes les relations à implanter entre ces sc_modules. Enfin, pour terminer ce premier niveau, il faut déterminer le type d'activation de chaque fonction (voir paragraph 3.2.1) afin de déterminer le nombre de sc_thread à déclarer et instancier dans chaque classe.

Le deuxième niveau concerne l'écriture du modèle interne de chaque fonction. Chaque fonction possède au moins une variable interne qui représente l'état de la fonction et une relation interne de type sc_event qui permet de notifier tout changement de valeur de l'état de la fonction. Le code du thread qui détermine la consommation de la fonction est relativement simple à écrire puisqu'il a pour simple objectif d'associer un niveau de consommation à un état. Le code associé au thread de comportement de la fonction (voir figure 3.2) est quant à lui plus complexe à écrire puisqu'il doit intégrer une grande partie des algorithmes et protocoles qui permettent de déterminer les valeurs de sortie des fonctions et ceux nécessaires à la détermination de la succession des états de la fonction. Enfin pour les fonctions dont le mode d'activation est de type "évènement déterministe", il faut écrire un thread supplémentaire pour réveiller le thread comportement, cela implique aussi la déclaration d'un élément interne au sc_module

de type sc_event.

Le troisième niveau consiste à ajouter au modèle les éléments nécessaires à la détermination de la puissance consommée. Pour cela, il faut déclarer une variable commune de consommation de chaque fonction, variable modifiée par le thread de consommation de la fonction concernée. Il faut aussi déclarer une classe qui hérite de sc_module et qui ne possède qu'un seul sc_thread effectuant le calcul de la consommation globale du noeud à partir de la consommation de chacune des fonctions.

Enfin le quatrième et dernier niveau consiste à écrire une classe associée au noeud et établissant les relations entre les différentes fonctions.

3.4.5 Modélisation du canal de communication

Dans cette section, le canal de communication est modélisé de façon simple. L'exemple de modèle de canal que nous présentons ici repose sur la distance entre les noeuds les uns par rapport aux autres. Cette distance peut ensuite faciliter le choix du chemin optimale faite par le protocole de routage. La distance entre le noeud émetteur et le noeud récepteur permet aussi de régler la puissance d'émission utilisée pour assurer une bonne transmission de données. Notre étude n'entre pas en détails sur la nature du canal, le type d'attenuation utilisé, la perte de données, le type de modulation, les réflexions, les obstacles, les phénomènes inhérents au médium sans fil, le temps de propagation, etc. Le modèle basique de canal présenté ici estime périodiquement la distance entre tous les noeuds qui peuvent se déplacer dans le réseau.

La figure 3.11 représente un exemple simple de réseau capteur mobile, comportant les modèles de noeud ainsi que le modèle simple du canal de communication. Ce modèle du canal se base sur une matrice présentée par la table 3.3 et composée par les distances qui séparent les noeuds. Cette matrice est mise à jour périodiquement suivant le déplacement des noeuds dans le réseau. Chaque noeud peut ensuite exploiter cette matrice pour connaître la distance qui le sépare de ses correspondants afin d'adapter sa puissance d'émission pour avoir une bonne transmission de donnée.

Dans cet exemple, on considère que le lien entre le noeud émetteur et le noeud récepteur est symétrique.

Le modèle de canal peut être associé au modèle de noeud présenté dans ce chapitre pour produire un modèle du réseau. Le modèle ainsi construit peut être utilisé pour analyser l'influence du comportement du réseau entier sur la consommation dans chaque noeud.

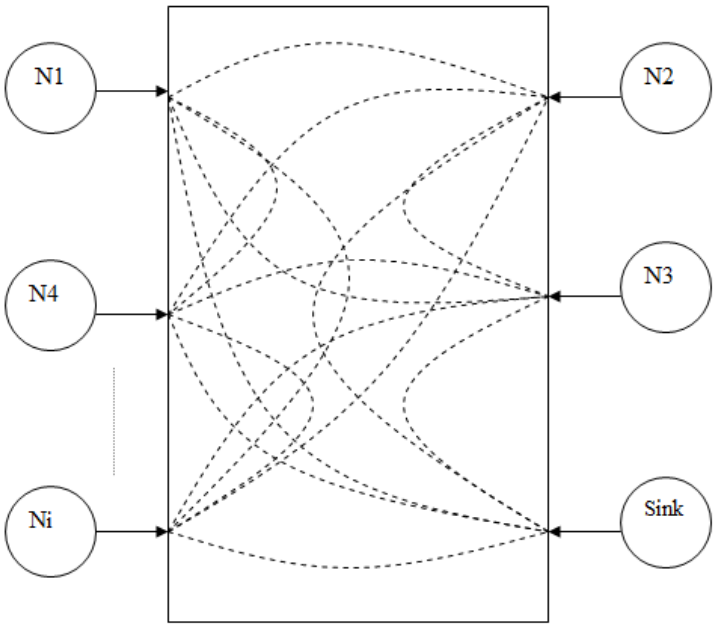

FIGURE 3.11 – Représentation simple du canal de communication

[m]	N1	N2	N3	N4	Ni	Sink
N1	X	30	70	22	54	25
N2	X	X	10	75	23	40
N3	X	X	X	50	18	34
N4	X	X	X	X	65	44
Ni	X	X	X	X	X	20
Sink	X	X	X	X	X	X

TABLE 3.3 – Exemple de matrice de distance entre les noeuds

3.5 Conclusion

En conclusion de ce chapitre, on a pu définir ce que l'on entend par modélisation par fonctions, nous avons décrit quelques variantes de fonctions ainsi que ses caractéristiques, ensuite nous avons proposé une classification de ces fonctions selon nos critères (mode d'activation et mode de gestion de données).

En tenant compte de notre objectif, nous avons modélisé la fonction de façon à pouvoir fournir les résultats que l'on veut obtenir, notamment sur l'évolution au cours du temps du profil de puissance des noeuds capteurs. Ainsi, nous avons modélisé le noeud avec cette approche par fonctions et un noeud capteur est désormais considéré comme un ensemble de fonctions. En effet, si l'on veut modéliser un type de noeud capteur,

on définit d'abord les différents comportements qu'il peut avoir, puis c'est à partir de ces comportements que l'on identifie les différentes fonctions que l'on veut ajouter ou retirer dans le modèle de noeud. Dans ce chapitre, nous avons présenté quelques types de modèles de noeuds capteurs (noeud mobile, noeud intelligent, etc). Et à partir de ces quelques modèles de base, il est plus facile d'ajouter ou d'enlever des fonctions selon le type de noeud que l'on veut obtenir. Nous avons décrit dans ce chapitre les besoins de notre simulation et SystemC est un candidat idéal puisque ses caractéristiques satisfont vraiment les besoins de nos modèles. On a modélisé les noeuds capteurs ainsi que le canal de communication selon quelques scénarios de simulations établis.

Nous avons pu voir dans ce chapitre que la modélisation par fonctions permet de créer un modèle de noeud et un modèle de réseau tout entier. A partir des exemples de modèles présentés dans ce chapitre, nous pouvons conclure que l'approche par fonctions permet l'adaptation du modèle à des situations dynamiques (mobilité des noeuds, comportement intelligent des noeuds, etc).

Dans le chapitre suivant, nous présentons des résultats de simulations de consommations obtenues par notre approche et nos modèles de noeuds et de réseaux. Nous proposons aussi de comparer ces résultats de simulations avec quelques résultats de mesures expérimentales.

4

Simulations et résultats

Dans ce chapitre, nous avons établi comme exemples quelques scénarios sur les réseaux de capteurs sans fil en se basant sur les modèles précédemment présentés. Nous avons vu dans le chapitre précedent que les noeuds capteurs et le réseau tout entier sont modélisés avec SystemC. Ainsi, nous avons utilisé Visual Studio 2010 de Microsoft pour simuler tous les modèles de SystemC. Une approche semblable pourrait être basée sur un autre environnement de compilation tel que GCC (GNU Compiler Collection) par exemple. Ensuite, nous avons utilisé Matlab pour éditer les résultats des simulations. L'outil GNU plot peut être aussi une altérnative pour éditer les résultats de simulations.

L'objectif de ce chapitre est de mettre en valeur les caractéristiques importantes de la modélisation par une approche par fonctions. Le chapitre est organisé de la manière suivante. Le premier exemple de simulation consiste à comparer l'énergie consommée par un noeud utilisant un protocole à acquittement (semblable à l'un des caractéristique de TCP) ou un protocole qui n'utilise pas d'acquittement (semblable à l'un des caractéristique de UDP). L'objectif ici est de montrer qu'avec l'approche par fonctions, le lien entre le comportement du noeud et la consommation d'énergie est facilement identifiable. On a ajouté aussi dans ce modèle le mécanisme CSMA/CA en mode beacon (envoi régulier d'une balise de synchronisation). Les résultats de simulations permettent de donner des informations sur la consommation d'énergie liée à l'utilisation de ces protocoles et de ce mécanisme.

Le deuxième exemple de simulation permet de simuler un réseau de capteurs avec des noeuds mobiles. L'objectif de ce deuxième exemple est de montrer l'influence de la mobilité et de l'activité des noeuds sur la consommation d'énergie.

L'exemple de simulation suivant est basé sur le modèle simple de noeud auquel on a ajouté une consommation liée à la transition d'état, c'est-à-dire que l'on considère dans le modèle, la durée de transition et la puissance consommée par la transition d'un état vers un autre dans une fonction. La modélisation de noeuds utilisant ces transitions d'état a pour objectifs de montrer que la modélisation adoptant l'approche par fonctions permet de bien visualiser, avec différents niveaux de précision, l'évolution de la puissance consommée au cours du temps.

Le dernier exemple de simulation considère les noeuds capteurs "intelligents" en terme de gestion d'énergie, en vue de démontrer que l'approche par fonctions permet de montrer l'interêt d'une fonction d'optimisation de l'utilisation de l'énergie dans un noeud. L'approche en effet peut modéliser un algorithme d'optimisation d'énergie en gérant simultanément l'état de plusieurs fonctions.

La dernière section de ce chapitre est consacrée à la validation de notre approche par des mesures réelles et une comparaison avec des résultats de simulations. Des mesures de consommation d'énergie sur un noeud capteur ont été effectuées. Les résultats de mesures sont ensuite comparés avec des résultats de simulation produits par le modèle de noeud, cela afin de valider la modélisation par l'approche par fonctions.

4.1 Simulation d'un réseau de capteurs classique

Dans cette simulation, on compare l'impact en consommation d'énergie d'un noeud zigbee n'utilisant pas le protocole à acquittement avec celui d'un noeud utilisant le protocole à acquittement et le mécanisme de réservation. Le but ici est de montrer que la modélisation adoptant l'approche par fonctions peut fournir le lien entre les différents protocoles/mécanismes intégrés dans le noeud avec la consommation d'énergie. En effet, on peut estimer l'impact en énergie des différents protocoles et mécanismes implémentés dans le noeud.

Toutefois, les chiffres donnés dans les résultats ne sont liés à aucune implémentation ou mesure, et sont donnés simplement à titre d'exemple.

4.1.1 Simulation des noeuds utilisant le protocole à acquittement

Cette simulation se base sur le modèle de noeud simple présenté précédemment. Le protocole à acquittement est souvent utilisé pour assurer l'intégrité des données transmises. Ainsi, il peut y avoir une retransmission de données en cas d'erreur de transmission. Toutefois, ce cas n'est pas considéré ici. La transmission se fait en mode point à point. Il est donc nécessaire d'échanger des trames de contrôle pour accéder au canal de communication, pour réserver le lien entre le noeud émetteur et le noeud récepteur et pour éviter les noeuds terminals cachés. Avant d'envoyer les données, le noeud émetteur envoie d'abord une trame RTS au noeud récepteur qui lui répond par une trame CTS pour accepter ou refuser le transfert de données. Si le CTS n'est pas reçu, le mécanisme estime que le RTS a subi une collision. Les trames RTS et CTS contiennent des informations sur la taille des données à transmettre. Ainsi, les noeuds voisins du côté noeud émetteur qui ne sont pas concernés par le transfert de données et qui reçoivent la trame RTS peuvent estimer, grâce à celle-ci, la durée totale nécessaire pour le transfert de données. De même, ceux du côté noeud récepteur qui ne sont pas concernés par le transfert et qui recoivent la trame CTS peuvent estimer également cette durée nécessaire à l'envoie de données. Ainsi, ces noeuds voisins ne doivent pas utiliser le canal pendant cette durée et doivent se mettre en veille pour attendre la fin du transfert de données.

Afin de pouvoir interpréter les résultats, et à titre d'exemple, nous avons pris arbitrairement les consommations indiquées dans la table 4.1 pour chacun des états.

Fonction	Etat	Puissance consommée
Emission	Emission	34 mW
	Idle	22.5 mW
	Veille	0.1 mW
Réception	Réception	37.5 mW
	Ecoute	22.5 mW
	Veille	0.1 mW
Acquisition	Acquisition	31 mW
	Veille	0.1 mW

TABLE 4.1 – Paramètres de simulation

La figure 4.1 montre l'évolution de la consommation d'un noeud utilisant le protocole à acquittement pendant une période de relais de données (partie gauche de la figure) et une période d'acquisition suivie d'une émission de données (partie droite). Dans l'exemple de scénario que nous avons défini, les noeuds utilisent le mécanisme

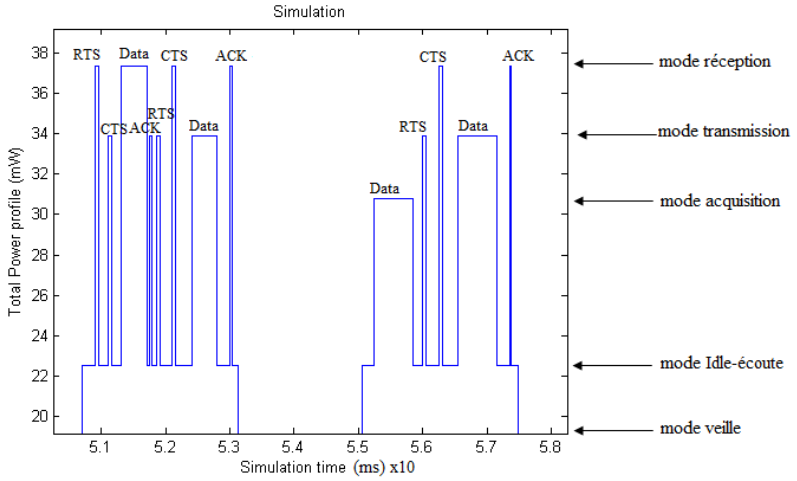

FIGURE 4.1 – Deux périodes d'activités d'un noeud utilisant le protocole TCP

de réservation. Ce qui justifie l'utilisation des trames de contrôle RTS/CTS. On voit bien sur la figure 4.1 l'impact de cet échange de trames de contrôle sur la consommation d'énergie. Avant la réception de donnée, le noeud récepteur reçoit d'abord le trame RTS venu du noeud émetteur, puis il envoie le trame CTS pour dire qu'il est prêt à recevoir, puis les données sont reçues et puisque le protocole à acquittement assure l'intégrité de données il envoie un trame d'aquittement ACK. Pour faire suivre les données reçues, le noeud envoie vers le prochain noeud le trame RTS au noeud destinataire, puis il reçoit le trame CTS de celui-ci, ensuite il envoie les données et attend la réception de l'acquittement ACK. A chaque transmission de données, chaque noeud doit effectuer ces étapes pour éviter les collisions et pour réserver le lien entre le noeud émetteur et le noeud récepteur. Dans la deuxième période (partie droite de la figure 4.1), on remarque que le noeud fait la capture de données puis entame les échanges de trames de contrôle avant l'émission de données acquises.

On peut noter que la modélisation par l'approche fonction permet d'avoir, comme résultat de simulation, l'évolution au cours du temps du profil de puissance du noeud. De plus, elle permet aussi de bien distinguer l'évolution du profil de puissance de chaque fonction dans le noeud. Ceci est intéressant lorsque l'on veut étudier une fonction particulière, en examinant le temps d'éxecution de la fonction ou la puissance consommée ou le temps de traitement de la fonction. A titre d'exemple, la figure 4.2 présente le profil de consommation individuelle des fonctions émission et réception pour l'exemple ci-dessus.

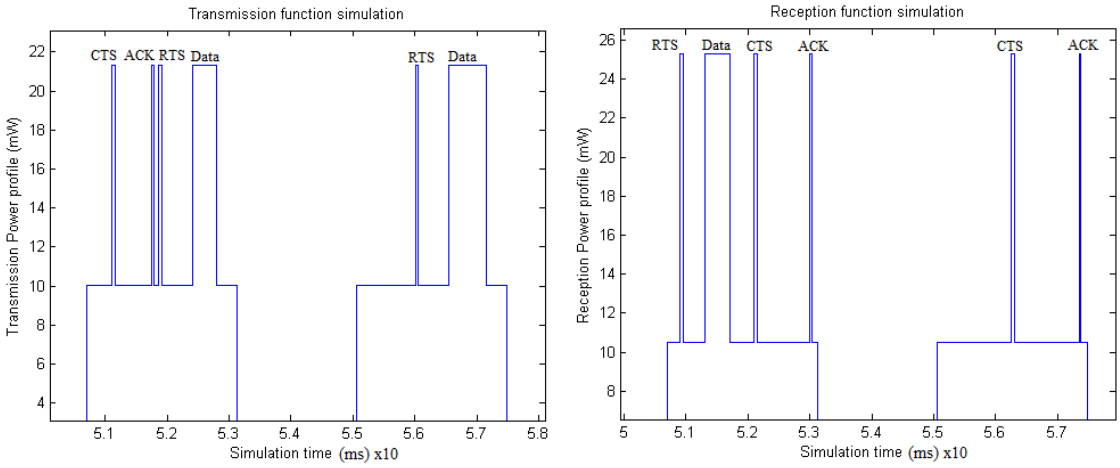

FIGURE 4.2 – Fonctions émission et réception

4.1.2 Simulation des noeuds utilisant le protocole sans acquittement

Cette simulation se base aussi sur le modèle de noeud simple présenté précédemment. Dans le cas de cette simulation, on a implémenté dans le modèle de noeud le protocole sans acquittement (de type UDP). Ce type de protocole est souvent utilisé pour transmettre des données de taille importante sans aucune validation de la transmission. Les applications utilisant ce protocole exigent souvent une transmission immédiate. En plus, les trames utilisés par ce type de protocole sont prioritaires dans le cas où les deux types de protocoles (avec acquittement et sans acquittement) cohabitent. Par exemple, les types de données qui nécessitent ce type de protocole sont : les données de types vidéos, images, les données qui ne nécessitent pas une exigence sur l'intégrité des données.

En se basant sur les noeuds capteurs simples (voir figure 3.5), le modèle de noeud est composé de trois fonctions de base : la fonction réception, la fonction acquisition et la fonction émission. La simulation se base aussi sur les niveaux de consommation des fonctions indiquées par le tableau 4.1. La figure 4.3 montre l'évolution du profil de puissance du modèle de noeud utilisant le protocole sans acquittement. Sur la première période (partie gauche de la figure), le noeud joue le rôle de relais. Il reçoit les données venant de l'un de ses noeuds voisins puis il retransmet ces données vers un autre noeud voisin. La deuxième période sur la figure (partie droite) montre une période d'acquisition de données suivie d'une transmission des données acquises.

On remarque qu'il n'y a pas d'échange de trames de contrôles entre les noeuds

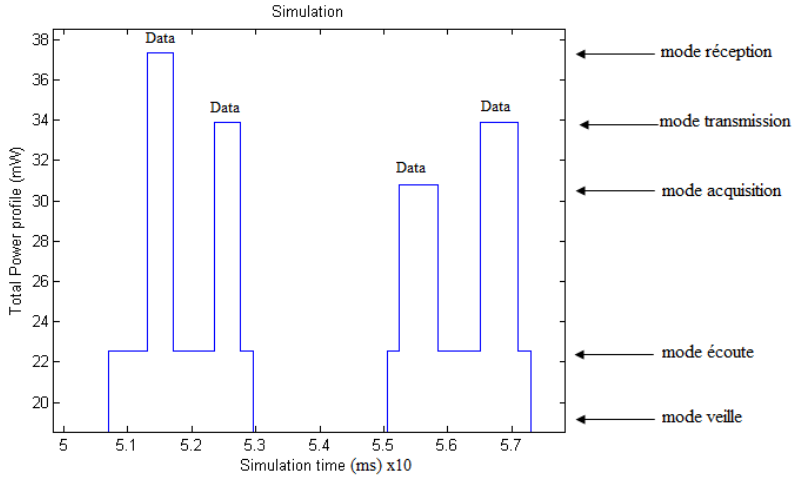

FIGURE 4.3 – Noeud utilisant le protocole sans acquittement

communicants comme dans le cas du protocole avec acquittement et avec le mécanisme de réservation puisque la transmission est en mode diffusion ou broadcast, et le protocole sans acquittement n'assure pas l'intégrité des données.

Dans cet exemple, l'énergie consommée par le noeud en émission et réception dépend uniquement de la taille de données traitées. Dans la figure 4.3, on peut suivre la variation du profil de puissance du noeud selon l'évolution du temps, ce qui n'est pas le cas des résultats issus des autres simulateurs de réseaux comme NS2 natif.

4.1.3 Exploitation des résultats de simulations

Sur un même scénario et une même configuration du profil de consommation des fonctions, la figure 4.4 montre l'évolution de la réserve d'énergie d'un noeud utilisant le protocole sans acquittement et celui d'un noeud utilisant le protocole à acquittement suite aux simulations précédentes.

On peut voir sur la figure 4.4 que le noeud utilisant le protocole à acquittement et le mécanisme de réservation de canal consomme plus d'énergie que le noeud utilisant le protocole sans acquittement. On peut constater que le bilan d'énergie, après 500 heures, indique que le noeud utilisant le protocole à acquittement a une consommation d'énergie de plus de 3% supérieure à celle d'un noeud utilisant le protocole sans acquittement. Il est à noter que cette estimation ne considère pas les retransmissions et les collisions de messages. Le protocole sans acquittement transmet l'intégralité des données en mode broadcast ou diffusion et ne se soucie pas de la réception des données tandis que le protocole à acquittement et le mécanisme de réservation de canal

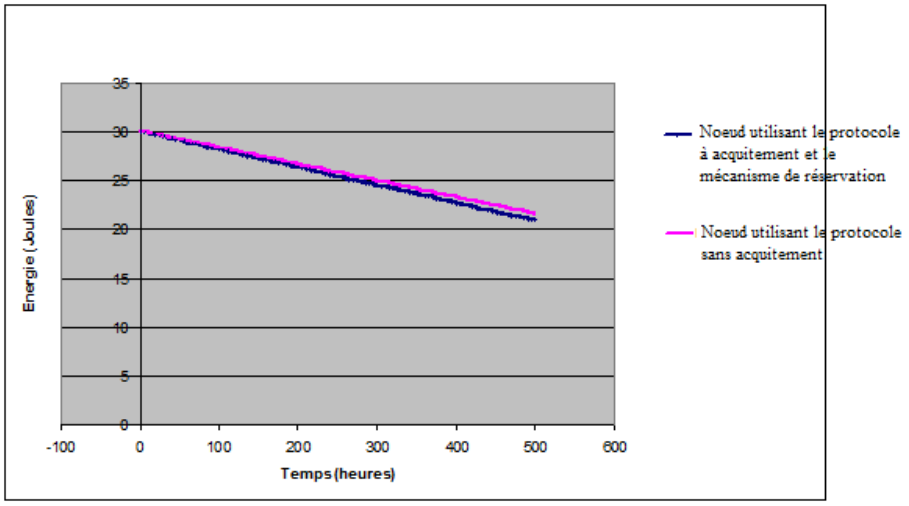

FIGURE 4.4 – Comparaison des deux types de protocoles sur le plan énergétique

assure l'intégrité de donnée et la réserve de canal. Ce qui implique l'utilisation des trames de contrôle qui ont un impact important sur la consommation d'énergie. En cas d'erreur, le protocole à acquittement fait appel à la retransmission de données pour assurer l'intégrité des données, d'où l'utilisation d'un trame d'acquittement qui est aussi une consommation de plus par rapport à celle du protocole sans acquittement dans le cas d'un canal de communication parfait comme dans notre exemple. Le protocole à acquittement utilise aussi un mécanisme de découpage de données en segments pour être ensuite transmis comme le montre la figure 4.5. Dans cette figure 4.5, les données reçues et les données acquises sont mémorisées et assemblées pour être ensuite découpées et transmises (partie droite de la figure). Chaque transmission de données nécessite un acquittement, ce qui induit une consommation plus forte par rapport au noeud utilisant le protocole sans acquittement.

Maintenant, prenons un exemple de topologie de réseau simple fixe, montré dans la figure 4.6, pour comparer les types de résultats de simulations que l'on peut obtenir avec nos modèles et avec ceux de NS2. Les résultats fournis par NS2 sont présentés par la figure 4.7 sous forme d'énergie restante au cours du temps. On peut remarquer dans cette figure 4.7 que, selon le temps de simulation, la réserve d'énergie du noeud 6 diminue plus vite que les autres noeuds puisqu'il joue aussi le rôle de relais des noeuds 1, 2 et 5. Par conséquent, il consomme plus d'énergie. On peut observer aussi que le noeud 2 consomme plus d'énergie que le noeud 3 puisque le noeud 2 retransmet aussi les données venant du noeud 4, alors que le noeud 3 ne fait que transmettre ses données capturées.

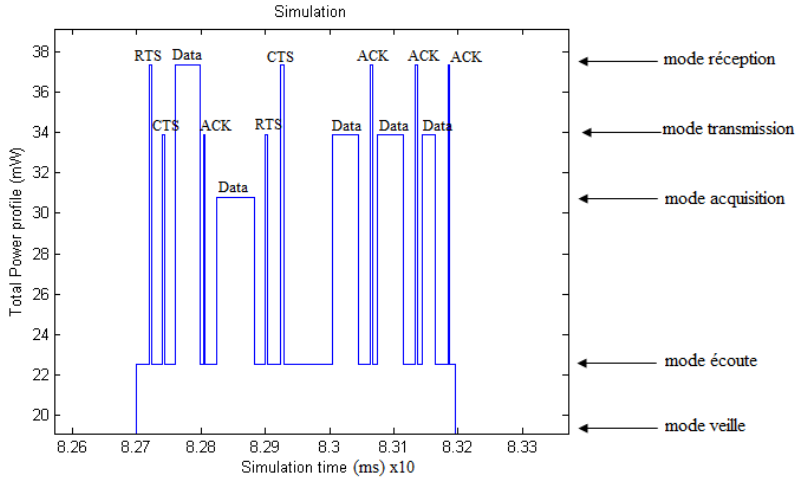

FIGURE 4.5 – Ségmentation des données transmises par TCP

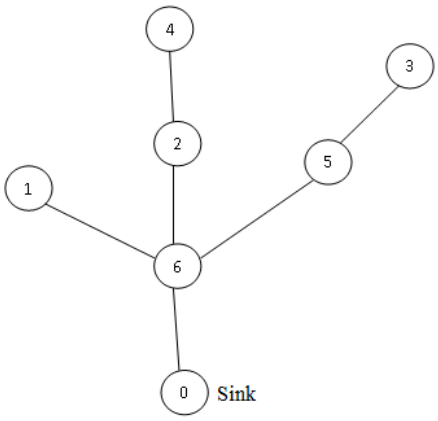

FIGURE 4.6 – Exemple de topologie de réseau

La figure 4.8 montre une vue globale du profil de puissance consommée par le noeud 6 issue du simulateur développé. On peut remarquer dans la figure 4.8 que le noeud 6 a une consommation régulière puisque le réseau est fixe et que le noeud 6 a toujours 5 noeuds en amont.

Avec la modélisation adoptant l'approche par fonctions, les résultats de simulations peuvent être présentés sous forme d'évolution au cours du temps de l'énergie restante dans le noeud (figure 4.4). Ce type de résultat de simulation est similaire aux résultats de consommation de NS2 (figure 4.7). Cependant, la modélisation par fonctions permet également de fournir comme résultat de simulation l'évolution au cours du temps du profil de puissance des fonctions et du noeud tout entier. En outre, la modélisation par l'approche par fonctions permet de fournir plus d'informations sur l'aspect énergie du

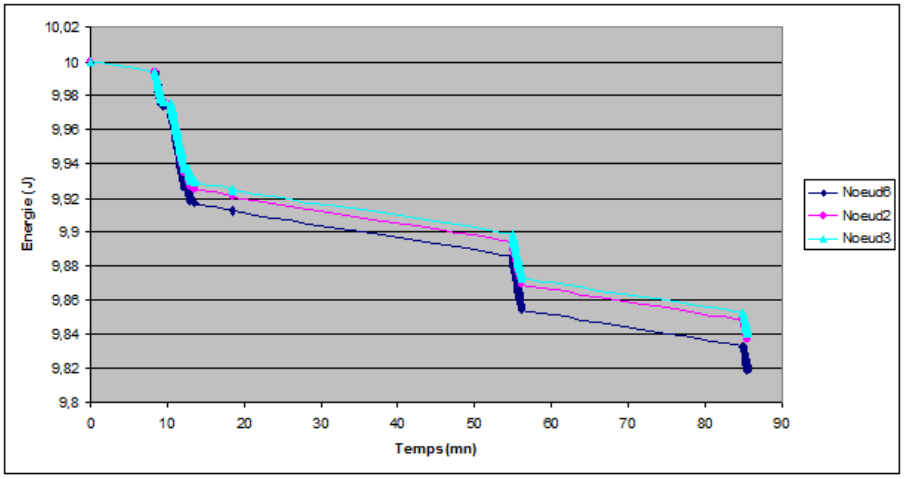

FIGURE 4.7 – Résultats de simulation avec NS2

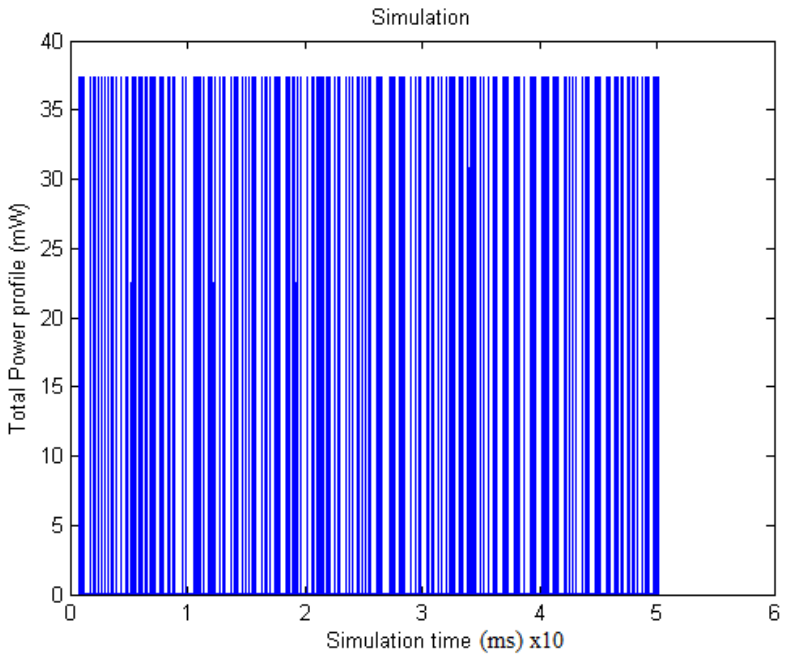

FIGURE 4.8 – Profil de puissance du noeud 6 dans un réseau fixe

noeud capteur (figure 4.5).

4.2 Simulation des noeuds capteurs mobile

L'objectif de ce deuxième exemple de simulation est de montrer que le modèle proposé permet de constater l'impact de la mobilité des noeuds sur leur consommation d'éner-

gie. Il s'agit aussi de montrer l'impact en énergie consommée lorsque l'on ajoute une fonction (fonction localisation) dans un noeud.

On reprend le modèle de noeud mobile présenté par la figure 3.7 et dont le modèle SystemC est présenté par la figure 3.8 dans le chapitre précédent.

Dans notre exemple, on a implémenté dans chaque noeud le mécanisme CSMA/CA en mode cascade de Beacon, on a utilisé le protocole à acquittement et le routage géographique (nombre de sauts par rapport au puits). Ensuite, nous avons simulé le déplacement des noeuds provoquant ainsi (voir figure 4.9) l'évolution de la topologie du réseau.

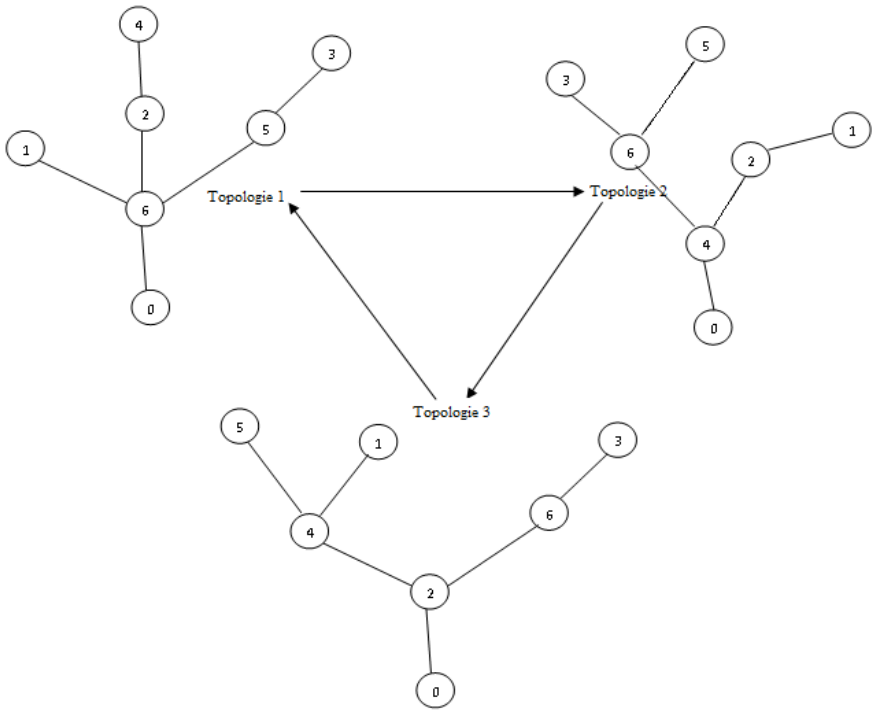

FIGURE 4.9 – Evolution de l'interconnection entre les noeuds

Pendant cette évolution de la topologie, à chaque connection des noeuds, ces derniers doivent d'abord se localiser dans le réseau. Puis, chaque noeud doit suivre les étapes de découverte des noeuds voisins en échangeant ses coordonnées et son calendrier de réveil avec ses noeuds voisins. Après ces différentes étapes, l'algorithme de localisation intervient pour localiser chaque noeud dans le réseau et pour ensuite mettre à jour sa table de routage.

Les paramètres utilisés dans cet exemple sont présentés dans la table 4.2.

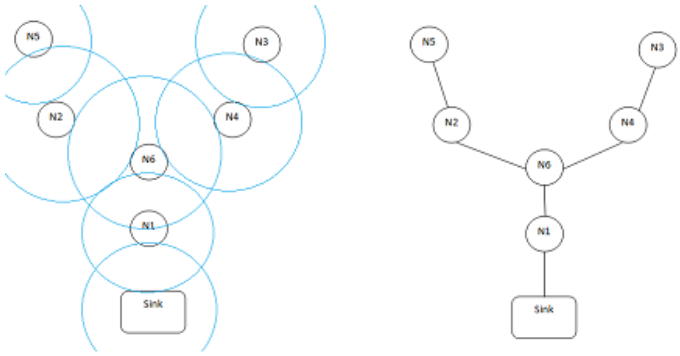

FIGURE 4.10 – La découverte des noeuds voisins

Fonction	Etat	Puissance consommée
Emission	Emission Idle Veille	32 mW 21.5 mW 0.01 mW
Réception	Réception Ecoute Veille	35 mW 21.5 mW 0.01 mW
Acquisition	Acquisition Veille	27 mW 0.01 mW

TABLE 4.2 – Paramètres de simulation pour le WSN mobile

La première période (partie gauche) de la figure 4.11 montre les échanges d'information (coordonnées et calendrier de réveil) du noeud 6 avec ses noeuds voisins. Il faut noter que les échanges de coordonnées et de calendrier de réveil sont en mode broadcast (de type UDP), ce qui signifie que les trames de contrôle ne sont pas utilisés. L'évolution de la découverte des noeuds voisins commence toujours à partir du coordinateur (sink). Ce dernier déclenche périodiquement les étapes de découverte des noeuds voisins, il envoie d'abord ses coordonnées à ses noeuds voisins et reçoit celles des voisins. Ainsi, les échanges de coordonnées et de calendrier de réveil se font en cascade à partir du coordinateur à travers le réseau jusqu'aux noeuds situés à l'extrémité du réseau.

Sur la première évolution de la topologie sur la figure 4.9 (topologie 1), on va examiner comme exemple l'évolution de la consommation de puissance du noeud 6, dans la figure 4.11, pendant la découverte de noeuds voisins. Tout d'abord, le noeud 6 reçoit les informations (coordonnées et calendrier de réveil) venant du coordinateur (noeud 0) puis il transmet une seule fois et en diffusion à son tour ses propres informations vers les noeud 1, 2 et 5 en amont. Ensuite, ces derniers envoient leurs informations à

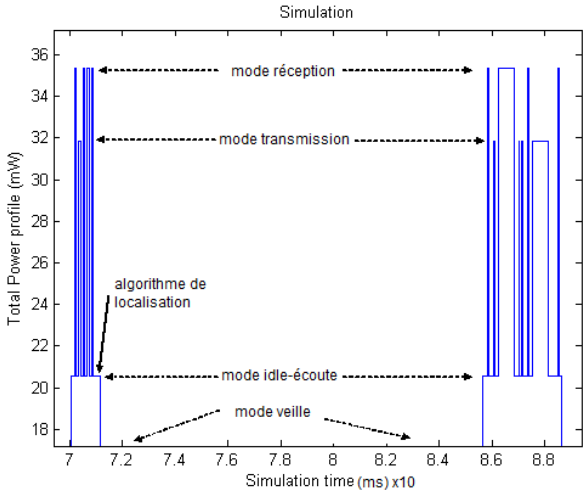

FIGURE 4.11 – Echange de coordonnées et relais de données (noeud 6, topologie 1)

destination du noeud 6. Après ces étapes, le noeud 6 calcule sa position en considérant les coordonnées de ses noeuds voisins et en utilisant son algorithme de localisation par la fonction localisation. Ensuite, il met à jour sa table de routage.

Généralement, ce mécanisme de découverte de noeuds voisins est implémenté dans les protocoles de routage. Par exemple, dans le protocole OLSR, il utilise un message "hello" contenant des informations sur l'état des liaisons et sur ses noeuds voisins. Par exemple, les simulateurs de réseaux tels que NS2 et OMNETT permettent de simuler ce type de protocole.

Dans l'exemple suivant, on a simulé le déplacement des noeuds selon l'évolution de la topologie présentée dans la figure 4.9. Le type de résultat que l'on peut obtenir est l'évolution du profil de puissance de chaque noeud dans le réseau. La figure 4.12 montre l'évolution du profil de puissance du noeud 6 relative à son comportement et à son déplacement dans le réseau. On identifie clairement les différentes topologies présentées par la figure 4.9. Dans le cas de la topologie 1, le noeud 6 est souvent sollicité pour retransmettre les messages venant des ses cinq noeuds en amont (noeud 1, 2, 3, 4 et 5). Il est beaucoup moins dans le cas des topologies 2 et 3.

En faisant un zoom sur cette figure 4.12, on peut observer les détails du comportement du noeud 6 dans ce résultat. La figure 4.11 montre les détails de l'évolution du profil de puissance du noeud 6 pendant la découverte de ses noeuds voisins et pendant une phase de relais.

La deuxième période (partie droite) dans la figure 4.11 montre une période de relais du noeud 6. Ce dernier fait les démarches nécessaires pour recevoir les données venant

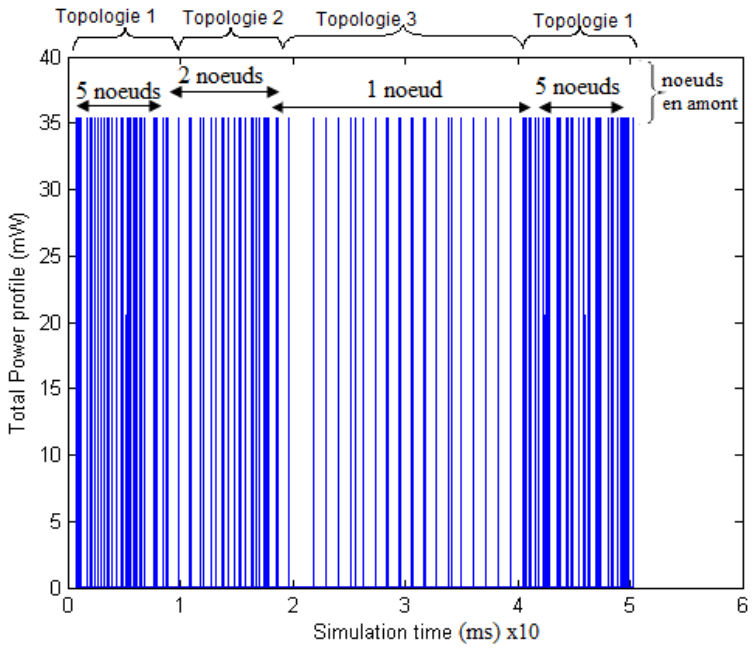

FIGURE 4.12 – Profil de puissance du noeud 6 pour différentes topologies

du noeud en amont et de les transmettre vers le noeud en aval. Avec son noeud voisin en amont, il reçoit un RTS, envoie un CTS, reçoit les données et envoie un acquittement. Puis, le noeud 6 commence à communiquer avec le noeud en aval destinataire, il envoie un RTS à ce noeud destinataire et reçoit ensuite un CTS de celui-ci, puis il envoie les données et reçoit un acquittement provenant de ce noeud destinataire.

Dans l'exemple suivant, on a simulé un réseau fixe classique (figure 4.6) et un réseau mobile (figure 4.9) comportant des noeuds qui utilisent les caractéristiques d'un noeud mobile (découverte des noeuds voisins, mise à jour de la table de routage, routage). L'objectif de cet exemple est de montrer l'impact en énergie de la mobilité des noeuds. Pour mieux comparer la différence de la consommation d'énergie d'un noeud fixe avec un noeud mobile, on n'a pas simulé le déplacement des noeuds mais juste les effets des fonctions et des mécanismes supplémentaires pour un noeud mobile.

La figure 4.13 montre l'énergie consommée par le noeud 6 dans les deux types de réseaux présentés par les figures 4.6 et 4.9. On remarque que le noeud mobile consomme légèrement plus d'énergie que le noeud fixe. Ceci est dû aux autres fonctionnalités supplémentaires implémentées dans le noeud mobile.

Dans cet exemple, nous avons utilisé les mêmes paramètres que dans la table 4.2 avec notre outil.

On a pu constater également que, d'après les paramètres utilisés comme exemple

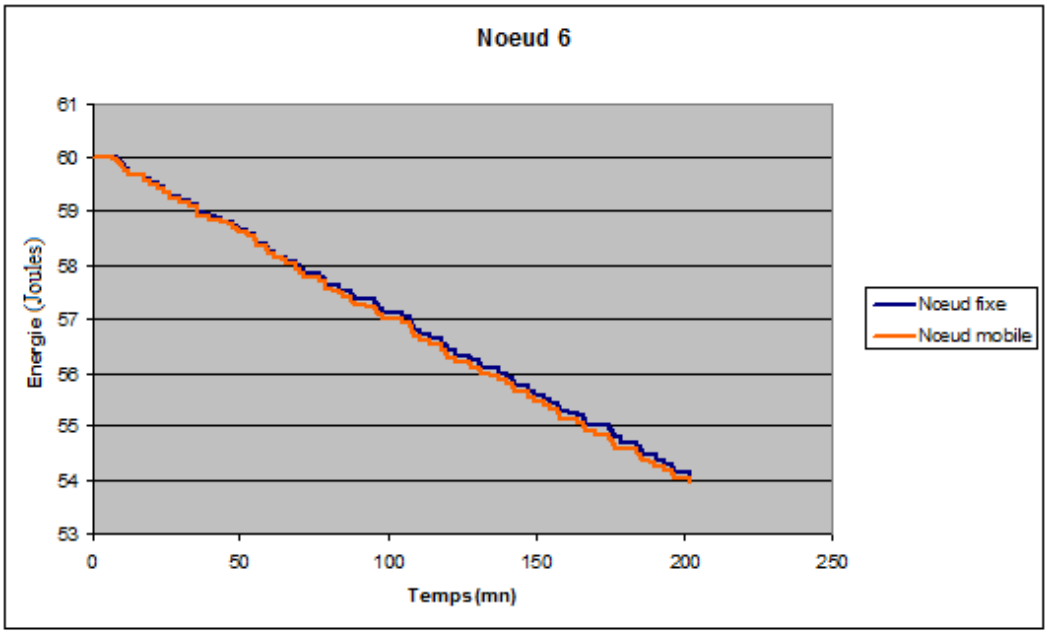

FIGURE 4.13 – Impact de la mobilité d'un noeud

dans cette comparaison, une période de découverte des noeuds voisins et de mise à jours de la table de routage d'un noeud mobile coute 0.05% en énergie. En effet, plus le noeud se déplace vite, plus la période de mise à jours de la table de routage est faible. Dans ce cas, le noeud mobile utilise plus souvent la découverte des noeuds voisins et met à jour sa table de routage plus fréquement. Dans la figure 4.13, on remarque une différence entre l'énergie consommée par le noeud mobile et celle du noeud fixe. Cette différence d'énergie est relative à la vitesse de déplacement du noeud capteurs, plus le noeud se déplace vite, plus il consomme de l'énergie puisqu'il doit se localiser dans le réseau et mettre à jour sa table de routage à l'aide de la découverte des noeuds voisins. Les périodes de localisation et de découverte des noeuds voisins dépendent du porteur du noeud capteur. Par exemple, la période de localisation est de 15 secondes pour une vitesse de déplacement d'une personne.

La figure 4.14 montre la consommation d'énergie du noeud 6 sur deux scénarios différents, réseau fixe (figure 4.6) et réseau mobile (figure 4.9). La courbe en dessus montre l'évolution de la consommation d'énergie, relative à l'évolution du profil de puissance dans la figure 4.12, selon le déplacement du noeud 6 dans les trois topologies montrées dans la figure 4.9. La courbe en dessous montre l'évolution de la consommation du noeud 6 correspondant à son profil de puissance dans la figure 4.8 évoluant dans le réseau fixe 4.6. On peut conclure que malgré l'ajout d'une fonction supplémen-

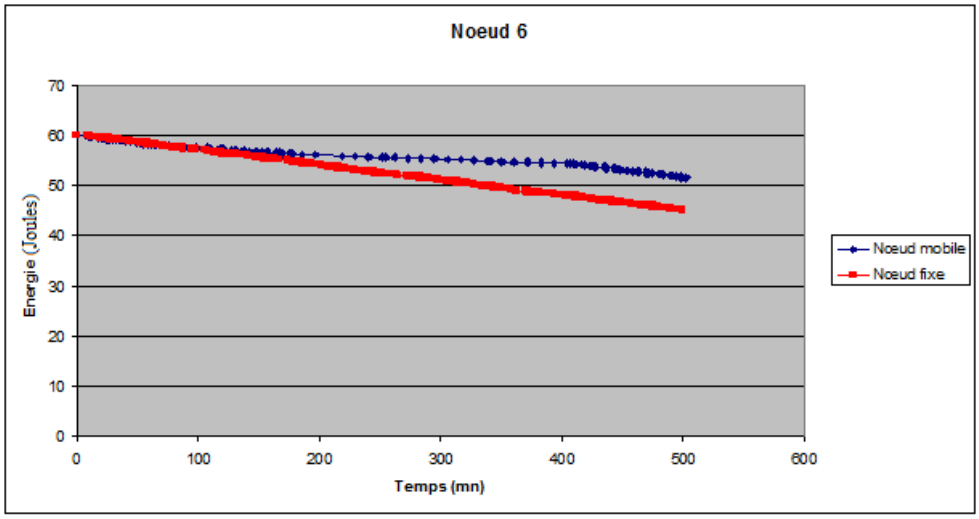

FIGURE 4.14 – Evolution de l'énergie restante d'un noeud fixe et mobile

taire au niveau du noeud mobile 6, sa consommation d'énergie dépend fortement de sa position par rapport à ses noeuds voisins et la position du puits.

4.3 Simulation considérant les transitions d'états

L'approche par fonctions permet de bien gérer la forme du profil de puissance désirée. L'objectif de cet exemple est de montrer que la modélisation par fonctions permet de définir le type de transition que l'on veut ajouter dans le modèle du noeud capteur. Dans ce chapitre, un changement d'état (transition) est défini par une puissance de transition et le temps de transition d'un état vers un autre état. Par exemple, durant la transition d'état sur le réveil d'un bloc radio, il y a changement de l'état veille vers l'état écoute du module radio. On peut définir alors des transitions d'état comme le changement d'état interne des fonctions du noeud capteur. Ces transitions d'état sont utilisées pour rapprocher nos résultats des mesures réelles.

Reprenons comme exemple le cas d'un réseau capteur classique, le modèle de noeud est composé de trois fonctions de base : la fonction d'acquisition, la fonction de réception et la fonction d'émission. On a implémenté le mécanisme CSMA/CA et on a utilisé le protocole TCP.

Des modifications ont été effectuées dans chacun des modèles de noeud afin de pouvoir prendre en compte les puissances de transitions et les temps de transitions dans les fonctions émission et réception. Il s'agit donc de modéliser la puissance consommée pendant que la fonction réception et la fonction émission commutent du mode

veille au mode écoute et au mode idle, respectivement.

La figure 4.15 montre deux types de transitions d'état différents. La partie droite de la figure montre que le noeud présente un pic de consommation lors du réveil et lors de la mise en veille des fonctions réception et émission. Tandis que la partie gauche de la figure 4.15 illustre que le noeud utilise deux types de transitions d'état différents pendant le réveil et la mise en veille des fonctions réception et émission, à savoir une augmentation de la consommation lors de l'éveil et une diminution de la consommation lors de la mise en veille.

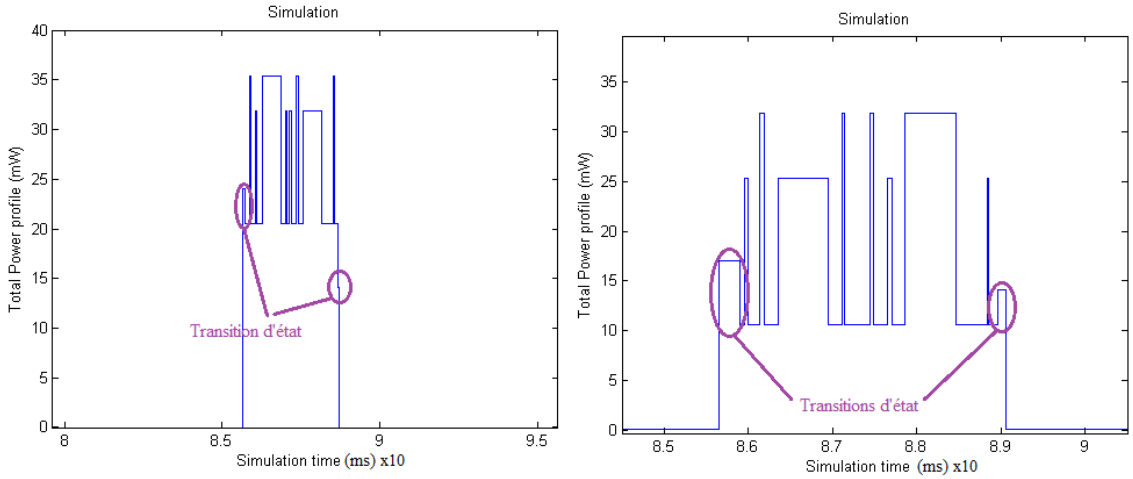

FIGURE 4.15 – Transitions d'état

L'un des avantages de la modélisation par l'approche fonction est la capacité de définir la forme des transitions d'état, c'est-à-dire, selon le temps de simulation, on peut définir l'évolution du profil de puissance des transitions d'état pour rapprocher nos résultats avec les mesures réelles.

4.4 Simulation d'un noeud capteur intelligent

L'objectif de ce dernier exemple est de montrer que l'approche par fonctions peut modéliser à un très haut niveau d'abstraction ce qui permet de simuler le comportement intelligent du noeud avant sa conception et donc de montrer les avantages ou les défauts des algorithmes envisagés pour la gestion de l'énergie par exemple. Avec cette approche, une fonction peut gérer les états des autres fonctions du noeud dans le but de rendre le noeud plus intelligent dans le sens où il contrôle sa consommation d'énergie.

Dans cet exemple, pour rendre un noeud capteur intelligent en consommation d'énergie, une fonction supplémentaire appellé "fonction de supervision" a été ajoutée.

Comme ce qui a été mentionné dans le chapitre précédent, la fonction de supervision intègre les équations qui caractérisent les décisions qui peuvent être prises par cette fonction.

Dans cette section, on présente quelques exemples d'équations simples qui caractérisent à titre d'exemple la décision sur la compression de données et la mise en veille du noeud capteur. En connaissant son calendrier de réveil et la distance du noeud avec ses voisins, un noeud peut décider de modifier l'état interne de ses fonctions ou de le maintenir.

Du côté de la fonction réception, les paramètres qui doivent être pris en compte sont : la puissance dans chaque état interne, la puissance requise pour la transition d'un état vers un autre et la durée de la transition. En d'autres termes, l'énergie consommée pendant le mode veille du noeud et celle pendant le réveil du noeud doivent être connues, en prenant en compte le calendrier de réveil du noeud.

En première approximation, l'énergie consommée par la fonction réception est donc définie par :

$$E_{Rx} = P_{Rx\_veille} t_{veille} + P_{Rx\_\text{écoute}} t_{\text{écoute}} + P_{Rx\_\text{réception}} t_{\text{réception}} \tag{4.1}$$

L'énergie totale consommée par la fonction réception est la somme de : l'énergie consommée pendant la durée (t_{veille}) où la fonction réception est en mode veille (puissance consommée P_{Rx_veille}), l'énergie consommée pendant la durée $(t_{\text{écoute}})$ où la fonction réception est en mode écoute (puissance consommée $P_{Rx\_\text{écoute}}$) et l'énergie consommée pendant la durée $(t_{\text{réception}})$ où la réception des données (puissance consommée $P_{\text{Rx\_réception}}$).

Intuitivement, l'état de la fonction réception peut être commutée en mode veille quand les deux inégalités présentées par (4.2) sont vérifiées. C'est-à-dire, pour une durée t_1, si l'énergie consommée pendant la transition d'état (veille-écoute) additionnée à l'énergie consommée en mode veille est inférieure à l'énergie consommée en mode écoute. Il est alors préférable de commuter l'état de la fonction réception en mode veille, sinon, elle doit rester en mode écoute.

$$\begin{cases} P_{veille\_\text{écoute}} t_{transition} + P_{veille}(t_1 - t_{transition}) \leq P_{\text{écoute}} t_1 \\ t_1 > t_{transition} \end{cases} \tag{4.2}$$

De la même manière, la fonction émission a trois états possibles : veille (fonction totalement éteinte), idle (fonction prêt à transmettre un message) et émission (fonction en cours d'émission d'un message). Une optimisation possible consiste à considérer le passage du mode idle au mode veille. Les conditions nécéssaires pour commuter l'état de la fonction émission en mode veille sont alors exprimées par les deux inégalités de (4.3) :

$$\begin{cases} P_{veille\_idle} t_{transition} + P_{veille}(t_6 - t_{transition}) \leq P_{idle} t_6 \\ t_6 > t_{transition} \end{cases} \quad (4.3)$$

Si ces deux inégalités ne sont pas vérifiées, il est préférable de garder l'état de la fonction émission en mode idle.

Plusieurs études [98] [99] [100] proposent d'optimiser l'énergie dans un noeud en limitant le temps de transmission de messages. Cela passe par la compression des données transmises si l'énergie nécessaire à la compression d'un message reste inférieure au gain d'énergie pour l'émission du message compressé.

La décision de compression dépend donc des deux inégalités présentées dans (4.4).

$$\begin{cases} \frac{P_{Tx\_émission}}{R_b} \geq \frac{P_{comp}}{V_{comp}} + \frac{P_{Tx\_émission}}{R_b}(1 - CR) \\ CR \leq CR_{max}(H) \end{cases} \quad (4.4)$$

Où,

$P_{Tx\_transmit}$: puissance d'émission [Watt]
P_{comp} : puissance requise pour la compression [Watt]
V_{comp} : vitesse de compression [bit/seconde]
R_b : débit de transmission [bit/seconde]
CR : taux de compression [%]
$CR_{max}(H)$: taux de compression maximal [%]

Les inégalités présentées en (4.2) (4.3) (4.4) et les prises de décision associées ont été intégrées dans la fonction supervision du modèle de la figure 3.10.

Les paramètres utilisés dans cet exemple sont présentés dans les tables 4.3 et 4.4.

En considérant la topologie présentée précédemment (figure 4.9) et sans introduire la fonction supervision, l'évolution du profil de puissance du noeud 6 est montré par la figure 4.16. On peut remarquer qu'après chaque période, le noeud 6 commute son bloc radio (fonctions réception et émission) en mode veille.

Lorsque la fonction de supervision est implémentée dans le modèle de noeud 6, la simulation donne le profil présenté dans la figure 4.17. On peut remarquer alors que si

Fonction	Etat	Puissance consommée
Emission	Emission Idle Veille	25.5 mW 10.5 mW 0.01 mW
Réception	Réception Ecoute Veille	32 mW 10.5 mW 0.01 mW
Acquisition	Acquisition Veille	20 mW 0.01 mW

TABLE 4.3 – Paramètres de simulation pour le noeud 6

Puissance de transition veille-écoute	17 mW
Puissance de transition écoute-veille	13 mW
Durée de transition veille-écoute	1,6 μs
Durée de transition écoute-veille	0,8 μs
Débit de transmission	250 kbps
Taille de données traitées	1250 bits
Taux de compression	50%

TABLE 4.4 – Autres paramètres pour le noeud 6

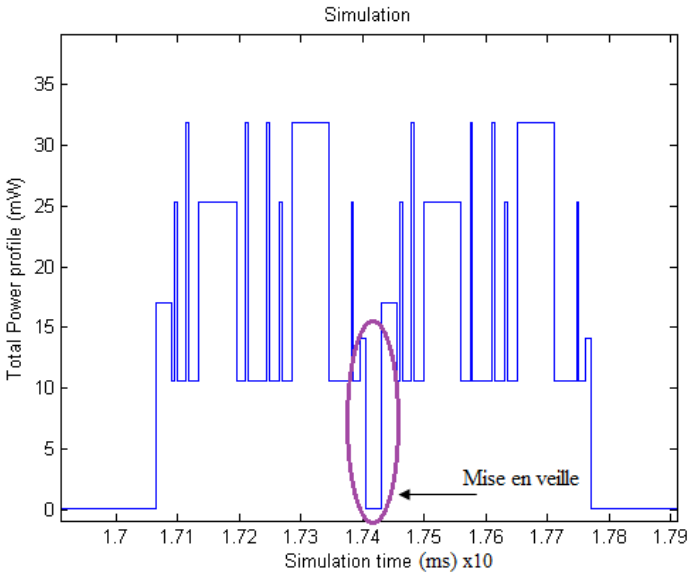

FIGURE 4.16 – Deux périodes de relais sans l'effet de la fonction de supervision

deux périodes sont très proches, la fonction de supervision contrôle les deux fonctions réception et émission de façon à ce qu'elles ne soient pas commutées en mode veille

mais reste en mode idle-écoute.

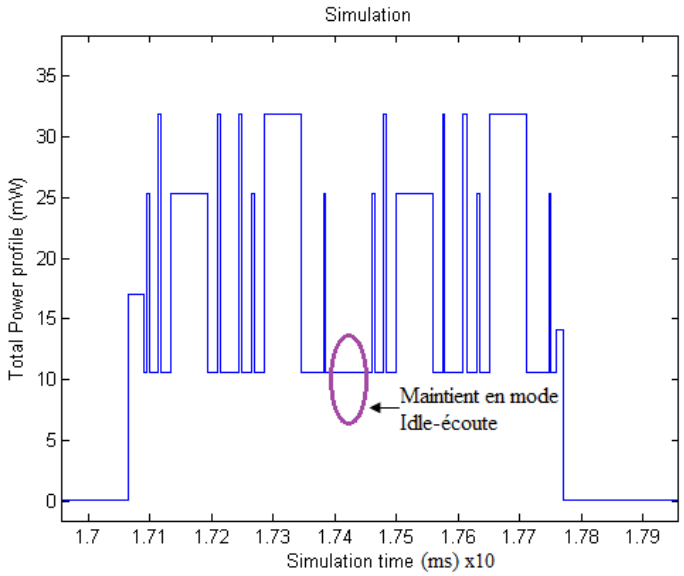

FIGURE 4.17 – Influence de la fonction de supervision

La figure 4.18 montre une comparaison de la consommation d'énergie d'un noeud capteur simple et un noeud capteur intelligent durant deux périodes de relais montrées dans les figures 4.16 et 4.17.

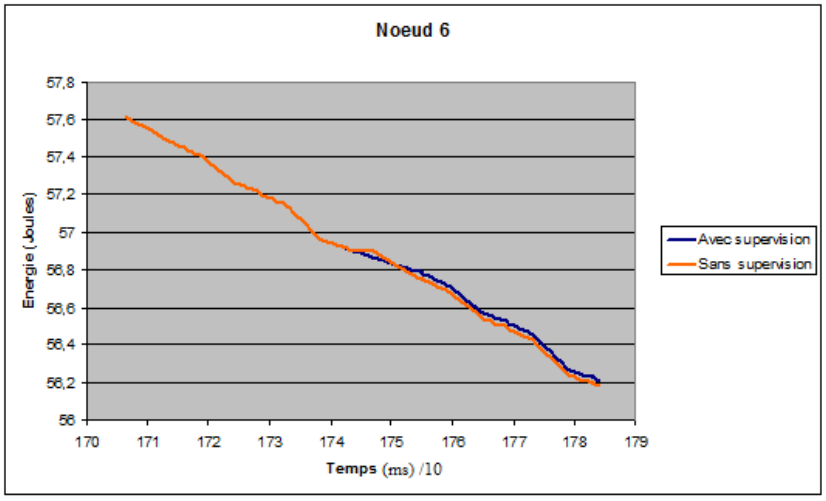

FIGURE 4.18 – Gain en énergie de la fonction supervision

Concernant la compression de données, la figure 4.19 montre une période d'acquisition et des échanges de trames de contrôle avant (émission de RTS, réception de CTS) et après (réception de ACK) l'émission des données acquises et non compressées.

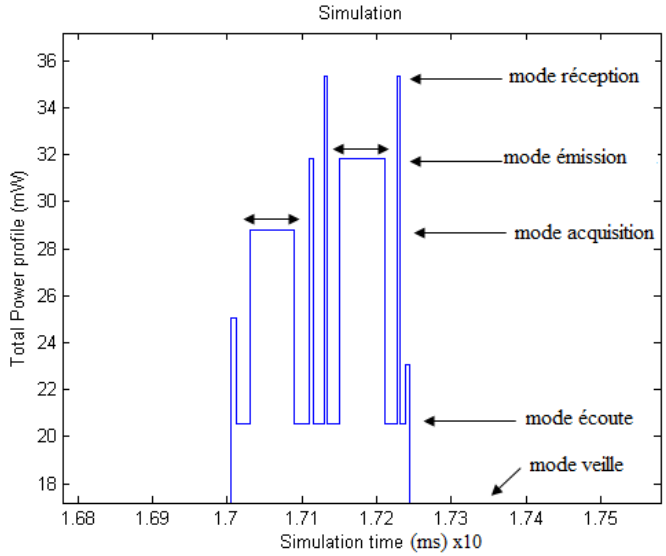

FIGURE 4.19 – Acquisition puis émission de données sans compression

La figure 4.20 montre l'évolution du profil de puissance du noeud 2 pendant une période d'acquisition suivie d'une émission des données acquises compressées. On peut observer le délai supplémentaire lié à la compression entre l'acquisition des données et les échanges des trames de contrôle (RTS et CTS). On peut aussi observer la diminution du temps requis pour l'émission.

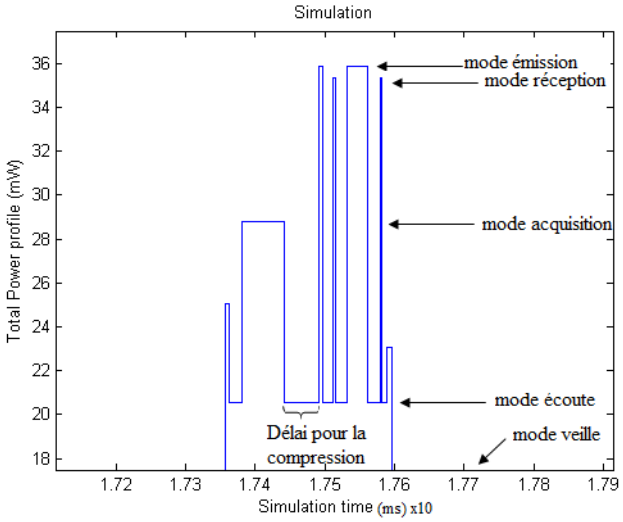

FIGURE 4.20 – Acquisition et émission de données compressées

Dans le cas d'un noeud qui retransmet des données reçues, la décision de compression est liée à la puissance requise pour l'émission, elle même dépendante indi-

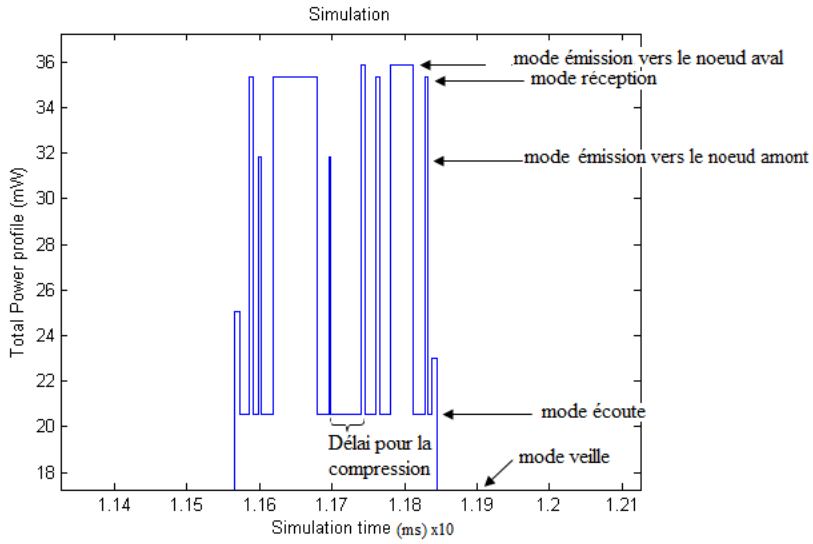

FIGURE 4.21 – Réception et émission de données compressées

rectement de la distance séparant les noeuds concernés par un échange. Il est ainsi possible de recevoir des données non compressées, de décider localement une compression avant de les transmettre. La figure 4.21 montre le profil de consommation d'un noeud où une telle décision est prise car la puissance requise pour l'émission vers un noeud aval (plus proche du sink) est supérieure à la puissance d'émission vers le noeud amont.

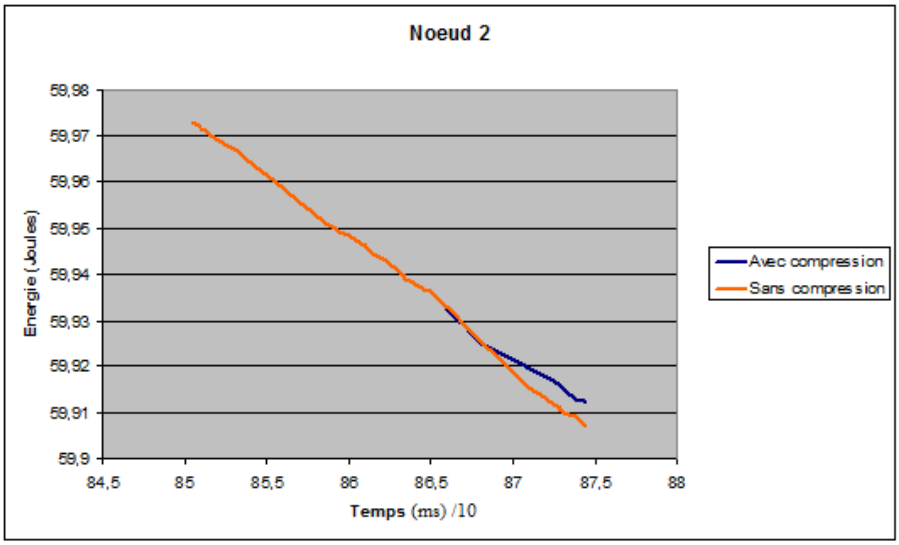

FIGURE 4.22 – Influence de la compression en énergie

La figure 4.22 montre le gain en énergie lors de la compression des données. Cette

situation montre que la simulation proposée permet de considérer l'effet d'un algorithme destiné à optimiser la consommation d'énergie d'un noeud.

4.5 Validation par des mesures réelles

Afin de valider le travail effectué, une expérience de modélisation appelée également testbed a été menée avec pour objectif de retrouver par simulation des résultats de mesures effectuées sur un noeud de capteur. Pour l'expérience, le noeud est constitué d'une carte Arduino (voir figure 4.23) et d'un module de transmission Zigbee XBee (voir figure 4.24) intégrant en matériel toute la partie protocole de transmission au format Zigbee. Cette architecture permet de créer simplement des réseaux de capteurs et de distinguer la puissance utilisée par la carte Arduino de la puissance utilisée par le module XBee.

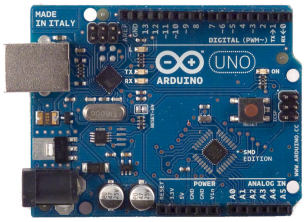

FIGURE 4.23 – Carte Arduino

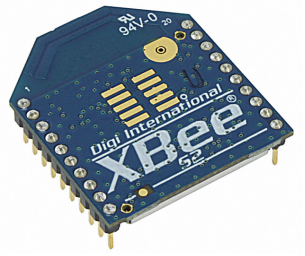

FIGURE 4.24 – Module de transmission Zigbee Xbee

Concernant le module XBee, la mesure ne permet pas de distinguer les consommations liées aux différentes couches de communication, ni de distinguer la consommation liée à la partie émission de celle liée à la partie réception. La validation du principe de modélisation nous amène à obtenir des profils de consommation, les niveaux exacts importent peu dans le cadre de cette validation puisque ce sont des constantes intégrées dans le modèle. Par contre ces niveaux (constantes) deviennent importants pour la modélisation d'une application réelle.

4.5.1 Application considérée

L'application considérée pour cette mesure est volontairement simple et correspond à celle modélisée dans le paragraphe (3.4.1). Pour les tests effectués, l'application ne comporte que 2 noeuds. Le premier joue le rôle de noeud puits dans le réseau et le rôle de coordinateur Zigbee. Le second joue le rôle de noeud capteur et de noeud émetteur. Le noeud capteur transmet périodiquement vers le puits des messages de taille 20 octets ou 5 octets. La structure des fonctions présentes sur le noeud capteur est présentée par la figure 4.25 et également identique à celle présentée par la figure 3.5.

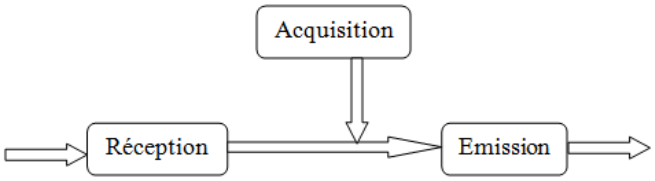

FIGURE 4.25 – Structure des fonctions présentes sur le noeud capteur

L'application est composée de 3 fonctions acquisition, réception et émission. Les fonctions réception et émission sont associées aux transmissions sur le réseau Zigbee. La fonction émission émet périodiquement des messages. Dans cette application, la fonction réception ne reçoit pas de données, mais est utilisée en lien avec la fonction émission pour gérer convenablement l'émission des trames de contrôle sur le réseau Zigbee.

4.5.2 Mesure

Les mesures effectuées concernent seulement le second noeud de l'application, c'est-à-dire le noeud capteur de l'application qui est un noeud terminal pour le réseau Zigbee. La phase de connexion entre le noeud terminal et le noeud coordinateur n'est pas considérée dans les mesures, celles-ci étant effectuées une fois que la connexion a eu lieu et que l'application est en fonctionnement normal. Pour chaque transmission d'un message de 20 octets, il y a 3 périodes d'émission/réception comme le montre la figure 4.26. Sur cette figure, 2 courbes sont présentes. Ce sont des mesures qui montrent l'évolution de la consommation des différentes parties en fonction du temps. Celle qui se trouve en haut de la figure montre l'évolution de la consommation du module XBee et celle du bas montre l'évolution de la consommation de la carte Arduino. La mesure de consommation du module XBee est bruitée, probablement en raison du principe de

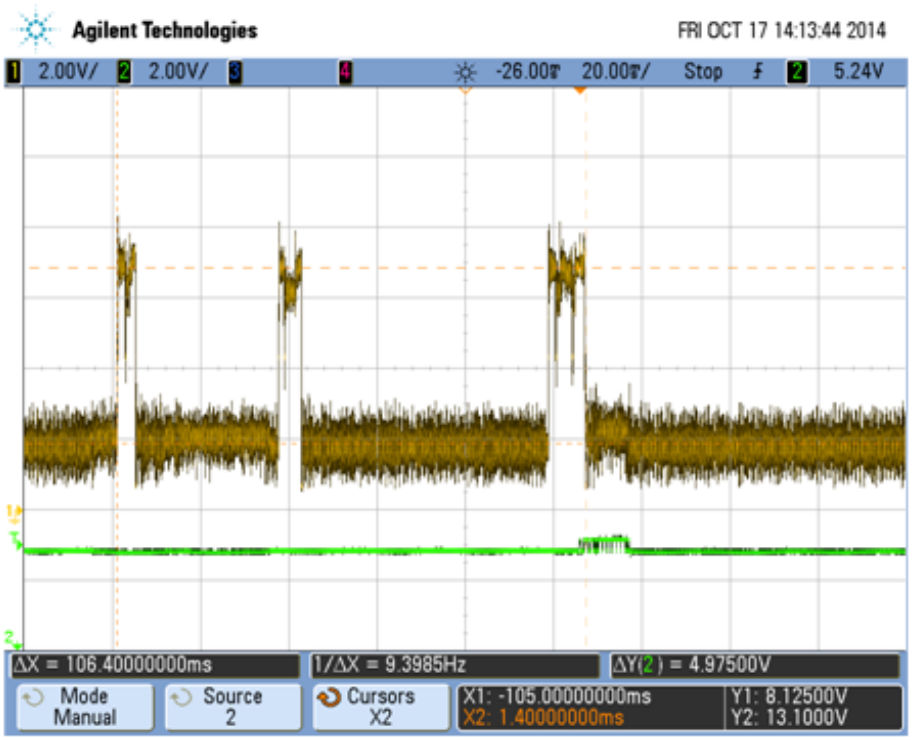

FIGURE 4.26 – Mesure complète d'une transmission de 20 octets

mesure. L'objectif de ce travail étant la modélisation et la simulation, une analyse des courbes relevées en considérant simplement la valeur moyenne a été effectuée et aucun travail de précision du principe de mesure n'a été mené.

La figure 4.27 montre un agrandissement des différentes phases de transmission de la figure précédente. La partie a) montre un agrandissement de la première phase, la partie b) présente un agrandissement de la seconde partie et la partie c) présente un agrandissement de la troisième partie de la communication.

Pour une transmission de données de taille 5 octets, la transmission ne nécessite qu'une seule période d'émission/réception comme le montre la figure 4.28. La transmission des données est faite de manière périodique.

Dans ce travail, le principe de mesure adopté convertit le courant absorbé en tension tout en maintenant la tension d'alimentation du système observé stable afin de rendre le principe de mesure non intrusif sur le comportement du système.

4.5.3 Modélisation

Dans le cadre de cette modélisation plusieurs états sont considérés pour chaque fonction. Pour la fonction capteur, la partie intéressante concerne l'activation périodique

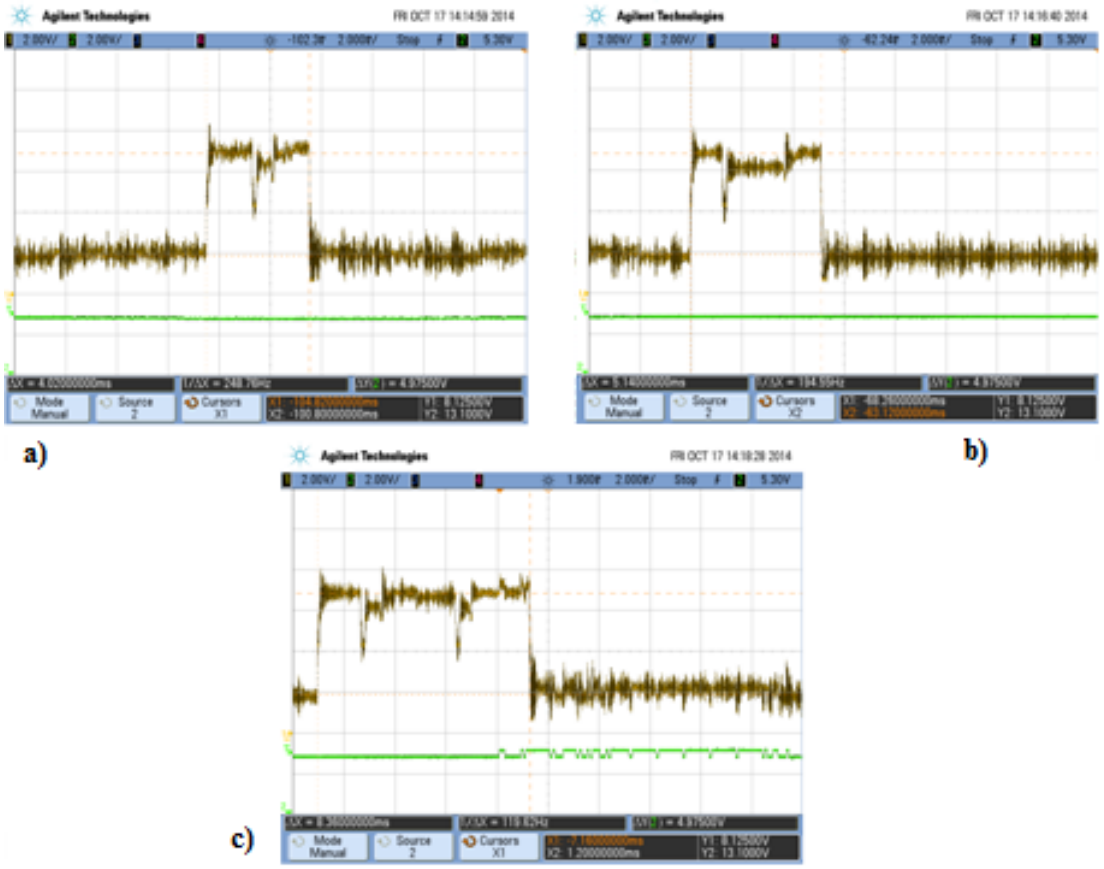

FIGURE 4.27 – Agrandissement des phases de transmission

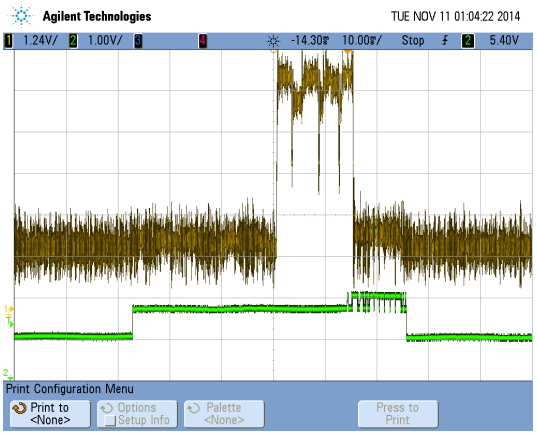

FIGURE 4.28 – Mesure d'une transmission de données de 5 octets

du comportement. La fonction capteur ne change pas de mode pour ce qui est de sa consommation.

La fonction émission est quant à elle activée par la réception de données. Elle peut

avoir 3 niveaux de consommation qui correspondent à l'état de la fonction : Off, Ready et Transmit.

La fonction réception peut se trouver dans 4 états différents : Off, Listen, Receive et Forward. Ce dernier correspond à l'énergie consommée par la fonction pour transmettre un acquittement. Le comportement de cette fonction est non seulement déclenché par la réception de données en provenance d'un autre noeud du réseau, mais aussi par la fonction émission du même noeud. En effet, en cas de transmission de message par la fonction émission, la fonction réception est mise à l'état Off pour ne pas solliciter les fonctions du noeud suite à la réception de données que le noeud a lui-même transmis. De plus, le protocole d'échange implique des échanges entre le noeud destinataire et le noeud émetteur, donc en cas de transmission de message il y a une évolution de l'état des fonctions émission et réception.

La modélisation de chaque fonction a nécessité plusieurs threads et plusieurs variables internes au module SystemC modélisant la fonction. La création de ces éléments et donc la génération du code SystemC modélisant l'application entière est faite à partir de l'analyse des fonctions qui composent l'application, ainsi que leur mode d'activation et la description sous forme d'état de leur évolution. Tout cela conforte l'idée que la définition d'un générateur automatique de code à partir des interfaces graphiques est envisageable pour automatiser ce travail de modélisation et de visualisation de résultats.

4.5.4 Résultats de simulation

A la suite d'une exécution du code SystemC correspondant au modèle de l'application et une mise en forme des résultats à l'aide de Matlab, les résultats présentés par la figure 4.29 ont été obtenus.

La figure 4.29 montre le résultat de simulation d'une transmission de 20 octets correspondant à la figure 4.27 sans la modélisation du bruit de mesure. La partie a) de la figure 4.29 montre les phases 1 et 2 de l'échange et la partie b) montre la phase 3. La figure 4.29 montre bien que la modélisation permet d'avoir comme résultat la même forme de consommation que dans les mesures que l'on a effectuées.

La figure 4.30 montre le résultat de simulation d'une transmission de 5 octets. On remarque que les données transmises ne sont pas segmentées mais sont transmises en une seule fois.

La comparaison entre les mesures effectuées sur une cible réelle et les résultats obtenus par simulation montre la correspondance possible dès le niveau fonctionnel

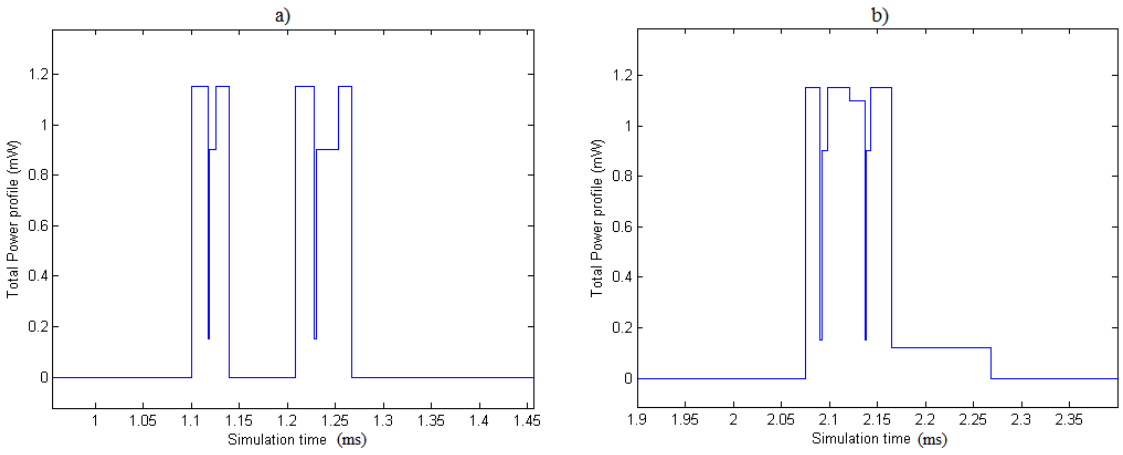

FIGURE 4.29 – Résultats de simulation d'une transmission de 20 octets

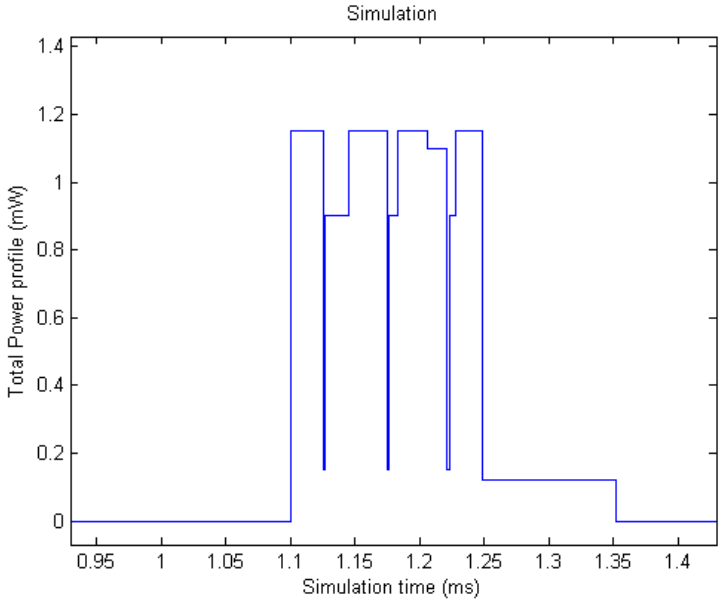

FIGURE 4.30 – Phase de transmission de données 5 octets

entre un travail de modélisation et la réalité. Cela permet d'envisager, sans posséder le matériel final, une évaluation de plusieurs solutions d'implantation des fonctions à partir du moment où l'on obtient des informations, même approximatives, sur les durées, les différents états et consommations que peuvent avoir une fonction pour un mode particulier d'implantation.

4.6 Conclusions

En guise de résumé, on a vu dans ce chapitre les avantages que peut apporter la modélisation adoptant l'approche par fonctions. Quelques exemples ont été présentés pour mettre en valeur chaque caractéristique importante de cette approche. L'exemple qui compare l'utilisation du protocole avec et sans acquittement, en utilisant ou pas un mécanisme de réservation de canal, permet de montrer que la modélisation par fonctions fournit comme résultat l'évolution du profil de puissance selon le comportement du noeud. Cette relation entre le comportement et la consommation d'énergie n'est pas facilement identifiable dans le simulateur NS2.

On peut identifier dans le résultat de simulation l'évolution du profil de puissance de chaque fonction dans un noeud. L'utilisation du mécanisme de réservation de canal est aussi facilement identifiable grâce à l'utilisation des trames de contrôles (RTS/CTS). Le comportement d'un noeud utilisant le protocole à acquittement est alors différent de celui d'un noeud qui utilise le protocole sans acquittement, d'où leurs profils de puissance légèrement différents. L'influence en énergie de l'utilisation de ces deux protocoles a été comparée. Ensuite, quelques hypothèses ont été proposées pour expliquer la différence de l'énergie consommée par un noeud utilisant le protocole à acquittement et celle d'un noeud utilisant le protocole sans acquittement.

L'exemple d'un réseau de capteurs avec des noeuds mobiles a permis de mettre en valeur une spécificité de l'approche par fonctions, la relation entre chaque fonction est la consommation d'énergie. Une fonction de localisation a été ajoutée dans chaque noeud mobile, intégrant ainsi le protocole de routage géographique et le mécanisme de découverte des noeuds voisins. L'intérêt de cet exemple est aussi de mettre en évidence l'impact de la mobilité sur la consommation d'énergie. Une comparaison sur la consommation d'énergie d'un noeud fixe et un noeud mobile a été présentée.

Le dernier exemple sur les noeuds capteurs intelligents a bien montré la capacité de la modélisation proposée à simuler le comportement intelligent des noeuds afin d'optimiser leur consommation d'énergie. La modélisation par l'approche par fonctions autorise qu'une fonction (fonction supervision) gère les états des autres fonctions dans le but d'optimiser la consommation d'énergie. On a pu ainsi distinguer l'impact en consommation d'énergie quand on a ajouté ou quand on a retiré la fonction supervision dans le noeud.

Les premières mesures qui ont été faites nous ont permis de montrer que la modélisation puis simulation sont en accord avec des résultats de mesures réelles. Cela permet en quelque sorte de conforter l'idée que l'approche par fonctions est un moyen inté-

ressant et performant pour modéliser la consommation d'énergie des noeuds capteurs très tôt dans les étapes de conception du système.

Conclusion générale

Dans ce rapport de thèse, nous avons proposé une autre façon de modéliser les réseaux de capteurs sans fil en utilisant l'approche par fonctions. Cette approche permet de bien considérer la consommation d'énergie de chaque activité du noeud dans le réseau. Par conséquent, l'approche par fonctions permet d'avoir le lien entre chaque fonction du noeud et l'énergie consommée par celle-ci. La modélisation par fonctions intervient très tôt dans le processus de conception (conception fonctionnelle). Cela permet de placer cette étude à un très haut niveau d'abstraction. En effet, les modèles de noeuds sont plus facilement manipulables et surtout on peut avoir un impact important sur les éventuelles optimisations. Ainsi, l'objectif de notre travail est de modéliser, avec l'approche par fonctions, la consommation d'énergie de quelques variantes de noeuds capteurs dans le but de donner aux concepteurs des informations bien avant la conception du noeud.

Dans le premier chapitre, nous avons défini les principales caractéristiques des noeuds dans les réseaux de capteurs sans fil WSN. Nous avons exposé les généralités sur les noeuds capteurs sans fil. Quelques définitions sur la durée de vie d'un noeud ont été citées afin de bien définir notre objectif. Puis, nous avons décrit globalement l'aspect matériel et l'aspect logiciel d'un noeud. Nous avons cité les caractéristiques et les rôles des différentes couches protocolaires dans les WSN. Nous avons présenté également quelques variantes de protocole MAC et routage parmi les plus populaires. Enfin, nous avons présenté l'aspect surconsommation d'énergie que l'on rencontre souvent dans les noeuds capteurs sans fil. L'objectif de ce chapitre est de connaître quelques variantes de noeuds parmi les plus connus ainsi que leurs fonctionnements.

Dans le deuxième chapitre, nous avons introduit le concept de la modélisation dans le domaine des WSN. Puis, nous avons décrit les différents niveaux d'abstraction de modélisation d'un système. Quelques simulateurs parmi les plus connus et les plus utilisés figurent aussi dans ce chapitre. Nous avons pu alors constater que ces différents simulateurs ne satisfont pas nos besoins, aucun d'eux ne convient au modèle de niveau fonction.

Ainsi, dans le chapitre trois, nous proposons la modélisation avec l'approche par fonctions. En premier lieu, nous avons commencé par présenter les caractéristiques de quelques fonctions parmi les plus connues dans les noeuds capteurs. Puis, nous avons décrit la modélisation d'une fonction. Ensuite, avec cette approche par fonctions, nous avons présenté une manière de modéliser un noeud capteurs. Quelques exemples de modèles de noeuds ont été présentés. Les règles de traduction de ces modèles en programmes basés sur SystemC sont présentées, ce qui permet de rendre le modèle simulable. Nous avons modélisé le canal de communication de façon très simple. Le principe de ce modèle de canal est de fournir périodiquement une matrice de valeurs de distance entres les noeuds mobiles. Quelques exemples de modèles de WSN sont alors prêts à être simulés.

Dans le quatrième chapitre, nous avons simulé tous les modèles de WSN décrits dans le chapitre précédent. Les résultats de ces simulations démontrent la capacité du principe de modélisation à montrer l'influence de choix du protocole de communication, de la mobilité, ou de l'intelligence embarquée dans un noeud sur sa consommation d'énergie. On a pu remarquer l'intérêt de la modélisation du canal de communication dans le cas d'un WSN mobile. Cette approche par fonctions semble intéressante parce qu'elle permet de voir le lien entre les différents mécanismes/applications implémentés dans le noeud et la consommation d'énergie du noeud. En résumé, cette modélisation par fonctions permet de fournir des informations aux concepteurs bien avant la conception du noeud capteur pour faciliter le choix d'application et les mécanismes qui seront utilisés, en tenant compte de la consommation d'énergie. Des mesures réelles ont été effectuées dans le but de valider notre approche. En effet, la ressemblance de nos résultats de mesure avec nos résultats de simulation montre que la modélisation proposée permet d'estimer d'une manière fidèle la consommation d'énergie du noeud.

L'une des perspectives importantes de cette étude est de trouver une bibliothèque de consommation de puissance d'une plateforme fédératrice ouverte. En effet, cette bibliothèque de consommation de puissance est très utile pour définir les paramètres d'entrée qui seront utilisés dans nos modèles. Il est aussi envisageable de créer des modèles correspondant aux différentes couches protocolaires avec des paramètres permettant de les adapter. L'une des grandes perspectives de cette étude est également de modéliser la recolte d'énergie (energy harvesting) au niveau des noeuds capteurs. Cette technique de recolte d'énergie est très utilisée pour augmenter la durée de vie des noeuds. Elle permet de transformer les sources d'énergies extérieures (rayon solaire, vibration, température, etc) en énergie électrique que l'on peut stocker. D'autres protocoles et mécanismes peuvent être également ajoutés dans le modèle de noeud, par

exemple, tenir compte des éventuelles erreurs de transmission et la retransmission de données, ou le choix de tête de cluster selon le niveau de batterie des noeuds, etc. Le modèle de canal peut être amélioré en considérant le temps de propagation, les perturbations, les obstacles, les différents typs de canals (unidirectionnel, bidirectionnel, symetrique, etc), etc. Il est envisagable de considérer les différents modes de propagation radio (two-ray ground, free space et shadowing). La modélisation d'une technologie telle que la CRWSN (Cognitive Radio-based Wireless Sensor Networks) est également une perspective très intéressante, notament en ce qui concerne les protocoles MAC et routage ainsi que l'estimation de canal. Cela permettrait de simuler un réseau composé de plusieurs appareils sans fil intelligents capables de coexister d'une manière efficace dans une zone géographique. Dans ce cas, les noeuds sont capables d'adapter leurs paramètres de transmission, tels que la fréquence, la puissance d'émission, la modulation et les protocoles pour avoir une transmission plus efficace. Tout compte fait, la modélisation utilisant l'approche par fonctions permet de fournir un modèle de noeud qui n'est pas contraint en architecture matérielle et logicielle. Une perspective sur l'environnement de modélisation de type framework est aussi envisageable pour la modélisation par l'approche par fonctions. D'ailleurs, plusieurs simulateurs font appel à ce type d'environnement de modélisation pour compléter les manques dans leurs modèles. Par exemple, OMNET++ sollicite PAWiS pour modéliser le canal de communication. Dans notre cas, on peut proposer un environnement de modélisation portant sur la consommation d'énergie relative aux différentes activités du noeud capteur dans le réseau. Ainsi, la modélisation par l'approche par fonctions peut également contribuer à l'amélioration des simulateurs existants en termes de consommation d'énergie. Il est aussi envisageable de créer un outil permettant de générer le code SystemC d'un modèle à partir de la description graphique des fonctions qui composent le noeud et leur description algorithmique.

Bibliographie

[1] V. NAIR. **Heterogeneous Wireless Communication Devices- Present and Future**. *International Conference on Recent Advances in Microwave Theory and Applications*, 2008. 10, 11

[2] S. RAO, S. KESHRI, D. GANGWAR, P. SUNDAR, AND V. GEETHA. **A Survey and Comparison of GTS Allocation and Scheduling Algorithms in IEEE 802.15.4 Wireless Sensor Networks**. *IEEE Conference on Information & Communication (ICT)*, 2013. 11

[3] T. NAUMOWICZ, R. FREEMAN, A. HEIL, M. CALSYN, E. HELLMICH, A. BRANDLE, T. GUILFORD, AND J. SCHILLER. **Autonomous monitoring of vulnerable habitats using a wireless sensor network**. *Workshop on Real-World Wireless Sensor Networks, REALWSN'08*, 2008. 16

[4] R. KUMAR, A. RANJAN, P. JAIN, AND V.R. BAGREE. **wildcense : Gps based animal tracking system**. *In Intelligent Sensors, Sensor Networks and Information Processing ISSNIP 2008*, 2008. 16

[5] A. ARORA, P. DUTTA, S. BAPAT, V. KULATHUMANI, H. ZHANG, V. NAIK, V. MITTAL, H. CAO, M. DEMIRBAS, M. GOUDA, Y. CHOI, T. HERMAN, S. KULKARNI, U. ARUMURAM, M. NESTERENKO, A. VORA, AND M. MIYASHITA. **A line in the sand : A wireless sensor network for target detection, classification and tracking**. *IEEE Computer Networks*, **46(5)** :605–634, 2005. 16

[6] C. L. JAM, D. A. KNAPP, Z. M. KOENIG, S. J. LUKE, B. A. POHL, A. SCHACH VON WITTENAU, J. K. WOLFORD, T. B. GOSNELL, AND J. M HALL. **Technical Repport. Gamma-ray identification of nuclear weapon materials**. Technical report, Lawrence Livermore National Lab., Livermore, CA (US), 2007. 16

[7] M.J. BROWN. **Technical Repport. Users guide developed for the jbrews project**. Technical report, Los Alamos National Laboratory of California University, 1999. 16

[8] H. RAMAMURTHY, B. S. PRABHU, R. GADH, AND A. M. MADNI. **Smart sensor platform for industrial monitoring and control**. *4th IEEE SENSORS*, 2005. 16

[9] H. RAMAMURTHY, B. S. PRABHU, R. GADH, AND A. M. MADNI. **Wireless industrial monitoring and control using a smart sensor platform**. *IEEE Sensors Journal*, **7(5)** :611–618, 2007. 16

[10] S. KIM, S. PAKZAD, D. E. CULLER, J. DEMMEL, G. FENVES, S. GLASER, AND M. TURON. **Wireless sensor networks for structural health monitoring**. *In Andrew T. Campbell, Philippe Bonnet, and John S. Heidemann, editors : SenSys*, 2006. 16

[11] Y. GUO, F. KONG, D. ZHU, A. TOSUN, AND Q. DENG. **Sensor placement for lifetime maximization in monitoring oil pipelines**. *In ICCPS '10 : Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, 2010. 16

[12] I. STOIANOV, L. NACHMAN, S. MADDEN, AND T. TOKMOULINE. **Pipeneta wireless sensor network for pipeline monitoring**. *In IPSN '07 : Proceedings of the 6th international conference on Information processing in sensor networks*, 2007. 16

[13] C. R. BAKER, K. ARMIJO, S. BELKA, M. BENHABIB, V. BHARGAVA, N. BURKHART, A. DER MINASSIANS, G. DERVISOGLU, L. GUTNIK, M. B. HAICK, C. HO, M. KOPLOW, J. MANGOLD, S. ROBINSON, M. ROSA, M. SCHWARTZ, C. SIMS, H. STOFFREGEN, A. WATERBURY, E. S. LELAND, T. PERING, AND P. K. WRIGHT. **Wireless sensor networks for home health care**. *In AINA Workshops (2), IEEE Computer Society*, 2007. 16

[14] D. MALAN, T. FULFORD-JONES, M. WELSH, AND S. MOULTON. **Codeblue : An ad hoc sensor network infrastructure for emergency medical care**. *In International Workshop on Wearable and Implantable Body Sensor Networks*, 2004. 16

[15] K. LORINCZ, BOR RONG CHEN, G. WERNER CHALLEN, A. R. CHOWDHURY, S. PATEL, P. BONATO, AND M. WELSH. **Mercury : a wearable sensor network platform for high-fidelity motion analysis**. *In Sens 09 : Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009. 16

[16] A. SHRESTHA AND L. XING. **A Performance Comparison of Different Topologies for Wireless Sensor Networks**. *IEEE Conference on Technologies for Homeland Security*, 2007. 17

[17] H. CUI, C. FENG, R. P. LIU, X. HUANG, AND Y. LIU. **Prolonging Network Lifetime in Two-level Heterogeneous Wireless Sensor Networks**. *2012 International Symposium on Communications and Information Technologies (ISCIT)*, 2012. 19, 41, 46

[18] R. N. ENAM, S. MISBAHUDDIN, AND M. IMAM. **Energy Efficient Round Rotation Method for a Random Cluster Based WSN**. *2012 International Conference on Collaboration Technologies and Systems (CTS)*, 2012. 19, 41, 46

[19] J-S. KIM, S-Y CHOI, S-J HAN, AND J-H CHOI. **Alternative Cluster Head Selection Protocol for Energy Efficiency in Wireless Sensor Networks**. *Software Technologies for Future Dependable Distributed Systems*, 2009. 19

[20] N. FOURTY, A. V. DEN BOSSCHE, AND T. VAL. **An advanced study of energy consumption in an IEEE 802.15.4 based network : everything but the truth on 802.15.4 node lifetime**. *Computer Communications, Elsevier, Numéro spécial Wireless Green*, **35** :1767–1767, 2012. 19

[21] K. HOLGER AND ANDREAS WILLIG. **Protocols and Architectures for Wireless Sensor Networks**. *Wiley*, 2005. 20

[22] J-H. CHANG AND L. TASSIULAS. **Energy conserving routing in wireless ad-hoc networks**. *In Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, **1** :22–31, 2000. 20

[23] A. GIRIDHAR AND P. R. KUMAR. **Maximizing the functional lifetime of sensor networks**. *In Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, page 2, 2005. 20

[24] V. P. MHATRE, C. ROSENBERG, D. KOFMAN, R. MAZUMDAR, AND N. SHROFF. **A minimum cost heterogeneous sensor network with a lifetime constraint**. *IEEE Transactions on Mobile Computing*, pages 4–15, 2005. 20

[25] W. WANG, V. SRINIVASAN, AND K-C. CHUA. **Using mobile relays to prolong the lifetime of wireless sensor networks**. *In Proceedings of the 11th annual international conference on Mobile Computing and networking (MobiCom'05)*, pages 270–283, 2005. 20

[26] A. CERPA AND D. ESTRIN. **Ascent : Adaptive self-configuring sensor networks topologies**. *IEEE Transactions on Mobile Computing*, pages 272–285, 2004. 20

[27] J. DENG, Y. S. HAN, W. B. HEINZELMAN, AND PRAMOD K. **Scheduling sleeping nodes in high density cluster-based sensor networks**. *Mobile Networks and Applications, Varshney*, page 2005, 825-835. 20

[28] K. HELLMAN AND M. COLAGROSSO. **Investigating a wireless sensor network optimal lifetime solution for linear topologies**. *Journal of Interconnection Networks*, pages 91–99, 2006. 20

[29] D. TIAN AND N. D. GEORGANAS. **A coverage-preserving node scheduling scheme for large wireless sensor networks**. *In Proceedings of the 1st ACM international workshop on Wireless Sensor Networks and Applications (WSNA'02)*, pages 32–41, 2002. 20, 21

[30] B. CĂRBUNAR, A. GRAMA, J. VITEK, AND O. CĂRBUNAR. **Redundancy and coverage detection in sensor networks**. *ACM Transactions on Sensor Networks*, pages 94–128, 2006. 20, 21

[31] W. MO, D. QIAO, AND Z. WANG. **Mostly-sleeping wireless sensor networks : Connectivity, k-coverage, and alpha-lifetime**. *In Proceedings of the the 43rd Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA*, 2005. 20

[32] M. CARDEI, MY T. THAI, Y. LI, AND W. WU. **Energy-efficient target coverage in wireless sensor networks**. *In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, **3** :1976–1984, 2005. 20

[33] H. LIU, PENG-JUN WAN, CHIH-WEI YI, X. JIA, S. A. M. MAKKI, AND N. PISSINOU. **Maximal lifetime scheduling in sensor surveillance networks**. *In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFO-COM'05)*, **4** :2482–2491, 2005. 20

[34] M. BHARDWAJ AND A. CHANDRAKASAN. **Upper bounds on the lifetime of wireless sensor networks**. *In Proceedings of the IEEE International Conference on Communications (ICC'01)*, 2001. 20

[35] M. BHARDWAJ AND A. P. CHANDRAKASAN. **Bounding the lifetime of sensor network via optimal role assignments**. *In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, **3** :1587–1596, 2002. 20

[36] H. ZHANG AND J. C. HOU. **Maintaining sensing coverage and connectivity in large sensor networks**. *Ad Hoc & Sensor Wireless Networks*, 2005. 20

[37] H. ZHANG AND J. C. HOU. **Maximizing α-lifetime for wireless sensor networks**. *Proceedings of the 3rd International Workshop on Measurement, Modeling, and Performance Analysis of Wireless Sensor Networks (SenMetrics'05)*, 2005. 20

[38] H. ZHANG AND J. C. HOU. **On the upper bound of α-lifetime for large sensor networks**. *ACM Transactions on Sensor Networks*, pages 272–300, 2005. 20

[39] K. WU, Y. GAO, F. LI, AND Y. XIAO. **Lightweight deployment-aware scheduling for wireless sensor networks**. *Mobile Networks and Applications*, pages 837–852, 2005. 21

[40] G. XING, X. WANG, Y. ZHANG, C. LU, R. PLESS, AND C. GILL. **Integrated coverage and connectivity configuration for energy conservation in sensor networks**. *ACM Transactions on Sensor Networks*, pages 36–72, 2005. 21

[41] M. CARDEI AND J. WU. **Coverage in wireless sensor networks**. *In Y. Yu, R. Govindan, and D. Estrin, editors, Handbook of Sensor Networks*, 2004. 21

[42] A. KANSAL, A. RAMAMOORTHY, MANI B. SRIVASTAVA, AND G. J. POTTIE. **On sensor network lifetime and data distortion**. *In Proceedings of the IEEE International Symposium on Information Theory (ISIT'05)*, pages 6–10, 2005. 21

[43] K. SHA AND W. SHI. **Modeling the lifetime of wireless sensor networks**. *Sensor Letters*, 2005. 21

[44] C. F. CHIASSERINI, I. CHLAMTAC, P. MONTI, AND A. NUCCI. **Energy efficient design of wireless ad hoc networks**. *In Proceedings of the 2nd International IFIP-TC6 Networking Conference on Networking Technologies, Services, and Protocols ; Performance of Computer and Communication Networks ; and Mobile and Wireless Communications (NETWORKING'02)*, pages 376–386, 2002. 21

[45] S. SORO AND W. B. HEINZELMAN. **Prolonging the lifetime of wireless sensor networks via unequal clustering**. *In Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 2005. 21

[46] D. M. BLOUGH AND P. SANTI. **Investigating upper bounds on network lifetime extension for cell-based energy conservation techniques in stationary ad hoc networks**. *In Proceedings of the 8th annual international conference on Mobile Computing and networking (MobiCom'02)*, pages 183–192, 2002. 21

[47] S. KUMAR, A. ARORA, AND T. H. LAI. **On the lifetime analysis of always-on wireless sensor network applications**. *In Proceedings of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05)*, pages 186–188, 2005. 21

[48] S. TILAK, N. B. ABU-GHAZALEH, AND W. HEINZELMAN. **A taxonomy of wireless micro-sensor network models**. *SIGMOBILE Mobile Computing and Communications Review*, pages 28–36, 2002. 21

[49] I. F. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, AND E. CAYIRCI. **Wireless Sensor Networks : A Survey**. *Computer Networks*, **30** :393–422, 2002. 22, 27

[50] A. MERENTITIS, N. KRANITIS, A. PASCHALIS, AND D. GIZOPOULOS. **Low Energy Online Self-Test of Embedded Processors in Dependable WSN Nodes**. *IEEE Transaction on Dependable and Secure Computing*, 2012. 22

[51] M-A. HANSON, H-C. POWELL JR, A-T. BARTH, K. RINGGENBERG, B-H. CALHOUM, J-H. AYLOR, AND J. LACH. **Body Area Sensor Networks : Challenges and Opportunities**. *IEEE Computer*, 2009. 22

[52] OLIVIER SENTIEYS. **Efficacité énergétique : les technologies de l'information**. Technical report. 23

[53] A. BUHRIG. **Optimisation de la consommation des noeuds de réseaux de capteurs sans fil**. *Thèse de doctorat*, 2008. 23

[54] H. KARL AND A. WILLING. *Protocols and Architectures for Wireless Sensor Networks*. Wiley & Sons, 2005. 23

[55] C. LAW, A. K. MEHTA, AND K-Y. SIU. **A New Bluetooth Scatternet Formation Protocol**. *Mobile Networks and Applications*, **8** :485–498, 2003. 24

[56] W-Z. SONG, Y. WANG, C. REN, C. WU, AND X-Y. LI. **Multi-hop Scatternet Formation and Routing For Large Scale Bluetooth Networks**. *International Journal of Ad Hoc and Ubiquitous Computing*, **4** :251–268, 2009. 24

[57] Q. WANG AND W. YANG. **Energy Consumption Model for Power Management in Wireless Sensor Networks**. *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2007. 25, 41, 43, 44

[58] H. UNTERASSINGER, M. DIELACHER, M. FLATSCHER, S. GRUBER, G. KOWALCZYK, J. PRAINSACK, T. HERNDL, J. SCHWEIGHOFER, AND W. PRIBYL. **A Power Management Unit for Ultra-Low Power Wireless Sensor Networks**. *IEEE Africon 2011*, 2011. 25, 41

[59] I. AKYILDIZ, W. SU, Y. SANKARASUBRAMANIAM, AND E. CAYIRCI. **A Survey on Sensor Networks**. *IEEE Communication Magasine*, **40** :102–114, 2002. 26

[60] C.-Y. SHIH AND P.J. MARRÒN. **COLA : Complexity-Reduced Trilateration Approach for 3D Localization in Wireless Sensor Networks**. *Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, 2010. 26

[61] F. TIAN, W. GUO, C. WANG, AND Q. GAO. **Robust Localization Based on Adjustment of Trilateration Network for Wireless Sensor Networks**. *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08)*, 2008. 26

[62] Z. FANZHEN, Y. MIN, Z. CHENGWU, AND G. JIANLIANG. **An Improved Point-in-Triangulation Localization Algorithm Based on Cosine Theorem**. *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, 2012. 26

[63] O. TEKDAS AND V. ISLER. **Sensor Placement for Triangulation-Based Localization**. *IEEE Transactions on Automation Science and Engineering*, **7** :681–685, 2010. 26

[64] X. XIANG, Z. ZHOU, AND X. WANG. **Self-Adaptive On Demand Geographic Routing Protocols for Mobile Ad Hoc Networks**. *26th IEEE International Conference on Computer Communications (INFOCOM)*, IEEE'2007. 28, 41, 65

[65] http://www.tinyos.net/tinyos-1.x/doc/nesc/ref.pdf. 29

[66] Q. CAO, T. ABDELZAHER, J. STANKOVIC, AND T. HE. **The LiteOS Operating System : Towards Unix Like Abstraction for Wireless Sensor Networks**. *In Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008)*, 2008. 30

[67] A. DUNKELS, B. GRONVALL, AND T. VOIGT. **Contiki a Lightweight and Flexible Operating System for Tiny Networked Sensors**. *In Proceedings of the 9th Annual IEEE International Conference on Local Computer Networks, Washington, DC, USA*, 2004. 30

[68] P. JACQUET, P. MUHLETHALER, T. CLAUSEN, A. LAOUITI, A. QAYYUM, AND L. VIENNOT. **Optimized link state routing protocol for ad hoc networks**. *Multi Topic Conference IEEE INMI, Technology for the 21st Century*, 2001. 31

[69] T. CLAUSEN AND P. JACQUET. **Optimized Link State Routing Protocol**. http://www.ietf.org/rfc/rfc3626.txt. 31

[70] C. E. PERKINS, E. M. ROYER, AND S. R. DAS. **Ad hoc on demand distance vector (aodv) routing**. *In IETF, Internet Draft, draft-ietf-manet-aodv-05.txt*, 2000. 31, 32

[71] W. YE, J. HEIDEMANN, AND D. ESTRIN. **An energy-efficient MAC protocol for wireless sensor networks**. *In procedings of the Twenty-First Annual Joint*

Conference of the IEEE Computer and Communications Societies In INFOCOM 2002, **3**, 2002. 33

[72] P. LIN, C. QIAO, AND X. WANG. **Medium access control with a dynamic duty cycle for sensor networks**. *In Wireless Communications and Networking Conference, 2004 WCNC*, **3**, 2004. 34

[73] H. PHAM AND S. JHA. **An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC)**. *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2004. 34

[74] T. VAN DAM AND K. LANGENDOEN. **An adaptive energy-efficient MAC protocol for wireless sensor networks**. *In Proceedings of the 1st international conference on Embedded networked sensor systems, SenSys'03, New York, USA*, 2003. 34

[75] G. LU, B. KRISHNAMACHARI, AND C.S. RAGHAVENDRA. **An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks**. *In Proceedings of the 18th International in Parallel and Distributed Processing Symposium*, 2004. 34

[76] J. POLASTRE, J. HILL, AND D. CULLER. **Versatile low power media access for wireless sensor networks**. *In Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys'04, New York, USA*, 2004. 34

[77] A. EL-HOIYDI AND J.-D. DECOTIGNIE. **WiseMAC : an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks**. *In Proceedings of the Ninth International Symposium on Computers and Communications, Washington, DC, USA*, 2004. 34

[78] L. BERNARDO, R. OLIVEIRA, M. PEREIRA, M. MACEDO, AND P. PINTO. **A Wireless Sensor MAC Protocol for Bursty Data Traffic**. *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC 2007*, 2007. 34

[79] M. BUETTNER, G. V. YEE, E. ANDERSON, AND R. HAN. **X-MAC : a short preamble MAC protocol for duty-cycled wireless sensor networks**. *In Proceedings of the 4th international conference on Embedded networked sensor systems, SenSys'06*, 2006. 34

[80] R. LE MOIGNE. **Modélisation et simulation basée sur SystemC des systèmes monopuces au niveau transactionnel pour l'évaluation de performances**. *Thèse de doctorat*, 2005. 39

[81] A. SALHIEH, J. WEINMANN, M. KOCHHAL, AND L. SCHWIEBERT. **Power Efficient Topologies for Wireless Sensor Networks**. *International Conference on Parallel Processing*, 2001. 41, 43, 44, 46

[82] H. ZHOU, D. LUO, Y. GAO, AND D. ZUO. **Modeling of Node Energy Consumption for Wireless Sensor Networks**. *Wireless Sensor Network, vol. 3 No 1, 2011, pp 18-23. doi : 10.4236/wsn.2011.31003*, 2011. 41

[83] P. PARVATHI. **Comparative Analisis of CBRP, AODV, DSDV Routing Protocols in Mobile Ad-hoc Networks**. *International Conference on Computing, Communication and Applications (ICCCA)*, IEEE'2012. 41, 46, 65

[84] R. DING AND L. YANG. **A Reactive Geographic Routing Protocol for Wireless Sensor Networks**. *Sixth Internatinal Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, IEEE'2010. 41, 46, 65

[85] Q. WEI-QING. **Cluster head Selection Approach Based on Energy and Distance**. *International Conference on Computer Science and Network Technology*, 2011. 41, 46

[86] Q. WANG, M. HEMPSTEAD, AND W.YANG. **A Realistic Power Consumption Model for Wireless Sensor Network Devices**. *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, 2006. 41

[87] M. A. PASHA, S. DERRIEN, AND O. SENTIEYS. **System-Level Synthesis for Ultra Low-Power Wireless Sensor Nodes**. *13th Euromicro Conference on Digital System Design : Architectures, Methods and Tools (DSD)*, 2010. 41

[88] W. DU, D. NAVARRO, F. MIEYEVILLE, AND I. O'CONNOR. **IDEA1 : A Validated SystemC-Based Simulator for Wireless Sensor Networks**. *Eighth IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, 2011. 41, 57

[89] M. VASILEVSKI, F. PECHEUX, H. ABOUSHADY, AND L. DE LAMARRE. **Modeling Heterogeneous Systems Using SystemC-AMS Case Study : A Wireless Sensor Network Node**. *IEEE International Behavioral Modeling and Simulation Workshop*, 2007. 41

[90] A. COURTAY, A. PEGATOQUET, M. AUGUIN, AND C. CHABAANE. **Wireless Sensor Network Node Global Energy Consumption Modeling**. *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2010. 41

[91] B. KAN, L. CAI, L. ZHAO, AND Y. XU. **Energy Efficient Design of WSN Based on An Accurate Power Consumption Model**. *International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007)*, 2007. 41

[92] A. P. ABIDOYE, N. A. AZEEZ, A. O. ADESINA, AND K. K. AGBELE. **ANCAEE : A Novel Clustering Algorithm for Energy Efficiency in Wireless Sensor Networks**. *Wireless Sensor Network*, **3** :307–312, 2011. 43, 44

[93] W. HEINZELMAN, A. CHANDRAKASAN, AND H. BALAKRISHNAN. **Energy-Efficient Communication Protocol for Wireless Micro sensor Networks**. *In proc of the Hawaii International Conférence on Systems Science*, **8** :8020, 2000. 43, 44

[94] M. N. HALGAMUGE, M. ZUKERMAN, K. RAMAMOHANARAO, AND H. L. VU. **An Estimation of Sensor Energy Consumption**. *Progress In Electromagnetics Reser*, **12** :259–295, 2009. 44

[95] M. A. RAZZAQUE AND S. DOBSON. **Energy-Efficient Sensing in Wireless Sensor Networks Using Compressed Sensing**. *Sensors*, **14** :2, 2013. 45

[96] M. J. MILLER AND N. H. VAIDYA. **A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio**. *IEEE TRANSACTIONS ON MOBILE COMPUTING*, 2005. 46

[97] R. JURDAK, A. G. RUZZELLI, AND G.M.P. O'HARE. **Radio Sleep Mode Optimization in Wireless Sensor Networks**. *IEEE Transactions on Mobile Computing*, 2010. 46

[98] Y. YU, B. KRISHNAMACHARI, AND V. K. PRASANNA. **Data Gathering with Tunable Compression in Sensor Networks**. *IEEE Transactions on Parallel and Distributed Systems, pp : 276-287*, 2008. 46, 65, 100

[99] F. MARCELLONI AND M. VECCHIO. **A Simple Algorithm for Data Compression in Wireless Sensor Networks**. *IEEE Communication Letters*, **13** :6, June 2011. 46, 65, 100

[100] F. CHEN, Y. SHEN, J. LIU, AND F. WEN. **Nonthreshold-Based Node level Algorithm of Data Compression over the Wireless sensor networks**. *2nd International Conference on Signal Processing Systems (ICSPS)*, 2010. 46, 100

[101] B. YING, W. LIU, Y. LIU, H. YANG, AND H. WANG. **Energy-efficient Node-level Compression Arbitration for Wireless Sensor Networks**. *11th International Conference on Advanced Communication Technology (ICACT)*, 2009. 46

[102] R. A. F. MINI, A. A. F. LOUREIRO, AND B. NATH. **A State-based Energy Dissipation Model for Wireless Sensor Nodes**. *10th IEEE Conference on Emerging Technologies and Factory Automation*, 2005. 47

[103] A. SHAREEF AND Y. ZHU. **Energy Modeling of Wireless Sensor Nodes Based on Petri Nets**. *39th International Conference on Parallel Processing*, 2010. 47

[104] D. LOZNEANU, G. PANA, AND E.-L. MIRON. **Energy Model of Sensor Nodes in WSN**. *Review of the Air Force Academy*, **9** :33–38, 2011. 48

[105] A. R. KHAN, S. M. BILAL, AND M. OTHMAN. **A performance comparison of open source network simulators for wireless networks**. *IEEE International Conference on Control System Computing and Engineering*, Nov 2012. 50

[106] X. XIAN, W. SHI, AND H. HUANG. **Comparison of OMNET++ and other simulator for WSN simulation**. *3rd IEEE conference on industrial electronics and applications (ICIEA)*, 2008. 50

[107] G. JOSE MOSES, D. SUNIL KUMAR, P. SURESH VARMA, AND N. SUPRIYA. **A Simulation Based Study of AODV, DSR, DSDV, Routing Protocols in MANET Using NS-2**. *International Journal of Advanced Research in Computer Science and Software Engineering*, March 2012. 50, 51, 58

[108] NITIN G. PALAN AND ADITI P. KHADILKAR. **Media access control protocol modelling for mobile sensor network-using OMNETT++ -MiXiM network simulator**. *International Conference on Sustainable Energy and Intelligent Systems (SEISCON)*, 2011. 52, 58

[109] MEENAKSHI, SATVIKA, AND A. KAUSHIK. **Comparative analysis of handover and traffic classes in UMTS using OPNET simulator for improving**

QoS. *International Conference on Signal Propagation and Computer Technology (ICSPCT)*, 2014. 53

[110] X. ZENG, R. BAGRODIA, AND M. GERLA. **GloMoSim : a library for parallel simulation of large-scale wireless networks**. *Twelfth workshop on Parallel And Distributed Simulation (PADS)*, **1998** :154–161. 53

[111] A. SOBEIH, J. C. HOU, L.-C. KUNG, N. LI, H. ZHANG, W.-P. CHEN, H. Y. TYAN, AND H. LIM. **J-Sim : a simulation and emulation environment for wireless sensor networks**. *Wireless Communications*, **13(4)** :104–119, 2006. 53

[112] S. SUNDRESH, W. KIM, AND G. AGHA. **Sens : A sensor, environment and network simulator**. *In Procedings of the 37th annual symposium on Simulation, IEEE Computer Society*, **2004** :221. 54

[113] D. WEBER, J. GLASER, S. A. MDANI, AND S. MAHLKNECHT. **Power aware simulation framework for wireless sensor networks and nodes**. *EURASIP Journal on Embedded Systems*, **2008**. 54

[114] G. CHEN, J. BRANCH, M. PFLUG, L. ZHU, AND B. SZYMANSKI. **Sense : A wireless sensor network simulator**. *Advances in Pervasive Computing and Networking*, **2005** :249–267. 54

[115] R. NATH. **A TOSSIM based implementation and analysis of collection tree protocol in wireless sensor networks**. *International Conference on Communications and Signal Processing (ICCSP)*, 2013. 55

[116] P. KUGLER, P. NORDHUS, AND B. ESKOFIER. **Shimmer, Cooja and Contiki : A New Toolset for the Simulation of On-node Signal Processing Algorithms**. *IEEE International Conference on Body Sensor Networks (BSN)*, 2013. 55

[117] L. SUN, J. WU, Y. ZHANG, AND H. YIN. **Comparison between physical devices and simulator software for CISCO network technology teaching**. *8th International Conference on Computer Science & Education (ICCSE)*, 2013. 55

[118] A. FRABOULET, G. CHELIUS, AND E. FLEURY. **Worldsens : Development and Prototyping tools for Application Specific Wireless Sensors Networks**. *6th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007. 55, 56

[119] N. FERRY, S. DUCLOYER, N. JULIEN, AND D. JUTEL. **Power/Energy estimator for designing WSN nodes with ambient energy harvesting feature**. *EURASIP Journal on Embedded Systems*, **2011** :17. 57

[120] M. KUORILEHTO, M. KOHVAKKA, M. HÄNNIKÄINEN, AND T. D. HÄMÄLÄINEN. **High Abstraction Level Design and Implementation Framework for Wireless Sensor Network**. *Embedded Computer Systems : Architectures, Modeling, and Simulation Lecture Notes in Computer Science*, **3553** :384–393, 2005. 58

[121] D. JUNG, T. TEIXEIRA, AND A. SAVVIDES. **Sensor node lifetime analysis : Models and tools**. *ACM Transaction on Sensor Networks (TOSN)*, **5** :1–3, 2009. 58

[122] S. PARK, A. SAVVIDES, AND M. B. SRIVASTAVA. **SensorSim : A simulation framework for sensor networks**. *In Procedings of the 3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'00)*, **5** :104–111, 2000. 58

[123] T. W. CARLEY AND R. BARUA. **Sidh : A wireless sensor network simulator**. *Institute for System Research Technical Reports*, 2005. 58

[124] D. NICULESCU AND B. NATH. **Ad Hoc Positioning System (APS) using AoA**. *IEEE INFOCOM 2003*, 2003. 65

[125] A. JAGOE. **Mobile Location Service : The Definitive Guide**. *Upper Saddle River : Prentice-Hall*, 2003. 65

[126] A. SAVVIDES, C.-C. HAN, AND M. B. STRIVASTAVA. **Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors**. *MobiCom 2001*, 2001. 65

[127] J. LAURENT, N. JULIEN, E. SENN, AND E. MARTIN. **Functionnal level power analysis : an efficient approach for modeling the power consumption of complex processors**. *in Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, 2004. 69

Thèse de Doctorat

Aina A. RANDRIANARISAINA

MODÉLISATION DE LA CONSOMMATION D'ÉNERGIE EN VUE DE LA CONCEPTION CONJOINTE (MATÉRIEL/LOGICIEL) DES APPLICATIONS EMBARQUÉES. APPLICATION AUX RÉSEAUX DE CAPTEURS SANS FIL (WSN).

ENERGY CONSUMPTION MODELING TOWARDS COMBINE (HARDWARE/SOFTWARE) DESIGNING OF EMBEDDED APPLICATIONS. APPLICATION TO WIRELESS SENSOR NETWORKS (WSN).

Résumé

Dans le domaine des systèmes embarqués, les contraintes sont souvent liées à l'encombrement du dispositif et à l'énergie consommée par celui-ci. Les applications embarquées sont de plus en plus sensibles à la consommation d'énergie. De celle-ci dépend bien sûr l'autonomie de l'application, mais aussi les performances que l'on peut en attendre. Dès lors, la conception sous contraintes de consommation de systèmes embarqués représente un enjeu essentiel, et la prise en compte des contraintes de consommation d'énergie doit intervenir dès les premières étapes de spécification et de conception d'un produit. Les travaux de cette thèse se placent dans le domaine des réseaux de capteurs, pour lesquels il est nécessaire de définir des stratégies innovantes de conception afin de respecter les contraintes d'embarquabilité.

Notre objectif est de contribuer à la définition de modèles et de méthodes pour l'aide à la conception des architectures logicielles et matérielles de systèmes embarqués en tenant compte de contraintes de consommation d'énergie. Les modèles proposés serviront à la simulation mais aussi au dimensionnement des architectures des systèmes.

Mots clés

Modélisation, Noeud capteurs, Consommation de puissance.

Abstract

In the field of embedded systems, constraints are often related to the size and energy. Embedded applications are becoming more sensitive to energy consumption. In fact, this latter depends on the autonomy of the application, but also its performances. One must take into account the constraints of energy consumption in the early stages of the specification and the design of a product. Therefore, the design of embedded systems constrained consumption is a key issue facing the need to control energy costs. For example, in the field of sensor networks, this aspect is a major constraint for which it is necessary to define innovative design strategies touching on all aspects involved in these systems.

Our goal is to contribute to the definition of models and hardware and software architectures design methods of embedded systems, taking into account the constraints of energy consumption. The proposed models will be used for simulation but also to the design of systems architectures.

Key Words

Modelization, Sensor node, Power consumption.