# Multidimensional cross-level tool for Wireless Sensor Networks constrained by energy

Michel Bakni

THÈSE PRÉSENTÉE POUR OBTENIR LE GRADE DE

# DOCTEUR DE L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE : SCIENCES PHYSIQUES ET DE L'INGÉNIEUR

SPÉCIALITÉ : ÉLECTRONIQUE

THÈSE PRÉPARÉE à l'ESTIA PAR :

**Michel BAKNI**

# Outil de dimensionnement trans-niveaux de réseaux de capteurs sans fil contraints en énergie

Sous la direction de : Octavian CUREA, Professeur, ESTIA

Co-encadrée par : Guillaume TERRASSON, Enseignant-chercheur, ESTIA

Date de soutenance : 25 Mars 2021

**Membres du jury :**

| | | |
|---|---|---|
| Najiba MRABET BELLAAJ | Professeur, Inst. Supérieur d'Informatique, Tunisie | Examinateur |
| Cécile BELLEUDY | HDR, Univ. de Nice-Sophia Antipolis, France | Rapporteur |
| Abdallah MAKHOUL | Professeur, Univ. de Franche-Comté, France | Président, rapporteur |
| Philippe ROOSE | HDR, Univ. Pau et des Pays de l'Adour, France | Examinateur |
| Renaud BRIAND | Directeur R&D Aquitaine Électronique, Serres-Castet, France | Invité |

# Acknowledgement

These are humble words I am writing to express my gratitude towards persons close to my heart. I am thankful they helped me climbing this hill, I will always be, until the end of my days.

I express my gratitude to Mrs. Cécile BELLEUDY, HDR at University of Nice-Sophia Antipolis and Mr. Abdallah MAKHOUL, professor at University of Franche-Comté, for giving me the honour of being the reporters of this work, of the interest shown, and of having read and judged my work. Your relevant suggestions and the various questions contributed to the improvement of this study.

In the same way, all my thanks go to Mrs. Najiba MRABET BELLAAJ, professor at “Institut Supérieur d'Informatique” of Tunis and Mr. Philippe ROOSE, HDR at University of “Pau et des Pays de l’Adour”, for their role as examiners during my defense, your scientific questions and comments have improved this manuscript.

I am grateful that I have the chance to work with my directors: Octavian CUREA and Guillaume TERRASSON. Serving under their guidance helped me coin my personality and develop my character to become the man I am today. I would like to mention their patient and the leading decisions they made in the darkest hours. Without their support, finishing this thesis would not have been possible.

I would also like, proud and with honour, to address all my teachers, private tutors and whomever taught me anything worth learning, I shall be forever indebted for that.

I dedicate this thesis to exiles, refugees and all people who were forced to quit their homes, leave their lives behind and become foreigners in God's land, because of the ideas they believe and their strong will to defend it.

Finally, I am also to mention my parents and my sisters for their endless love and my friends, here in France, especially Mr. Yannick RENAUD for his efforts to make me feel home in my foreignness.

# Titre : Outil de dimensionnement trans-niveaux de réseaux de capteurs sans fil contraints en énergie

**Résumé :** Un Réseau de Capteurs Sans Fil (RCSF) est composé d'un ensemble de nœuds alimentés par batterie, associant des capteurs à une unité de traitement et à un émetteur-récepteur sans fil. De nos jours, les RCSF font l'objet de nombreux travaux de recherche et développement. En effet, ces réseaux constituent une solution intéressante afin d'apporter une réponse à différents enjeux sociaux, sociétaux et économiques tels que le déploiement des Smart Grids ou la supervision à distance de la santé de personnes dans un contexte de vieillissement de la population. Ces RCSF ont la capacité d'être déployé dans des environnements contraints, tels que les forêts, les volcans et les bâtiments, pour atteindre divers objectifs : la localisation d'objets et d'individus mais aussi la surveillance de phénomènes physiques, comme les signaux physiologiques de patients ou la température ambiante d'un bâtiment.

Cependant, le déploiement de ces RCSF est souvent rendu critique en raison des contraintes imposées par l'environnement d'application, par exemple, les températures pouvant être subies lors de la supervision d'un volcan, ou la difficulté d'accès aux nœuds lorsque ceux-ci sont utilisés pour la surveillance de l'état structurel d'une autoroute ou d'un bâtiment. Par conséquent, les chercheurs et développeurs de ce type de réseau ont besoin d'outils pour tester et évaluer, dans le cadre du processus de conception d'un RCSF, les performances des nœuds et du réseau avant de le déployer dans un environnement réel.

Dans ce contexte, les outils de simulation peuvent apporter une solution permettant un gain de temps, une limitation des coûts et des efforts avant le déploiement d'un tel réseau dans son environnement d'utilisation. Ces outils sont donc aujourd'hui largement répandus et utilisés pour évaluer les performances d'un RCSF mais aussi les propositions issues de travaux de recherche relatifs à ce type de réseau. Néanmoins, la conception d'un RCSF, dédié à une application, doit tenir compte de sa structure multi-niveau : topologie, nœuds et circuits. Ainsi, pour aborder les principaux défis liés au RCSF, tels que la problématique d'autonomie énergétique des nœuds, la modélisation de ces réseaux est considérée comme une tâche complexe car l'approche de modélisation utilisée doit considérer sa structure multi-niveau afin de fournir des résultats exploitables provenant simultanément de niveaux d'abstraction différents.

Dans cette thèse, nous définissons, proposons et implémentons un modèle trans-niveaux orienté énergie dédié au RCSF permettant de considérer simultanément différents niveaux d'abstraction : topologie, nœuds et circuits. Ce modèle est capable de tracer la consommation énergétique à partir de différents points de vue : l'activité spécifique d'un circuit, les activités d'un circuit ou d'un nœud, ainsi que son impact sur la durée de vie du RCSF. Notre modèle est ensuite implémenté dans un simulateur de RCSF dédié afin, en considérant différents scénarios, de comparer les résultats obtenus avec un simulateur existant et des nœuds réels dans le but de valider la pertinence de notre approche.

**Mots clés :** Réseaux de capteurs sans fil, outils de simulation, approche de modélisation, trans-niveaux, orientée énergie

# Title: Multidimensional cross-level tool for Wireless Sensor Networks constrained by energy

**Abstract:** Wireless Sensor Network (WSN) is a set of battery-powered nodes that include sensors coupled with processing units and wireless transceivers. Nowadays, WSN is a major topic in the research and development domain. Indeed, it constitutes an interesting solution to give an answer to different situations related to social, societal and economic issues such as the need to manage the Smart Grids or to supervise patient's health in the context of the aging population. This kind of network has the capacity to be simply deployed in harsh environments, such as forests, volcanoes and buildings, to achieve various goals, like tracking targets, animals or human beings for example, or monitoring physical phenomena, such as patient physiological signals or ambient temperature in a building.

However, the deployment of WSNs can be critical because of the difficult conditions imposed by the application environment, for example, the high temperatures in the case of volcano activity supervision, or the impossibility of reaching the nodes after deployment, when the WSN must be used to structural health monitoring of a highway or a building. Therefore, researchers and developers need tools to test and evaluate, in the design process of a WSN, node and network performances before deploying it in real surroundings.

In this context, simulation can provide a solution that can save time, cost, and effort before deploying a WSN application in its real environment. This explains that simulation tools are widely used in WSN designing stages and for research works evaluation related to this kind of network. Nevertheless, designing a WSN, dedicated to a specific application, needs to address its multilevel structure: topology, nodes and circuits. Thus, to handle the main challenges of WSN design such as energy issues, WSN modelling is considered a complex task because the adopted modelling approach has to take into account the WSN multilevel structure in order to provide exploitable results from different points of view at the same time.

In this thesis, we define, propose and implement a cross-level energy-aware model for WSN that allows considering different levels of abstraction at the same time: circuits, nodes and topology. This energy-oriented model is able to trace the energy consumption from multiple points of view: a specific circuit's activity, circuit or node activities, as well as the impact on the WSN lifetime. The proposed model is implemented in a dedicated WSN simulator, which is used, defining different scenarios, to compare obtained results with a well-known simulator and physical WSN nodes with the aim to validate the relevance of our approach.

**Keywords:** Wireless Sensor Networks, simulation tools, modelling approach, cross-level, energy-awareness

## Unité de recherche

ESTIA Recherche, n°RNSR 201420655, Technopole Izarbel, 90 Allée Fauste d'Elhuyar, 64210 Bidart - FRANCE

# Résumé de la thèse

Avec l'avancée des nouvelles technologies de communication, faisant naitre le concept de l'Internet des Objets, et l'évolution des systèmes électroniques, les Réseaux de Capteurs Sans Fil (RCSF) apparaissent aujourd'hui comme une solution potentielle en réponse à de nombreux enjeux sociétaux tels que le vieillissement de la population, la préservation de l'environnement, une meilleure gestion de nos énergies... En effet, ces RCSF ont la capacité d'être déployés dans des environnements contraints, tels que les forêts, les volcans et les bâtiments, pour atteindre divers objectifs : la localisation d'objets et d'individus mais aussi la surveillance de phénomènes physiques, comme les signaux physiologiques de patients ou la température ambiante d'un bâtiment afin de la réguler plus efficacement. Néanmoins, le déploiement et l'utilisation de ces RCSF, comme partie prenante de l'Internet des Objets mais aussi de Systèmes Cyber-physiques plus complexes, restent confrontés à de multiples challenges et verrous technologiques, faisant l'objet de nombreuses activités de recherche : la collecte, la gestion et le traitement des données (Big Data), la sécurisation de ces données, l'interopérabilité (entre différents systèmes, réseaux de communication et même composants), l'autonomie énergétique de ces objets et leur adaptabilité à l'application visée en termes d'encombrement, poids et forme.

Dans ce contexte, les outils de simulation peuvent apporter une solution permettant un gain de temps, une limitation des coûts et des efforts avant le déploiement d'un tel réseau dans son environnement d'utilisation. Par conséquent, ces outils sont aujourd'hui largement répandus et utilisés afin d'aider à l'évaluation des performances d'un RCSF ainsi qu'à la validation des solutions proposées en réponse aux challenges soulevés précédemment. Cependant, le développement de ces outils de simulation et d'évaluation est aussi confronté à de nombreux verrous techniques. Par exemple, pour aborder les principaux défis liés aux RCSF, tels que la problématique d'autonomie énergétique des nœuds, l'approche de modélisation utilisée, au sein d'un outil de simulation, doit considérer la structure multi-niveau d'un RCSF (application, topologie, nœuds et circuits) mais aussi l'hétérogénéité potentielle des nœuds le composant afin de fournir des résultats provenant simultanément de niveaux d'abstraction différents.

L'objectif des travaux de recherche, menés dans le cadre de cette thèse, est de contribuer à la levée de certains de ces verrous afin d'initier le développement d'un outil de dimensionnement multi-niveaux de réseaux d'objets connectés, plus particulièrement de RCSF, dédié aux concepteurs de ces applications et donc de donner une continuité aux travaux déjà initiés, au

sein d'ESTIA-Recherche, sur les modèles d'énergie d'un nœud de réseau de capteurs. Ces travaux de recherche sont cofinancés par la Région Nouvelle Aquitaine et l'ESTIA, dans le cadre du projet OUDINI, projet porté par ESTIA-Recherche.

Ce mémoire de thèse se compose de quatre chapitres et adopte le plan de recherche suivant :

- Identifier les verrous et problématiques auxquels est confrontée la modélisation de RCSF ;
- Proposer et implémenter de nouveaux concepts et solutions en réponse aux verrous et problématiques précédemment identifiés ;
- Mettre en œuvre, évaluer et valider les concepts et solutions proposées.

**Chapitre 1 – Les Réseaux de Capteurs Sans Fil et leur simulation**

Dans le contexte énoncé précédemment, le Chapitre 1 de ce document dresse, en premier lieu, un état de l'art des Réseaux de Capteurs Sans Fil. Ces RCSF sont aujourd'hui destinés à de nombreuses applications souvent répertoriées en deux catégories principales : la localisation d'objets ou d'individus et la surveillance de phénomènes physiques. Un RCSF est défini comme un réseau ad-hoc composé de nœuds à ressources limitées, en termes d'énergie et de capacité de traitement, déployés dans des environnements plus ou moins contraints. Un nœud de RCSF est normalement composé d'un ou plusieurs capteurs, d'une unité de traitement, d'un émetteur-récepteur mais aussi d'une ou plusieurs sources d'énergie. Du fait de la multiplicité des solutions disponibles et en cours de développement, notamment en termes de composants et de technologies de communication, les nœuds d'un RCSF adoptent des architectures matérielles et logicielles, parfois hétérogènes au sein d'un même réseau, dépendantes de l'application visée et peuvent jouer différents rôles (nœuds capteurs, nœuds relais, nœuds « cluster head » …) en fonction de la topologie déployée. Par conséquent, il est démontré que le développement de ce type de réseau peut être considéré comme une tâche complexe, pour laquelle les outils de simulation constituent une solution pour évaluer plus rapidement les performances d'un RCSF avant son déploiement.

Partant de ce constat, nous nous focalisons, dans la deuxième partie de ce Chapitre 1, sur les simulateurs dédiés aux RCSF. Aujourd'hui, de nombreux simulateurs de RCSF sont disponibles. L'intérêt de ces simulateurs est de permettre, sans nécessité d'implémentation ou de déploiement réel, d'évaluer un RCSF, composé de centaines ou milliers de nœuds, en considérant l'application visée, différents scénarios de fonctionnement et le potentiel environnement de déploiement. Dans cette optique, l'approche de modélisation implémentée

au sein du simulateur joue un rôle important afin de fournir des résultats fiables et exploitables. Par conséquent, un aperçu de l'évolution de ces approches de modélisation utilisées au sein des simulateurs de RCSF est ensuite fourni. Cet aperçu nous permet de définir trois catégories principales d'approche de modélisation : simple niveau, multi-niveau et trans-niveaux. De plus, l'autonomie énergétique des nœuds d'un RCSF étant une problématique majeure, nous démontrons que les simulateurs existants peuvent être classifiés comme « non-orientés énergie » ou « orientés énergie ». Il est observé que les modèles d'énergie implémentés au sein des simulateurs « orientés énergie » suivent les mêmes approches de modélisation que celles mentionnées précédemment.

La synthèse de ce Chapitre 1 nous amène à définir qu'une approche de modélisation trans-niveaux est nécessaire pour fournir des outils de simulation décrivant précisément le comportement, notamment au niveau énergie, d'un RCSF à différents niveaux d'abstraction : application, topologie, nœuds et circuits. Cette synthèse permet de déterminer la question de recherche qui sera adressée dans le cadre de ce mémoire : Comment définir et implémenter un modèle de RCSF, tenant compte des aspects énergétiques, suivant l'approche de modélisation trans-niveaux ?

**Chapitre 2 – Les simulateurs de Réseau de Capteurs Sans Fil : approche de modélisation et considération des aspects énergétiques**

Dans ce second chapitre, afin de déterminer plus précisément les verrous relatifs à l'implémentation d'une approche de modélisation trans-niveaux et à la modélisation des aspects énergétiques au sein des simulateurs de RCSF, une analyse plus précise de simulateurs existants est nécessaire.

Dans un premier temps, du fait de la grande diversité des simulateurs existants, nous proposons d'étudier et de comparer plus précisément les caractéristiques d'une sélection de simulateurs. Sur la base d'une étude statistique, s'appuyant sur le nombre de citations et réalisée sur un ensemble d'articles issus d'une recherche par mots clés, relatifs au sujet de nos travaux de thèse, les simulateurs retenus pour cette étude sont les suivants : NS2, TOSSIM, OMNet++ et IDEA1. Par la suite, pour procéder à l'évaluation de ces simulateurs, nous avons défini un ensemble de critères qualitatifs et quantitatifs, en nous inspirant de critères issus d'études comparatives disponibles dans la littérature scientifique mais aussi de critères spécifiques définis au Chapitre 1, tels que l'approche de modélisation implémentée mais aussi l'existence d'un modèle d'énergie.

La synthèse de cette étude démontre, d'une part, que l'approche trans-niveaux n'est pas totalement implémenté dans les simulateurs étudiés et d'autre part, qu'aucun d'entre eux n'intègre un modèle d'énergie complet. En effet, NS2, OMNeT++ et TOSSIM sont des simulateurs implémentant une approche de modélisation considérant un seul niveau d'abstraction. NS2 et OMNeT++ sont des simulateurs généralement utilisés pour évaluer des algorithmes et protocoles réseaux. TOSSIM, quant à lui, est un émulateur matériel limité à des applications utilisant le système d'exploitation embarqué nommé TinyOS. Concernant les aspects énergétiques, NS2 et OMNeT++ proposent un modèle d'énergie très simple tenant seulement compte de l'activité de l'unité de communication. De son côté, TOSSIM n'intègre pas directement de modèle d'énergie. Enfin, IDEA1 adopte une approche de modélisation multi-niveau permettant l'interaction de paramètres appartenant à divers niveaux d'abstraction mais sans implémenter totalement une approche trans-niveaux. Ce simulateur intègre un modèle d'énergie plus complet que les autres simulateurs étudiés mais comportant encore des limitations.

Ces constats nous ont permis d'identifier un ensemble de challenges, relatifs à l'implémentation d'une approche de modélisation trans-niveaux et à la modélisation de l'énergie au sein d'un RCSF, à adresser, dans le but de proposer un modèle de RCSF trans-niveaux et orienté énergie.

**Chapitre 3 – Modèle de RCSF trans-niveaux orienté énergie**

Le Chapitre 3 est consacré à la définition, à la description et à l'implémentation d'un modèle de RCSF trans-niveaux orienté énergie, issu de ces travaux de thèse, permettant d'apporter des réponses aux challenges identifiés au Chapitre 2.

Dans un premier temps, la définition de plusieurs concepts structurels, à la base de l'approche de modélisation « cross-level » (trans-niveaux) proposée et implémentée dans ce modèle, tels que l'objet, le niveau et les types d'interaction, intra-niveau et trans-niveau, est fournie. Pour renforcer l'intérêt de notre proposition, en considérant le niveau nœud, il est démontré, pour chaque unité composant un nœud (unités capteurs, de traitement ou de transmission), les possibles interactions intra et trans-niveaux existantes. De la même manière, les concepts temporels de phase et de « pattern », proposés pour modéliser les aspects énergétiques d'un RCSF, sont définis.

Ensuite, basés sur ces concepts et sur la démonstration des interactions existantes entre paramètres au sein d'un RCSF, notre modèle trans-niveaux orienté énergie et ses particularités sont présentés. Ce modèle est divisé en quatre niveaux d'abstraction distincts : application,

topologie, nœuds et circuits. Chaque niveau d'abstraction inclut des objets paramétrables pouvant interagir entre eux. L'implémentation structurelle de notre modèle permet, de plus, de tenir compte de l'hétérogénéité potentielle des nœuds composants un RCSF. En effet, il est possible de définir plusieurs types de nœuds, donc composés de circuits différents, au sein d'un même réseau. Au niveau comportemental, il est décrit comment sont modélisés chaque unité pouvant composer un nœud. Il est ainsi démontré comment sont utilisés les concepts de phase et de « patterns » pour décrire l'activité, temporelle et énergétique, aussi bien au niveau d'un circuit qu'au niveau d'un nœud, associant plusieurs circuits (ou unités).

Enfin, d'un point de vue énergétique, notre modèle, contrairement à ceux issus des simulateurs étudiés au Chapitre 2, intègre tous les consommateurs composants un nœud et décrit l'interaction des paramètres entre chacun des différents niveaux : cas d'utilisation, topologie, nœud et circuit. De plus, au sein d'un nœud, la batterie, en tant qu'unité fournissant l'énergie, est généralement connectée aux différentes unités composant le nœud au travers de régulateurs de tension, linéaires ou à découpage. Les caractéristiques de ces régulateurs pouvant avoir un impact sur l'énergie consommée, notre modèle tient compte de ceux-ci afin de pouvoir déduire plus précisément l'autonomie d'un nœud mais aussi la durée de vie du réseau/de l'application.

**Chapitre 4 – Validation du modèle de RCSF proposé**

Le Chapitre 4 est consacré à la validation du modèle proposé. L'objectif de cette phase de validation est de démontrer l'intérêt de l'approche de modélisation considérée mais aussi des solutions intégrées à ce modèle permettant de simuler plus précisément la consommation énergétique d'un nœud, et donc d'estimer la durée de vie du RCSF.

Pour ce faire, dans un premier temps, un noyau de simulateur, basé sur une approche orientée objet, a été développé, à l'aide de l'environnement MATLAB, pour implémenter notre modèle. Ce simulateur permet de configurer, soit à partir d'une interface, soit par l'importation de fichiers XML, différents scénarios d'application intégrant des RCSF à nœuds homogènes ou hétérogènes. Une interface graphique permet l'affichage des résultats, tels que la visualisation de la topologie du RCSF configuré mais aussi le détail de la consommation d'un nœud ou d'un circuit spécifique de celui-ci.

En définissant différents scénarios, la mise en œuvre de ce simulateur va permettre, dans un second temps, la validation du modèle de RCSF trans-niveaux orienté énergie. Cette validation est effectuée en trois étapes.

Premièrement, il est nécessaire de valider l'approche de modélisation « cross-level » suggérée et les concepts proposés. En utilisant des scénarios spécifiques, les résultats obtenus avec le simulateur nous ont permis de mettre en évidence, au sein de notre modèle énergétique, de nombreuses interactions inter-niveaux et trans-niveaux. Nous avons ainsi pu tester l'influence de plusieurs paramètres (la distance entre les nœuds, le seuil de fragmentation et la consommation d'énergie) afin de valider les concepts implémentés dans notre modèle.

Puis, afin de démontrer les capacités de notre modèle par rapport à ceux implémentés dans un des simulateurs étudiés dans le Chapitre 2, les résultats de notre simulateur sont comparés avec les résultats obtenus à partir de NS2, en implémentant sur chacun des deux simulateurs un même scénario. Ainsi, en considérant seulement les activités de l'unité de communication, la comparaison avec les résultats de NS2 montre que l'écart entre NS2 et notre modèle n'excède pas 3,5%. De plus, sur la base du même scénario, nous démontrons, que contrairement à NS2, notre modèle permet de comparer différentes configurations en considérant l'ensemble des unités composants un nœud. Ces résultats peuvent être exploités, en phase de conception amont, par un concepteur de RCSF afin de l'aider dans ces choix de composants matériels ou logiciels.

Enfin, afin de valider la précision de notre modèle, nous comparons les résultats obtenus à partir du simulateur avec les résultats issus de véritables nœuds de RCSF. Ces résultats montrent que le modèle proposé, intégré dans le simulateur développé, permet d'obtenir, en considérant quelques ajustements liés au fonctionnement réel de l'émetteur-récepteur des nœuds mis en œuvre, un comportement de cette unité de transmission proche de celui obtenu par l'expérimentation.

**Conclusion générale**

Dans le cadre de ces travaux de thèse, nous nous sommes intéressés à la modélisation et à la simulation, plus particulièrement, de l'énergie au sein des Réseaux de Capteurs Sans Fil.

Basé sur une approche de modélisation trans-niveaux, un modèle de RCSF orienté énergie a été proposé et implémenté dans un simulateur propre, développé en utilisant l'environnement MATLAB. Ce modèle décomposé en quatre niveaux (application, topologie, nœud, circuit) repose sur la définition d'un ensemble de concepts structurels (objet, paramètres) et temporels (phase, « pattern »). L'implémentation de ces concepts offre la capacité aux paramètres de chacun des objets, composants ce modèle, de pouvoir interagir, en fonction de leur interdépendance, entre eux, permettant ainsi des interactions inter-niveau et trans-niveaux. Ainsi, il est possible, en utilisant ce modèle trans-niveaux, d'étudier simultanément l'influence de paramètres, appartenant à différents niveaux d'abstraction, sur le « pattern » traçant l'activité

d'un nœud (ou d'un circuit) spécifique et donc sur la consommation énergétique de ce même nœud. De plus, dans notre modèle, l'impact des régulateurs de tension, pouvant être associés à chaque circuit, est pris en compte afin d'obtenir une estimation plus fine de l'autonomie d'un nœud voire de la durée de vie du RCSF considéré dans son entièreté.

Premièrement, en paramétrant des scénarios simples, les résultats obtenus, à l'aide du simulateur implémentant le modèle de RCSF trans-niveaux orienté énergie issu de ces travaux de thèse, nous ont permis de valider le fonctionnement de l'approche de modélisation proposée. Par la suite, les résultats issus de notre simulateur ont été confrontés avec ceux obtenus via NS2, un simulateur réseau très largement utilisé dans la recherche sur les RCSF. La déviation maximale, entre les résultats issus de notre modèle orienté énergie et ceux du modèle d'énergie implémentée dans NS2, est inférieure à 3,5 %, ce qui parait très acceptable. De plus, il a été démontré que, contrairement à NS2, le modèle proposé permet de tracer l'influence, sur la consommation énergétique d'un nœud, de l'activité non seulement de l'unité de communication mais aussi des unités capteur et de traitement. Enfin, de premiers résultats expérimentaux permettant de relever le « pattern » de consommation d'une unité de communication d'un nœud spécifique confrontés à ceux issus de notre simulateur nous ont démontré que ceux-ci étaient très proches. Ainsi, l'ensemble des résultats obtenus ont permis de valider notre modèle et de démontrer l'intérêt de l'approche de modélisation trans-niveaux adoptée, notamment pour fournir un outil d'aide à la conception de RCSF généralement contraint par des ressources limitées en énergie.

Toutefois, du fait que le sujet de la modélisation et de la simulation de RCSF est vaste et complexe, de nombreux travaux de recherche restent encore à mener pour améliorer et compléter nos propositions.

- Certains objets de notre modèle, comme celui concernant l'unité de traitement mais aussi celui modélisant le canal de transmission, doivent être améliorés afin de pouvoir mieux étudier l'impact de ces éléments d'un RCSF sur son comportement et sa durée de vie.
- Il est aussi nécessaire de pouvoir simuler des scénarios plus complexes. En effet, les activités des nœuds d'un RCSF peuvent être non périodiques (liés à des événements) mais aussi avec des périodes décalées, par exemple pour un nœud comportant deux capteurs prenant des mesures avec des périodes différentes. De ce fait, une autre amélioration majeure, impactant les aspects temporels, de notre modèle consisterait à

trouver des solutions afin d'implémenter le concept d'événement et de pouvoir construire des « patterns » plus complexes.

- Les interfaces de paramétrage et de visualisation des résultats du simulateur proposé doivent être complétés afin de faciliter la simulation de divers scénarios mais aussi l'analyse des résultats obtenus à différents niveaux d'abstraction.
- La validation de ce modèle et des amélioration apportées doivent faire l'objet d'une confrontation plus approfondie avec des résultats expérimentaux.

# Table of Content

Table of Figures ..... xv

Table of Tables ..... xix

List of abbreviations ..... xx

INTRODUCTION ..... 1

Chapter 1. Wireless Sensor Networks and their simulation ..... 5

1.1. Introduction ..... 6

1.2. WSN applications ..... 7

1.2.1. Tracking applications ..... 7

1.2.2. Monitoring application ..... 8

1.3. WSN description ..... 10

1.3.1. WSN architecture ..... 10

1.3.2. Node ..... 11

1.3.3. Network ..... 15

1.3.4. Main challenges in WSN design ..... 24

1.4. Approach and energy-awareness in WSN simulation ..... 26

1.4.1. WSN modelling approaches ..... 26

1.4.2. Energy modelling in WSN simulators ..... 33

1.5. Conclusion ..... 34

Chapter 2. Wireless Sensor Networks simulators: modelling approach and energy awareness ..... 36

2.1. Introduction ..... 37

2.2. Selection of WSN simulators ..... 38

2.2.1. Search and categorisation of papers ..... 38

2.2.2. Statistical analysis ..... 42

2.3. Evaluating WSN simulators ..... 43

2.3.1. Evaluation criteria ..... 43
2.3.2. Proposed methodology for criteria application ..... 46
2.3.3. Methodology implementation and obtained results ..... 46
2.4. Conclusion ..... 67
Chapter 3. Cross-level energy-aware modelling for WSN ..... 71
3.1. Introduction ..... 72
3.2. Cross-level modelling approach ..... 72
3.2.1. Concept definitions ..... 73
3.2.2. Cross-level model overview ..... 74
3.2.3. Cross-level interactions in and between the nodes ..... 76
3.2.4. Model description ..... 80
3.3. Energy-awareness modelling ..... 90
3.3.1. Regulator modelling ..... 90
3.3.2. Node lifetime modelling ..... 92
3.4. Conclusion ..... 95
Chapter 4. Model validation with simulators and real nodes ..... 96
4.1. Introduction ..... 97
4.2. Simulator description and capabilities ..... 98
4.2.1. General overview ..... 98
4.2.2. Model implementation ..... 99
4.2.3. Scenario creation ..... 104
4.2.4. Simulation and analysis stages ..... 107
4.2.5. Other functionalities of the simulator ..... 111
4.3. Comparison with NS2 ..... 116
4.3.1. General settings ..... 117
4.3.2. Simulations using Telecommunication Unit only ..... 118

4.4. Validation with real nodes ..... 122
4.4.1. Description of the circuits ..... 122
4.4.2. Results ..... 126
4.5. Conclusion ..... 129
General Conclusion ..... 131
Appendix 1: List of Articles resulted from the first phase of methodology in chapter 2 ..... 135
Appendix 2: Electrical specifications for the circuits ..... 140
References ..... 145

# Table of Figures

Figure 1-1: Applications of Wireless Sensor Networks [YIC08] ..... 7

Figure 1-2: Typical WSN architecture ..... 11

Figure 1-3: Architecture of WSN as illustrated in [NAV14a] ..... 11

Figure 1-4: WSN node architecture proposed in [AKY02] ..... 13

Figure 1-5: WSN node architecture proposed in [BOU18] ..... 13

Figure 1-6: Software-oriented architecture for WSN node ..... 15

Figure 1-7: Different types of topologies for WSNs ..... 16

Figure 1-8: Different layered-stack models for WSNs ..... 18

Figure 1-9: Comparison between the TCP/IP model and the proposed model ..... 19

Figure 1-10: Different interaction approaches between network stack layers (a) Traditional stack of layers TCP/IP model, (b) a cross-layer model for ad-hoc networks ..... 27

Figure 1-11: WSN modelling approach proposed in [AKY02] ..... 28

Figure 1-12: Emerging to multilevel modelling approach, the WSN stack is two-level stack proposed in [KUO05] ..... 28

Figure 1-13: The “tier” concept ..... 29

Figure 1-14: The 3-level model for WSN proposed in [WAN10a] ..... 30

Figure 1-15: Classification of WSN models based on the modelling approach ..... 31

Figure 1-16: Multilevel architecture adopted by Jsim, where environment parameters can affect the node through the wireless and sensor channels ..... 32

Figure 1-17: Energy-aware cross-level classification for WSN modelling approaches ..... 33

Figure 2-1: Proposed research approach ..... 37

Figure 2-2: Block diagram for the selection process ..... 38

Figure 2-3: Distribution of citations among the simulators ..... 42

Figure 2-4: Citations in Group 3 ..... 43

Figure 2-5: Modelling approach in NS2 ..... 48

Figure 2-6: UML Class diagram for the energy consumption model for a node in NS2 ..... 49
Figure 2-7: The machine state diagram for NS2 energy model ..... 51
Figure 2-8: OMNeT++ Design concept ..... 53
Figure 2-9: Energy model structure in OMNeT++ ..... 54
Figure 2-10: Energy storage models UML class diagram in OMNeT++ ..... 55
Figure 2-11: Energy Generation class diagram in OMNeT++ ..... 56
Figure 2-12: Energy consuming model class diagram in OMNeT++ ..... 58
Figure 2-13: Full energy model for OMNeT++ ..... 59
Figure 2-14: TOSSIM modelling approach ..... 61
Figure 2-15: Class diagram for objects related to energy consumption in TOSSIM ..... 62
Figure 2-16: IDEA1 modelling approach ..... 64
Figure 2-17: Schematic diagram for the structure of IDEA1 ..... 65
Figure 2-18: State-machine diagram for Processing Unit in IDEA1 ..... 66
Figure 2-19: State-machine diagram for Telecommunication Unit in IDEA1 ..... 67
Figure 3-1: Structural concepts used in the proposed model ..... 73
Figure 3-2: Concept of the pattern used in the proposed model ..... 74
Figure 3-3: The Proposed Cross-level model for Wireless Sensor Networks ..... 75
Figure 3-4: General time sequence considered to define Processing Unit activities ..... 76
Figure 3-5: Parameter-based pattern construction for the Processing Unit ..... 77
Figure 3-6: General time sequence considered to define Telecommunication Unit activities. 78
Figure 3-7: Parameter-based pattern construction for the Telecommunication Unit ..... 78
Figure 3-8: General time sequence considered to define Sensing Unit activities ..... 79
Figure 3-9: Parameter-based pattern construction for the Sensing Unit ..... 80
Figure 3-10: Object diagram for the proposed model ..... 81
Figure 3-11: Typical time sequence for the Processing Unit in the proposed model ..... 84
Figure 3-12: Proposed protocol stack for the Telecommunication Unit ..... 86

Figure 3-13: Time sequence for the transmission without acknowledgment ..... 88

Figure 3-14: Time sequence for the transmission of two sequential frames using acknowledge ..... 88

Figure 3-15: Simple time sequence for a Telecommunication Unit ..... 89

Figure 3-16: Time sequence for Sensing Unit with two measurement activities ..... 90

Figure 3-17: Schematic illustration of the regulator connections ..... 91

Figure 3-18: Schematic model for the node with different types of regulators ..... 94

Figure 4-1: Three stages process of the proposed simulator ..... 99

Figure 4-2: Scenario construction approaches ..... 99

Figure 4-3: XML tree of parameters for the Scenario object ..... 100

Figure 4-4: XML tree of parameters for the Topology object ..... 100

Figure 4-5: XML tree of parameters for the Node object ..... 101

Figure 4-6: XML tree of parameters for the Node Type object ..... 101

Figure 4-7: XML tree of parameters for the Processing Unit object ..... 102

Figure 4-8: XML tree of parameters for the Regulator object ..... 102

Figure 4-9: XML tree of parameters for the Telecommunication Unit object ..... 103

Figure 4-10: XML tree of parameters for the Protocol object ..... 103

Figure 4-11: XML tree of parameters for the Sensing Unit object ..... 104

Figure 4-12: XML tree of parameters for the Battery object ..... 104

Figure 4-13: Node A time sequence for the proposed scenario ..... 106

Figure 4-14: Simulator output at the topology level ..... 107

Figure 4-15: Distribution of the energy consumption. ..... 108

Figure 4-16: Part of Node A consumption pattern ..... 108

Figure 4-17: Part of Node B consumption pattern ..... 109

Figure 4-18: Energy consumption in each circuit of each node ..... 110

Figure 4-19: Sequential interactions showing the impact of the node positions on the system lifetime. ..... 112

Figure 4-20: One pattern energy consumption of the Telecommunication Unit exchange phase as a function to distance ..... 113

Figure 4-21: The fragmentation algorithm used in our model ..... 114

Figure 4-22: Per-phase energy consumption from the fragmentation scenario for the Telecommunication Unit. ..... 115

Figure 4-23: Pattern of the Fragmentation Scenario for the Node A ..... 116

Figure 4-24: Pattern of the Fragmentation Scenario for the Node B ..... 116

Figure 4-25: Energy consumption Node B patterns for the two considered wireless link protocols with Payload length = 100 Bytes and $F_p$ = 1 Hz ..... 118

Figure 4-26: Relative error between our simulator and NS2. ..... 120

Figure 4-27: The equal-energy curve as a function to $F_p$ and the payload size for the scenario limited only to the Telecommunication Unit ..... 121

Figure 4-28: Schematic structure for the equipment used in the validation stage ..... 122

Figure 4-29: Block diagram for the WSN node ..... 123

Figure 4-30: Pin connections between PIC18F6420 and MRF24J40 ..... 124

Figure 4-31: Block diagram for the current measurement ..... 124

Figure 4-32: Electronical diagram for the amplification circuit ..... 125

Figure 4-33: Adaptive multiplexing for the amplification stages outputs ..... 126

Figure 4-34: Node patterns for the experimental scenario ..... 127

Figure 4-35: Two measures for the delay before the transmission. ..... 127

Figure 4-36: Time delay imposed in the scenario ..... 128

Figure 4-37: Telecommunication unit consumption pattern (a) measured from the physical node (b) simulated in our simulator ..... 129

Figure 4-38: Energy consumption by phase for the Telecommunication Unit ..... 129

# Table of Tables

Table 1-1: Main characteristics of several link-layer standards adapted to WSN ..... 20

Table 1-2: comparison between WSN applications main categories ..... 25

Table 2-1: Results obtained for NS2 ..... 47

Table 2-2: Results obtained from OMNeT++ ..... 52

Table 2-3: Results obtained from TOSSIM ..... 60

Table 2-4: Results obtained from IDEA1 ..... 63

Table 2-5: Comparation between selected simulators based on the modelling approach, energy-awareness and Heterogeneity ..... 68

Table 4-1: Electrical characteristics for the selected circuits (Appendix 2)..... 105

Table 4-2: Relation between transmission power level, current consumption level and transmission distance for CC2420..... 111

Table 4-3: Wireless Link Protocol Settings ..... 117

Table 4-4: General settings of the scenarios limited to the Telecommunication Unit. ..... 118

Table 4-5: Energy Consumption of the phases in different scenarios ($F_P$ = 1 Hz) ..... 119

Table 5-1: Wireless Comparation between our simulator and the selected simulators based on the modelling approach, energy-awareness and Heterogeneity ..... 133

# List of abbreviations

**A**

| | |
|---|---|
| ADC | Analogue-to-Digital Converter |
| AFL | Academic Free Licence |
| AODV | Adhoc On-demand Distance Vector |

**B**

| | |
|---|---|
| BLE | Bluetooth Low Energy |
| BSD | Berkeley Software Distribution |

**C**

| | |
|---|---|
| CC | Charge/Current mode |
| CCA | Channel Clear Assessment |
| CL | Circuit Level |
| CoAP | Constrained Application Protocol |
| COOJA | Contiki OS JAVA |
| CPU | Central Processing Unit |
| CSMA/CA | Carrier Sense Multiple Access with Collision Avoidance |

**D**

| | |
|---|---|
| DARPA | Defense Advanced Research Projects Agency |

**E**

| | |
|---|---|
| EIGRP | Enhanced Interior Gateway Routing Protocol |
| EP | Energy/Power mode |

**F**

| | |
|---|---|
| FPGA | Field-Programmable Gate Array |

**G**

| | |
|---|---|
| GMR | Geographic Multicast Routing |
| GPS | Global Positioning System |
| GUI | Graphical User Interface |

**I**

| | |
|---|---|
| ICMP | Internet Control Message Protocol |
| IDEA1 | hIerarchical DEsign PlAtform |
| IEEE | Institute of Electrical and Electronics Engineers |
| IoT | Internet of Things |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |
| IS-IS | Intermediate System to Intermediate System |

**L**

| | |
|---|---|
| LAR | Location Aided Routing |
| LLC | Logical Link Control |

**M**

| | |
|---|---|
| MAC | Medium Access Control |

**N**

| | |
|---|---|
| NAM | Network AniMator |
| NED | Network Descriptive |
| NL | Node Level |
| NSF | National Science Foundation |
| NS2 | Network Simulator 2 |
| | **O** |
| OLSR | Optimized Link State Routing |
| OMNeT++ | Objective Modular Network Testbed in C++ |
| OTcl | Object-oriented Tool command language |
| OS | Operating System |
| OSI | Open System Interconnection |
| | **P** |
| PCHMR | Power-Controlled Hybrid Multicast Routing |
| PDU | Protocol Data Unit |
| PHY | Physical |
| PIC | Programmable Integrated Circuit |
| | **Q** |
| QoS | Quality of Service |
| | **R** |
| RIP | Routing Information Protocol |
| RST | Reset |
| | **S** |
| SCK | Slave Clock |
| SDI | Slave Data Input |
| SDO | Slave Data Output |
| SHM | Structural Health Monitoring |
| SIFS | Short InterFrame Spacing |
| SIG | Special Interest Group |
| SPI | Serial Peripheral Interface |
| SOS | Sensor Observation Service |
| | **T** |
| TCP | Transmission Control Protocol |
| TL | Topology Level |
| TOSSIM | Tiny Operating System Simulator |
| | **U** |
| UDP | User Datagram Protocol |
| UL | Use case Level |
| | **W** |
| WPAN | Wireless Personal Area Network |
| WSN | Wireless Sensor Network |

# INTRODUCTION

## General context

In recent years, Wireless Sensor Networks (WSN) have gained attention as a major research and development topic in electronics and computing fields. WSN consist of a set of nodes, which embed sensors coupled with processing units and wireless communication devices. Thanks to battery-powered and wireless connectivity features of nodes, WSNs are both highly flexible and scalable. Depending on the application, these nodes can have different architecture and purposes allowing a large set of implementations to be realised. In general, nodes have to fit into low-power, low-computational and limited-size conditions. For example, when the objective is to monitor physical parameters in harsh environments, such as volcanoes, WSN nodes need to be as autonomous as possible because neither charging nor battery changing is possible. In this type of application, the node lifetime, and thus the energy, is vital. Moreover, in other applications, nodes could be randomly distributed to cover a large space, such as a battlefield or forest. In these cases, nodes need to be able to play different roles, for example, a sensing unit, relay or gateway and could also be mobile. Thus, in these applications, the WSN needs to be flexible in terms of topology. Considering these necessities, developing this kind of networks has become a complex task, especially in the early stages of design, where essential decisions are to be made.

Therefore, in the context of WSN design, simulation tools constitute a useful solution. These tools provide a way to create, simulate and compare different scenarios. Each scenario can include hundreds or thousands of nodes, and parameters can be reconfigured repeatedly, without the need of a real testbed. Thus, in the early design stages, WSN simulation tools can save time, cost, and effort, especially when difficult conditions need to be addressed such as harsh environments. In the last twenty years, tens of simulators have been developed covering different perspectives of WSN. Some simulators provide a framework to virtually test high-level propositions that impact the entire system, such as protocols and topologies. Others supply a platform to simulate hardware-oriented models like specific nodes or circuits. In terms of results, and to provide reliable and considerable outputs, simulations need to be as close as possible to the real system that is being modelled. Therefore, considering WSN context and depending on the objective of the simulation tools, the models may accurately represent either

artificial systems such as network protocols and node's circuits, or physical phenomena like humidity, temperature and also signal propagation.

## Motivation and problematic

Because energy is a major challenge in WSN, over the last few years, many research works are oriented to enhance energy generation, storage, and consumption in every abstraction level of the WSN: topology, node and circuit. Therefore, in order to make the best trade-offs in the early design stages, the energy aspect requires to be addressed both from the high and low abstraction levels of a WSN at the same time. Thus, to provide accurate results, simulators need to be capable of handle energy following this same trend.

We observed, in the literature, that each existing simulator is developed to answer specific objectives. Thus, WSN were modelled from different perspectives (topology, node and circuit). To answer the objective of the simulators, different modelling approaches were adopted: single-level, multilevel and cross-level. For example, NS2 is a single-level simulator dedicated to evaluate network protocols, while TOSSIM, which is a single-level simulator too, is oriented toward hardware emulation.

As stated in the literature, modelling energy in WSN is a complex task because several parameters from different levels of abstractions impact the energy consumption. From energetic point of view, some simulators, such as NS2 and TOSSIM, are considered non-energy-oriented simulators, whereas other simulators, such as IDEA1, are classified as energy-oriented. Nevertheless, the study of the energy-oriented simulators demonstrate that none has a modelling approach that allow addressing several levels of abstraction together. Thus, as suggested in IDEA1, a specific modelling approach is needed to model and simulate energy in WSN from different perspectives at the same time.

## Main contributions

Considering that previous requirement, based on a survey covering the existing modelling approaches used for WSN, the first contribution of this thesis is to demonstrate that the cross-level approach is the most suitable one to answer the modelling requirements and energy constraints of WSN. Furthermore, a methodology was proposed to evaluate simulators focusing on modelling approach and energy awareness. Using this methodology, several simulators were evaluated to highlight their capabilities and limitations. With the obtained knowledge, the main

research question of our work is formulated as follows: In WSN, how to model energy from a cross-level perspective?

To answer this question, the second contribution of this thesis is to define, design and implement an energy-aware cross-level model for WSN. First, we introduce concepts such as phases and patterns that are used respectively to model accurately circuits' and nodes' activities. Based on these concepts, a specific model structure is provided to allow the implementation of the cross-level modelling approach. Secondly, to cover the energy-aware aspects, the node consumption behaviour is modelled using the pattern concept that associates different current or power levels in the function of time. Moreover, in our proposal, the energy storage and voltage level that is resulted from the regulator configuration are integrated to estimate the node energy consumption.

The development, using MATLAB, of a cross-level energy-aware simulator based on the proposed model constitutes the third contribution of our work. Considering the same application scenarios, outputs obtained from our simulator are compared with results provided by another WSN simulator, such as NS2, or by real WSN nodes. This work is used:

- To validate our implementation proposal of cross-level modelling approach but also to highlight their capabilities and limitations;
- To demonstrate, using the proposed model, the relevance, in the early WSN design phases, of the adopted modelling approach and the energy awareness concept.

## Thesis structure

The structure of this thesis is organised as follows.

**Chapter 1** demonstrates, using a state of the art of Wireless Sensor Networks, focusing on their applications and architectures, the need of simulation tools to answer the main design challenges of this kind of network. A survey on modelling approaches using to describe WSN is secondly provided in this chapter.

**Chapter 2** first introduces a methodology to evaluate and compare a set of simulators selected from a statistical study of the literature. After that, the evaluation of the selected simulators is provided focusing on the modelling approach and the energy-awareness aspects.

**Chapter 3** defines the proposed WSN model that is both cross-level and energy-aware. Its architecture and behavioural description are also detailed in this chapter.

**Chapter 4** explains the implementation of the proposed model in a specific WSN simulator. This simulator will be used to validate our model and to evaluate its performances. Thus, the outputs obtained from this simulator are compared with results provided by NS2 and with some measurements obtained using experimental nodes.

A **General Conclusion** concludes this thesis and outlines the activities of future research.

## List of publications

The research in this thesis has contributed to the following publications:

- **Michel Bakni**, Octavian Curea, Guillaume Terrasson, Alvaro Llaria, and Jessye dos Santos, "A Cross-level model for power-aware Wireless Sensor Networks design", 13ème Colloque National du GDR SOC2, Montpellier, 2019
- **Michel Bakni**, Luis Manuel, Moreno Chacón, Yudith Cardinale, Guillaume Terrasson, and Octavian Curea, "Methodology to Evaluate WSN Simulators: Focusing on Energy Consumption Awareness", 6th International Conference on Computer Science, Engineering and Information Technology (CSEIT-2019), Zurich, pp. 331-351, 2019
- **Michel Bakni**, Guillaume Terrasson, Octavian Curea, Alvaro Llaria, and Jessye Santos, "Energy-aware Cross-level Model for Wireless Sensor Networks", SENSORCOMM2019, Nice, pp. 46-51, 2019
- **Michel Bakni**, Luis Manuel, Moreno Chacón, Yudith Cardinale, Guillaume Terrasson, and Octavian Curea, "WSN Simulators Evaluation: An Approach Focusing on Energy awareness", International Journal of Wireless & Mobile Networks (IJWMN), Selected papers from WiMoNe-2019/CSEIT-2019, vol. 11, no. 6, pp.1-20, 2020
- **Michel Bakni**, Guillaume Terrasson, Octavian Curea, Alvaro Llaria, Jessye dos Santos, "An Approach for Modelling Wireless Sensor Networks: Focusing on the Design Concept and Energy Awareness", International Journal on Advances in Networks and Services, IARIA, vol. 13, no. 1 & 2, pp. 33-44, 2020

# Chapter 1. WIRELESS SENSOR NETWORKS AND THEIR SIMULATION

1.1. Introduction ..... 6
1.2. WSN applications ..... 7
1.2.1. Tracking applications ..... 7
1.2.2. Monitoring application ..... 8
1.3. WSN description ..... 10
1.3.1. WSN architecture ..... 10
1.3.2. Node ..... 11
1.3.3. Network ..... 15
1.3.4. Main challenges in WSN design ..... 24
.1.4 Approach and energy-awareness in WSN simulation ..... 26
1.4.1. WSN modelling approaches ..... 26
1.4.2. Energy modelling in WSN simulators ..... 33
1.5. Conclusion ..... 34

## 1.1. Introduction

A Wireless Sensor Network (WSN) is a set of interconnected and distributed sensor nodes used to gather information about physical or environmental conditions [AKY02]. Due to ease of deployment nature and their ability to reduce the costs associated with their installation and maintenance [YIC08], WSNs have been employed in a wide range of application domains. This explains why they are suitable for harsh and inaccessible natural environments such as ocean depths [AKY05] or volcanoes [WER06]. Additionally, WSNs have applications both in the military domain, for target tracking [AHM10], as well as in the civil domain, for individual health monitoring [GAO06].

Nevertheless, WSN application deployment needs to handle several challenges such as energy-awareness, network scalability and limited processing capabilities. For example, considering that nodes are battery-powered, the WSN lifetime depends on limited energy resources, whereas WSN applications generally require running for several months or years without maintenance. WSN has also to face up scalability constraints taking into account that the number of nodes might vary in scale: tens, hundreds, or even thousands of nodes that could be positioned in a local or in a wide area [YIC08].

Therefore, in the designing and developing stage, there is a general need for tools to test the deployment of WSN applications. In this context, simulators are useful tools to compare different kinds of scenarios in terms of scalability, network topology and node architecture including both software and hardware. For example, simulators provide a way to virtually test thousands of nodes using different sets of parameters. In addition to that, they supply the user with the necessary information to study and analyse system performances such as network and node lifetime for given scenarios, and thus, enhance the design in an early stage prior to the real implementation.

In order to answer the developer's needs, simulators require reliable and accurate models that precisely describe WSN as well as their functionalities, such as data exchange, and services like routing [KOR09]. These models should provide mechanisms to represent interactions not only between nodes and their components, but also with physical environment, where the network will be deployed.

In order to define what is a WSN, and how the network is modelled, this chapter is organized as follows. The first section is an introduction to the Wireless Sensor Networks. It provides an overview of WSN applications and their architecture. The second section is dedicated to the

modelling of Wireless Sensor Networks .This section traces the development of the modelling techniques and WSN simulators to better understand the challenges facing WSN simulators today.

## 1.2. WSN applications

As mentioned before, WSN is a set of nodes that are wirelessly connected aiming at gathering information by sensing the surrounding. In this context, a WSN application is the use of the network to perform one or more sensing-based service. As proposed in [YIC08], WSN applications are classified into two categories based on the type of the service they provide. Figure 1-1 provides a classification for WSN applications, namely monitoring and tracking. In tracking applications, the main objective is to follow a mobile target. Thus, nodes are deployed in a way that they cover the largest possible area where the target will move [YIC08]. On the other hand, in monitoring applications, network nodes are distributed inside or around a specific environment to observe natural or artificial phenomena such as for volcano and production lines supervision.

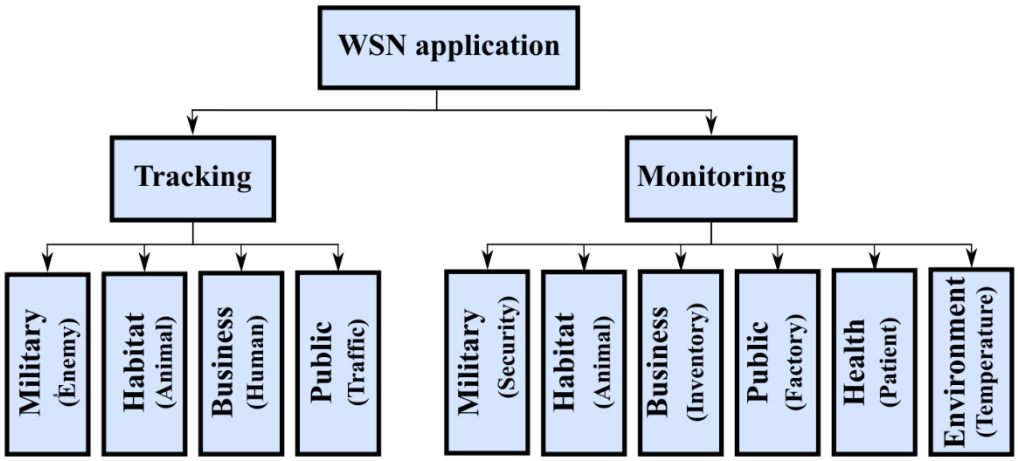



**Figure 1-1: Applications of Wireless Sensor Networks [YIC08]**

### 1.2.1. Tracking applications

In tracking applications, the main concern is to locate the position of a specific node, either according to a relative coordinate system or based on a global coordinate system such as GPS [KUS07a].

An example of public tracking applications is SVATS [SON08], an anti-car-theft system. SVATS is composed of a set of sensors that are being distributed within vehicles inside a parking. When the system is running, the nodes connect with each other and discover their

relative positions creating a pattern. After that, each nodes track changing relative positions. For example, if a vehicle is moved without authorization, the pattern will change causing a sound to alarm.

Tracking applications are also used to track living organisms such as a human being or an animal, domestic or wild. For example, the authors of [RAN14] present a global system for identifying and tracking humans worldwide. The suggested solution is based on the Global Positioning System (GPS) combined with a sensor network to recognize the presence of people tagged with RFID technology. WSNs are also used for tracking applications in the agriculture domain. For example, authors in [LLA15] suggest a system composed of GPS nodes to geolocate domestic animals in mountainous areas. Regarding animal tracking, another application is presented in [DOM16]. The authors deploy in Doñana national park in Spain a WSN to supervise animal behaviour. Each WSN node is composed of both GPS and several accelerometers that were installed on animals using collars. Then, considering row collected data from these nodes, a classification method was proposed aiming at detecting animal behaviour patterns, such as sleeping or coupling. Another example of application that is oriented to tracking animals is detailed in [AGU16]. In this article, a solution, based on a WSN, to track domestic pets in an indoor environment is described. The deployed solution follows the animal activities, collects data, and analyse it. Based on the aggregated data, the system proposes solutions to control animal movements, such as automatically closing some home sections in order to restrict pets' access.

Moreover, WSN tracking applications are also deployed in the military domain [AZZ17]. For example, the authors in [PRA15] describes a multi-target tracking system. To achieve that, instead of real-time surveillance, the system periodically locates the moving target. Then, collected data are used to predict the future movement of the supervised targets based on their previous movements. This strategy is designed and implemented considering the energy awareness of this kind of WSN application.

### 1.2.2. Monitoring application

Based on the previous classification, WSN monitoring applications could be classified according to their use into six categories: military, domestic, public, commercial, health, and environmental applications.

In the military domain, WSN technology have been identified to help command centres in their decision-making process by providing a flow of logistical and intelligence information that

covers an entire battlefield. Consequently, military monitoring applications have been a key driver for the development of Wireless Sensor Networks [DUR12]. For example, authors in [SIM04] propose a system for locating sniper using a Wireless Sensor Network. The idea of this application is to sense acoustic changes caused by sniper's shooting in order to determinate the shooting angles and distances. These acoustic changes are monitored using a set of sensors deployed that collects data to locate the sniper location. VigiNet [TIA06] is another example of a military application where the battlefield is the target to be surveyed. This application is based on a set of presence sensors that are synchronized together through wireless communication in order to collect and analyse information, such as the number of vehicles, their positions and their velocities, both from the battlefield or specific selected zones.

Due to their ability to be deployed in harsh environment and to cover vast areas, WSN are widely used for environmental monitoring applications. In [WER06], a 16-node Wireless Sensor Network is deployed to collect information about volcanic activity, the target is the volcano Tungurahua in Ecuador. Each node is equipped with a microphone and seismometer, collecting seismic and acoustic data from the volcanic area. Another example can be found in [TOL05]. WSN, in this example, is used to monitor ecological indicators, such as humidity and temperature, in the Redwood forest of California which spread over 500 km².

WSN monitoring applications also offer solutions to a range of medical challenges. The network ability to collect data can be used to gather patients' information including not only physiological data and biomarkers but also information to monitor their routine behaviour. For example, authors in [CHI10] propose a system to monitor health indicators of several patients in a clinic, such as cardiac parameters. In [CAT15], there is an example of an intelligent regulatory structure for health care monitoring based on network technologies such as RFID and 6LoWPAN. In this study, the WSN provided a framework for those technologies to function and interact with each other's to provide automatic monitoring of patients or medical staff, and biomedical devices within hospitals and nursing institutes. Moreover, WSN implicitly includes wireless connectivity. This is an advantage in the medical domain where cables and wires can draw strict limitations on the application [ZHA14]. For example, in [RAM 18], a WSN is deployed in an elderly house to provide information on people's activities and behaviours. Cables in an environment like this might block the movement and might be dangerous for the inhabitants.

In addition, monitoring applications could be applied in an industrial context. A platform, such as the one suggested in [NAC05], offers a general industrial monitoring system using Wireless

Sensor Networks. In this implementation, the nodes are featured with industrial vibration sensors. The network was deployed all around the factory to collect data in order to determine equipment health. Example of a WSN monitoring application in the industrial domain can also be found in [LLA16]. In this paper, the authors deployed a WSN to monitor and control an electrical grid. Nodes are located on the controller for both the energy sources and loads to supervise production and consumption of electricity at the same time. The WSN nodes are also composed of actuators that help to manage the grid by, for example, disconnecting an energy source. This kind of WSAN (Wireless Sensor and Actuator Network) application emphasise the concept of the smart grid. Another industrial monitoring application for WSN is Structural Health Monitoring (SHM). In this kind of applications, a WSN is deployed to collect data about the integrity of a specific structure [ARC17] such as bridges or buildings.

## 1.3. WSN description

This section explains what is WSN and how to address it. In the beginning, different architectures will be provided to better understand the main components of the network. Later, based on the abstraction levels, the main components will be explicated.

### 1.3.1. WSN architecture

As mentioned above, applications of WSN can vary in terms of objective. This has an impact on the topology of the network and structure of the nodes. However, based on functionalities, WSN nodes can be classified into three main categories [PAT13] [VAL11]:

- sensors, which collect information from the physical world,
- sinks that are the destination for all information collected by sensors,
- and relays, which are the nodes that collect information from the sensors and send them to the sink.

Based on that, WSN architecture can have different distribution, number, and capabilities of the nodes. However, a typical WSN has one or more sink, few relays, and most of the nodes are sensors [COU11].

Figure 1-2 shows a typical architecture of WSN where the sink node connects the whole network to the Internet. The sensor nodes are deployed inside the WSN domain. Their types were selected to reflect the two types of applications early mentioned in this chapter: tracking and monitoring.

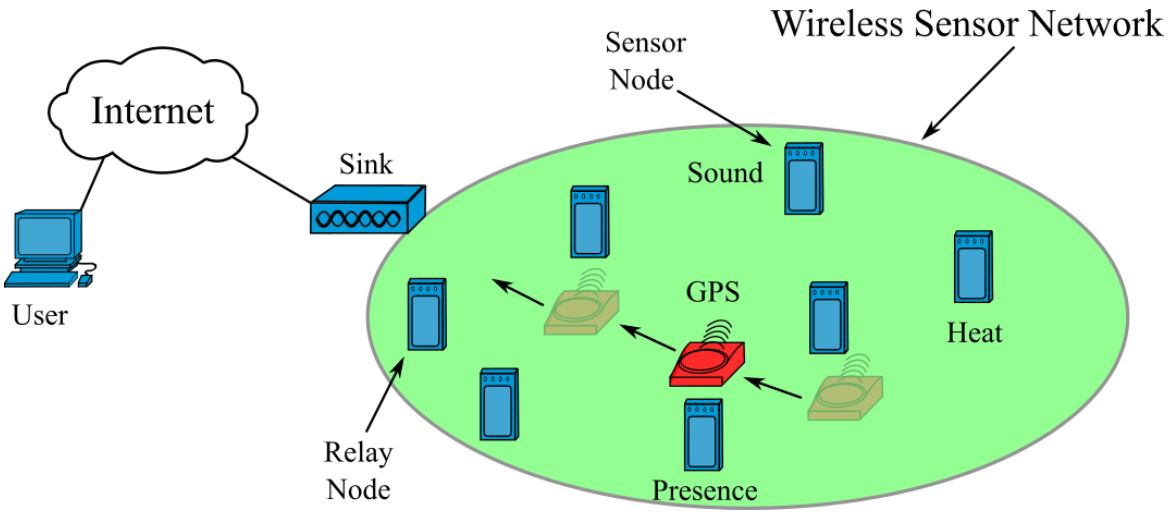



**Figure 1-2: Typical WSN architecture**

## 1.3.2. Node

In WSN, nodes the basic autonomous components. They are composed by hardware circuits and their associated software. In this section, several architectures of the nodes are to be presented as well as a brief overview of their software and hardware.

### 1.3.2.1. Node architecture

As explained above, the sensor node is the main component of a WSN. A general sensor node architecture is proposed in [NAV14a] and illustrated in Figure 1-3. Each node consists of hardware and software components. Hardware includes electronic circuits while software covers the operating system and network protocols.

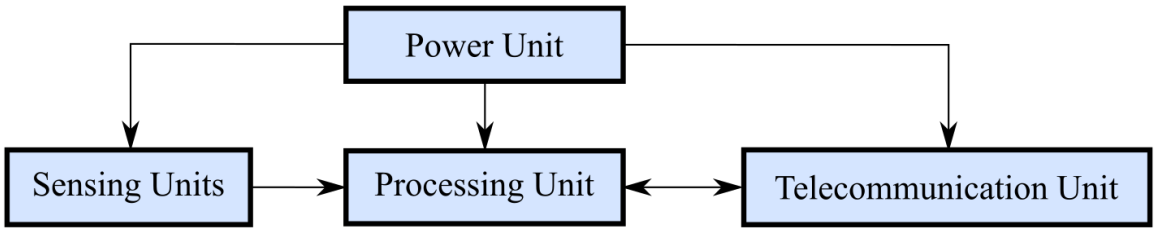



**Figure 1-3: Architecture of WSN as illustrated in [NAV14a]**

The node's hardware can be categorized according to the functionality as follows [SAK12] [HEA08] [CHE05]:

- Power Units. In WSN, nodes are generally supplied with limited power sources such as batteries. Depending on the application, the desired node lifetime could range from days to years [HEM08]. Because of that, energy issue is a main research topic in the WSN field [LIN17]. Thus, Power Units need to be carefully addressed, modelled and

implemented. All circuits that perform power-related functions are categorized under this item. This includes batteries, both rechargeable or non-rechargeable types, voltage converters and energy harvesters [CHE05].

- Processing Units. A Processing Unit constitutes the node's brain. It combines all the node components creating an autonomous stand-alone unit. It controls, generally using an Operating System (OS), all the tasks performed in the node such as data processing and time synchronization. This category includes the Central Processing Units (CPU) and all its peripherals such as memory and Analogue-to-Digital Converter (ADC), Programmable Integrated Circuit (PIC) and Field-Programmable Gate Array (FPGA) [VIE03]. Typical examples of Processing Unit circuits used in WSN are PIC18F4620 [PIC18], ATmega128L [MIC11] and MSP340 [TEX11].
- Telecommunication Units. This category covers all the circuits that serve the telecommunication onboard the node. Because WSN nodes are wirelessly connected, each node should be equipped with wireless circuits including RF transceivers and antennas. Different standards for wireless telecommunication might be used, this includes 802.11, 802.15, and Bluetooth standards. Moreover, some nodes can have wired interfaces such as Ethernet and USB providing a possibility to connect the node to a wired network. A variety of technologies such as WiFi and RFID can be supported as well [ABD13]. Examples of transceivers used in WSN are CC1000 [TEX09] and CC2420 [TEX19].
- Sensing Units. These units are used, depending on the WSN application, to measure one or more physical phenomena in the surrounding environment such as temperature or humidity. When the Processing Unit ask for a measure, the Sensing Unit captures a physical quantity and converts it into a signal which can be read, processed, and stored by the node Processing Unit. Sensors can be classified based on the sensor size, energy consumption, and operating modes [SOH07]. Examples of sensor circuits used in WSN are TMP102 [TEX07] and ISL29003 [INT11].

Hardware-oriented architectures can be derived from the basic one, shown in Figure 1-3. However, the WSN application pushes the development of the model to focus on specific hardware details. For example, authors in [AKY02] provide a detailed hardware-oriented architecture for the WSN node. The Processing Unit is further divided in processor and storage units. Other specific-purpose on-board circuits are added as well such as the location system and the power generator. Figure 1-4 illustrates this architecture.

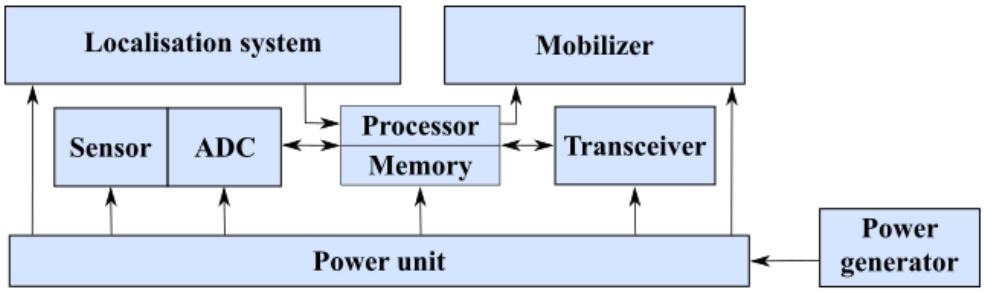



**Figure 1-4: WSN node architecture proposed in [AKY02]**

Another variation of the node architecture can be found in [BOU18]. In this architecture, presented in Figure 1-5, the links between the node components are specified into power and data link. This is an energy-oriented shift in the architecture design. It highlights the importance of energy-oriented modelling in WSN where node autonomy has a major priority.

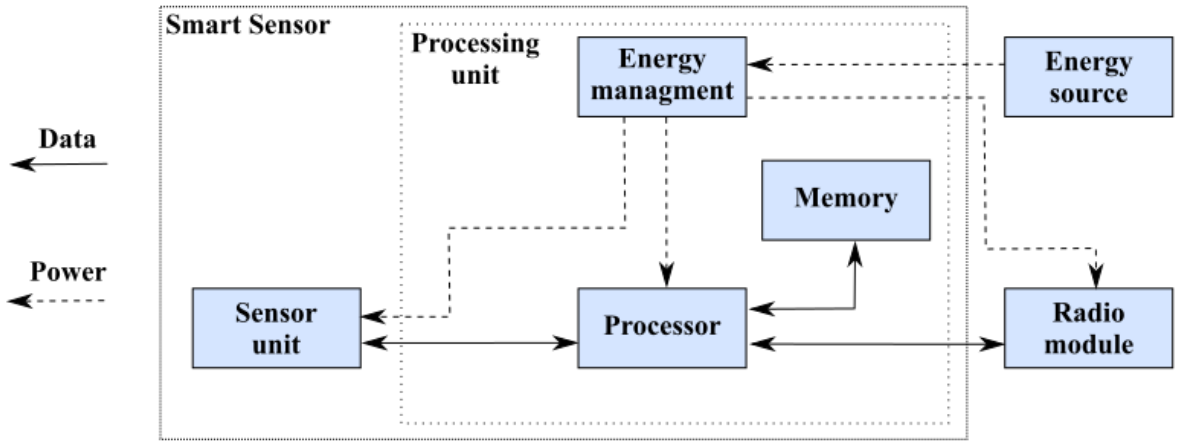



**Figure 1-5: WSN node architecture proposed in [BOU18]**

Based on the previous discussed architecture, several commercial WSN platforms, like MICAz [CRS14], TelosB [MOT04], and emote [MOT12] are available. [CHI12] and [JOH09] provide descriptions and comparisons between those commercial nodes as well as IRIS, SunSPOT and Cricket.

#### 1.3.2.2. Operating systems

Operating Systems (OS) provide the framework for the software, such as protocols and applications, to run. OSs used in WSN are an evolution of the traditional OSs. It extends the original model to support WSN-specific requirements such as high dynamics and energy constraints [ADI09]. The changes made to the traditional OS include aspects such as architecture, programming model, scheduling, memory management, network protocols support and resource sharing [FAR11].

WSN OSs can be categorized based on application orientation into general-purpose and specific-purpose OSs. The first category can run different types of applications, while the second is oriented toward a specific type such as applications that required parallel processing. In the following, two examples of general-purpose OS will be presented. The first example of general-purpose OS is TinyOS. This component-based Operating System can handle concurrent programs with very low memory requirements. It includes also support for network protocols, sensor drivers as well as a diversity of data acquisition tool [LEV05]. TinyOS supports application development using NesC. Platforms support TinyOS are MICA, MICA2, MICAz and Telos among others. Contiki is another example of general-purpose OS. This open-source Operating System was developed for low-power wireless devices. Contiki provides support of variety for network protocols including IPv4, IPv6 and 6LOWPAN for network layer and 802.15.4 for link layer [DUN04]. Contiki supports application development using C. Examples of platforms supporting Contiki are Tmote and MICAz.

On the other hand, MANTIS OS is an example of specific-purpose WSN OS. It is a parallel Operating System that suits parallel tasks such as signal processing and pre-emptive multithreading [BHA05]. Another example of specific-purpose WSN is the Sensor Observation Service (SOS). It is an Operating System that is oriented into data gathering from distributed sensors. It fits scalable applications where the number of the nodes have the major impact on the network performance [HAN05].

Additionally, the non-OS-based model is another option for implementing WSN node. In this approach, the system is simple and energy consumption can be significantly reduced thanks to the low-energy consumption technologies [MAG16]. However, when microcontrollers are used, the data computational capabilities become limited as well as the complexity of applications to be supported [VIE06].

Finally, back to the architecture presented in Figure 1-3, it is necessary to extend this architecture to include the previously mentioned software aspects. For example, authors in [BRI09] provide a software-oriented architecture for the WSN node which is illustrated in Figure 1-6. This architecture helps showing the role of OS in managing the hardware of a node. Moreover, in this architecture, the processing unit is also divided into sub-elements: the processor and the memory. This could be useful to consider software-related operations, such as loading the OS from memory when the node is initialized. In this architecture, the network module with a stack of layers is represented separately for the transmitter/receiver unit. In the next section, the network module will be analysed in detail.

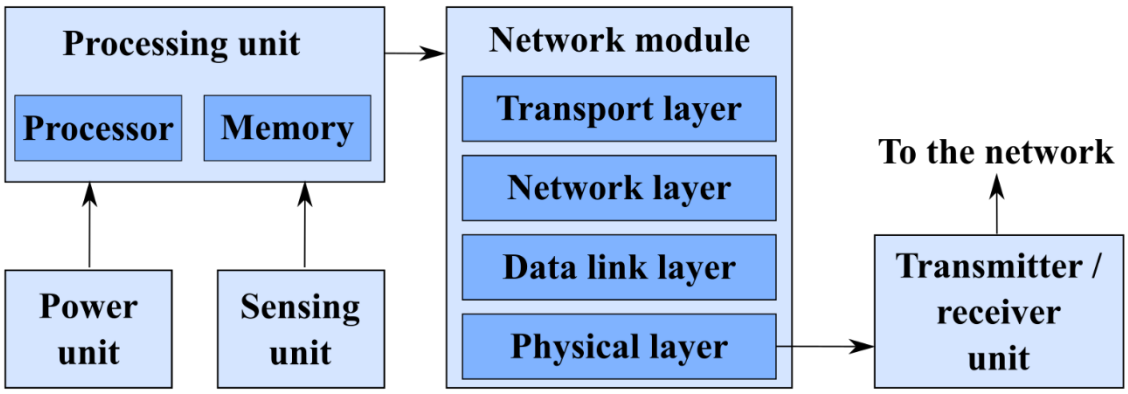



**Figure 1-6: Software-oriented architecture for WSN node**

## 1.3.3. Network

At the network level, the positioning of WSN nodes and their relationships they formed with each other are studied. Thus, in this section, both the WSN topologies and protocols are addressed.

### 1.3.3.1. Topology

Network topology could be defined as an arrangement of the network elements including both nodes and links, which can be physical cables or wireless channels. The topology covers two aspects: physical topology that includes the placement of nodes and components of a network in a coordinate system, and logical topology which covers the traffic within a network.

The design of network topology is one of the stages of WSN application development. Decisions related to topology can have impact on the following functions and services of the network: coverage, connectivity, lifetime, throughput and efficiency of energy consumption [GEN10].

Wireless Sensor Network topologies could be classified according to nodes distribution. Two main categories could be distinguished: predetermined topologies and randomly distributed topologies [AKY10]. In the predetermined topology, nodes locations are known prior to the deployment of the network. This type of topology could also be characterised based on the number of dimensions to a 2D and 3D topologies. Furthermore, each topology could be divided according to the maximum number of neighbours expected for a given nodes. For example, if the topology is described to include up to 4 neighbours, this means that each node can have at most four connections with adjacent nodes [SAL01]. After deployment, the nodes must then communicate with each other to identify and form the topology according to the predetermined

distribution. In randomly distributed topologies, the nodes can be deployed in a location-free structure manner, it is the case when nodes'positions are not predetermined [SHR07].

Both the predetermined and the randomly distributed topologies can be classified based on their structures to mesh, ring, tree or cluster topologies as illustrated in Figure 1-7. Several studies have attempted to explain the need to use cluster topology [VAL06] and how the multi-hop path affects network performance in that case [LEE09]. Mesh topologies are common forms in WSN topologies [SHA13]. In this kind of topology, all the peripheral nodes are connected, directly or indirectly, through other nodes, to a central hub. The central hub can be either the sink or a gateway node. A natural and logical extension of the mesh topology is the tree where sink node is the root and nodes at different levels in hierarchy are connected via direct links [SHR07]. The use of cluster topology instead of the mesh topology is a function of several factors, mainly the size of the network, i.e., the number of nodes, and the average distance between nodes [VAL11].

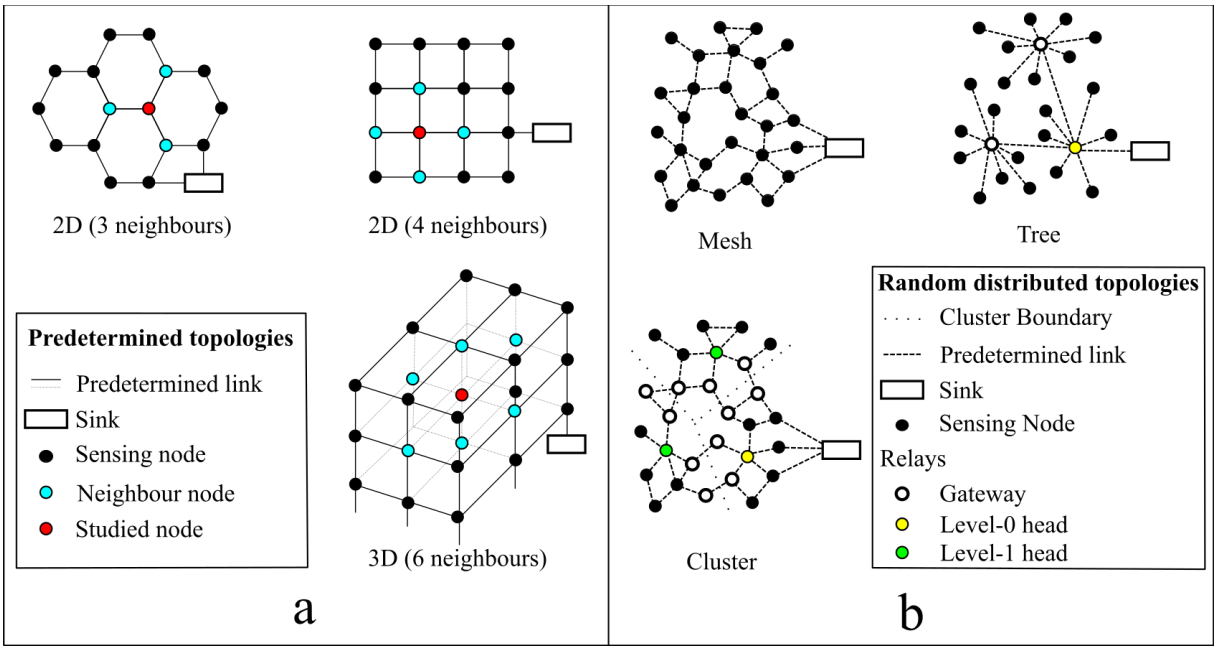



**Figure 1-7: Different types of topologies for WSNs**

In a WSN topology, as mentioned previously, nodes can be categorized according to their functionality: sensing, relay and sink nodes. The sensing node, also known as end-device node, is the basic building unit for topologies. Its task is to measure a physical phenomenon, and then, report the obtained value back towards a database to be collected and analysed.

The relay node is a special-purpose node that connects to sensing nodes within its coverage area, collects their data, and forwards it to the sink node. In some applications, relay nodes

could also embed, for example, functions to analyse the collected data and to put it in the desired format in order to significantly reduce the number of collisions in the wireless link and enhance the WSN performance [LIN17]. This kind of node could also play a major role in the creation of diverse forms of hierarchical topologies as illustrated in Figure 1-7. For example, a cluster head in a cluster topology is defined as a relay node. Thus, relays need to have an architecture that differs from the sensing node architecture, especially considering computing capabilities [VAL11].

The sink is also a distinctive node, which may be connected to a permanent power source, and have high processing capabilities. It could be associated with databases. Within the sensor network, the sink is the final destination for all data being collected [KHN13].

#### 1.3.3.2. Protocols

Network protocols defines a set of rules that governs the communication between all the network nodes. They establish, manage, close the connection channels and provide services, such as routing, security and connectivity.

Based on the functions performed, network protocols are generally categorized to fit in a layered-stack model that is used to represent a whole communication system. In this model, network functionalities are divided into a set of abstract entities called layers. The protocols that provided the functionalities of a specific layer will be considered resident of that layer. The layers are stacked together one above the other forming a stack of layers that represents a communication system. Within that communication system, each layer connects only to the adjacent layers by means of interfaces. In the same system, protocols reside at adjacent layers can form adjacent-layer interconnections. When two systems communicate, the protocols residing at the same layer exchange messages and construct same-layer interconnections [OSI94].

Figure 1-8 shows stack models that follows the previously discussed approach where (a) represents the Open System Interconnection (OSI) model [ZIM80], (b) shows the TCP/IP [SOC91], (c) displays Zigbee model [ZIG12], and (d) illustrates a five-layer stack model suggested in [AKY02].

In the following, a TCP/IP-based stack of layers model will be used. This model was selected because it was the starting point for the development of the WSN models [AKY02] [YIC08]. However, in our work, two modifications to the original TCP/IP model will be considered. First, the link layer will be extended to include the functionality of the physical layer proposed by the

other models. Second, an additional item that covers the routing protocols will be added. The reasons that justifies these modifications are explained in detail in the link layer and routing protocols subsections.

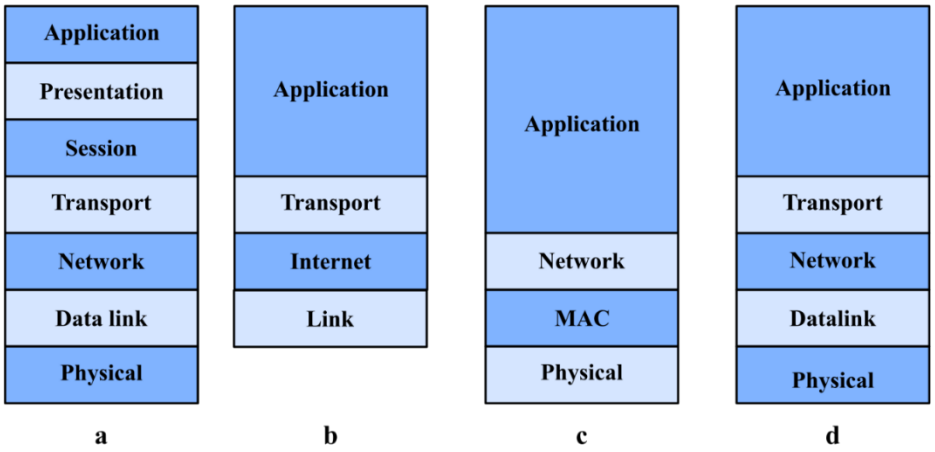



**Figure 1-8: Different layered-stack models for WSNs**

#### *1.3.3.2.1. Link layer*

The term "link-layer" is usually used ambiguously to refer to the functions and services provided by low-layers protocols: Logical Link Control (LLC), Medium Access Control (MAC) and Physical (PHY), such as the channel establishment, coding and modulation. This problem is due to a different definition in the standards. For example, on one hand, in OSI model, the functions of the low-layers protocols are divided upon two layers, the data link and physical [ZIM80]. On the other hand, the original Internet model describes a 4-layers model, providing only a link layer functions and ignoring the access to the physical medium [BRA89]. Moreover, later interpretations of the model add another layer usually referred to the network access layer [COM00]. Considering this, the model can describe the access to the medium.

The Institute of Electrical and Electronics Engineers (IEEE) adopts a stack model similar to the TCP/IP. However, it includes an additional layer at the bottom: the physical layer. IEEE manages both the developments and standardisation processes for most of low layers protocols used in WSN (Link and PHY). In the model proposed by IEEE, the link layer is divided into two sublayers: LLC and MAC. The first sublayer provides a way for different link protocols to connect with each other over the wireless medium without the need for routing, and the second sublayer provides the channel access control [IEE14].

In our work, as illustrated by Figure 1-9, the term "link-layer" includes all the low-layers functionalities, namely the time synchronization, physical addressing, power management and channel access.

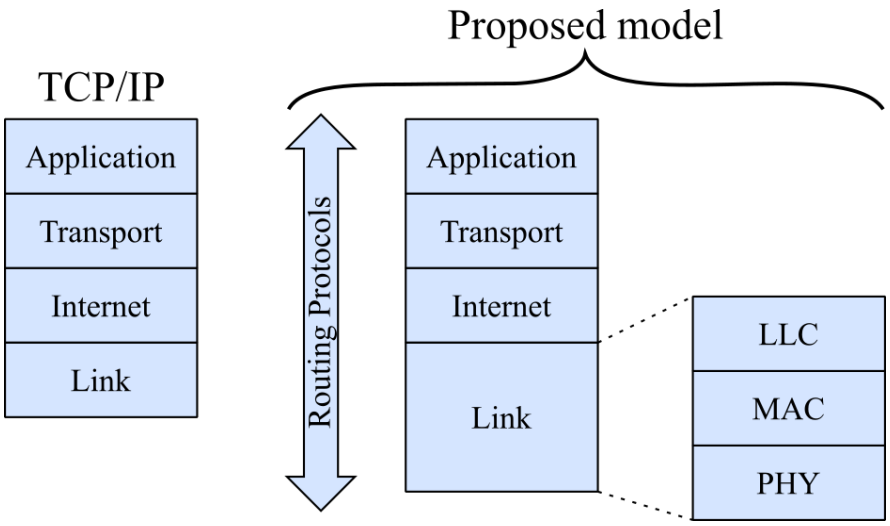



**Figure 1-9: Comparison between the TCP/IP model and the proposed model**

In the following, there is a brief summary of the most used Wireless link protocols in WSNs [AZM14]:

- 802.11 protocols family. The protocols included in this family were developed to define the one MAC sublayer and several PHY layers specifications for wireless connectivity for fixed, portable and moving stations within a local area [IEE16a]. The original standard was released in 1997 [IEE97]. Since then, dozens of modifications have been adopted under the commercial trademark WiFi to answer the market needs, including 802.11a [IEE98], 802.1b [IEE00], 802.11g [IEE03] and 802.11n [IEE09]. Some of these modifications were specifically developed to answer WSN needs (energy-awareness, network scalability and limited processing capabilities), such as 802.11ah which was developed to support large scale sensor networks [IEE17] and 802.11ba for energy awareness [IEE19].
- 802.15 protocols family. The 802.15 working group is dedicated for Wireless Personal Area Network (WPAN) aiming to define MAC sublayers and PHY layers specifications for wireless connectivity in devices with no battery or very limited battery consumption requirements. The workgroup developed several special purpose protocols including 802.15.3 for high rate WPAN [IEE16b], and 802.15.4 for the low rate WPAN [IEE16c].
- Bluetooth family. This standard was first developed as part of the 802.15 working group under the code name 802.15.1. The main goal was to create a wireless link protocol to exchange data in short distance for mobile devices [IEE05]. Since 2007, the standard is no longer a part of the 802.15 working group. Instead, the Bluetooth Special Interest Group (SIG) maintains it. Several updates of the standard were introduced in the last

few years to provide more adaptation for WSN. This includes an energy-oriented released adopted in 2010 called: Bluetooth Low Energy (BLE) [BLU10], and Bluetooth 5 that fully supports the Internet of Things (IoT) technologies [BLU16], which could be considered today as a major pillar in WSN evolution.

Table 1-1 provides a brief summarization for the characteristics of several link layer families and standards.

**Table 1-1: Main characteristics of several link-layer standards adapted to WSN**

| Release Date | Family | Standard | Frequency Band (GHz) | Bandwidth (MHz) | Maximum bit rate (bps) |
|---|---|---|---|---|---|
| **1999** | IEEE802.11 | 802.11a | 2.4 | 20 | 54 M |
| **2009** | | 802.11n | 2.4 & 5 | | 600 M |
| **2013** | | 802.11ac | 5 | | 1 G |
| **2003** | IEEE 802.15.4 | 802.15.4-2003 | 2.4 | 1 | 250 K |
| **2007** | Bluetooth | Bluetooth 2.1 + Enhanced Data Rate (EDR) | 2.4 | 1 | 3 M |
| **2010** | | Bluetooth Low Energy (BLE) | 2.4 | 2 | 1 M |

#### *1.3.3.2.2. Internet layer*

The internet layer is the second layer on the studied protocol stack where internetworking is being provided. This service is mainly focused on the idea of connecting the networks together creating a network of networks, and this is the base of the "INTerconnected NETworks " popularly called the Internet today. Protocols that provide this service are called the internetwork protocols [POS80a]. Internetwork protocols provide logical addressing to the network nodes. In addition to that, data fragmentation and the creation of the data packet is performed by these protocols at the network layer.

In the following, two internetwork protocols used in WSN are briefly described:

- Internet Protocol version 4 (IPv4) is the first stable version of the internet protocol. It was released in 1981 and is still in use until today [POS81a]. IPv4 has a main drawback, it is the scalability. The protocol uses 32-bit addresses that was not enough to answer the enormous expansion of the internet in the 90s [FUL06]. Thus, IPv6 was developed.
- Internet Protocol version 6 (IPv6) was developed in 1995 as a long-term response for the IPv4 address exhaustion [DEE17]. IPv6 logical addresses are 128 bits (16 bytes) long. The protocol provided a pack of new services and features such the auto configuration addressing [THO07]. IPv6 scales very well serving the increasing number of smart and connected devices such as WSN nodes used in the IoT applications [SAV13] [ZIE13].

Nevertheless, the use of the IPv6 features in WSNs was a complex issue. Each IPv6 address is 16 byte long. Additionally, a typical IPv6 header contains the source and the destination addresses. Totally, it means that 32 bytes for the source and destination addresses must be reserved, not to mention the other header fields. In many WSN applications, this can be longer than the size of the data. Therefore, comparing to IPv4, the use of IPv6 induces larger memory to store [KAW10] and a probably higher consumption for transmitting or receiving data whereas WSN are known for their low-infrastructure nature: limited memory, processing and computing resources as well as the requirements for low-energy consumption [YIC08]. Thus, many WSN applications are based on the use of IPv4 where the whole header length is 20 bytes. To answer that, 6LoWPAN was developed. It is a network-layer interface that allows the link layer of the WSN node to host an IPv6 address and to benefit from its features [KUS07b]. When 6LoWPAN is used, the addresses are abbreviated, and a short version of the header is created. A description of a specific IPv6 interface for 802.15.4 network-based can be found in [MON07] and for Bluetooth network-based in [NIE15].

#### *1.3.3.2.3. Transport layer*

The transport layer is the third layer in the considered protocol stack. The protocols residing in this layer provides end-to-end connectivity for the application creating virtual channel between the source and the destination. Additionally, mechanisms, such as error checking and recovery, to ensure channel reliability can be provided in this layer.

Because of the low-infrastructure and distributed nature of the nodes in WSN, transport layer was ignored in some of the WSN-oriented protocol stacks. For example, ZigBee, a well-known

standard for WSN node, was built over a network model, as illustrated by Figure 1-8, that does not include the transport layer [ZIG12]. However, the need to implement IoT applications and the development of 6LoWPAN both pushed towards providing the nodes with abilities to support transport protocol layers [KUS07b].

There are two protocols to introduce in this layer:

- Transmission Control Protocol (TCP) is one of the first protocols developed in the history of networks. Nowadays, it is used to establish an end-to-end channel, a very basic but vital function in networks. This protocol support mechanisms for flow control as well, adding reliability to the already established connection [POS81b]. However, these services and functions can exceed WSN limited capabilities in terms of memory, processing and energy resources [KUO06]. Considering these limitations, many solutions to integrate TCP in WSN are proposed. For example, in [GAS11], the authors provide the WSN with TCP services using an approach called Proxy TCP. In this method, a TCP proxy resides in the WSN and redirect all TCP messages into a remote TCP agent that resides out of the WSN.
- User Datagram Protocol (UDP) is a transport layer protocol that provides a light end-to-end connectivity with a minimum support for transport layer mechanisms [POS80b]. In some cases, applications have strict requirements for the Quality of Service (QoS). For example, WSN applications that include real-time video streaming do not need an error recovery mechanism, because the information has a value in the real-time only. Others might be sensitive to delay such as alarm applications [PAL16], or do not demand connection reliability such as environmental monitoring applications where several sensor nodes cover the same area [DAW12]. For these WSN applications, unlike TCP, UDP provides a connectionless end-to-end channel establishment, basic management services, and termination [SHA09].

#### *1.3.3.2.4. Application layer*

The application layer operates at the top of the protocol stack in the studied network model. It includes interfaces that allow application parts, such as client and server, to communicate through the network. The protocols that manage these interfaces are called the application layer protocols.

A service-based categorization of the application-layer protocols is provided in [AKY10]. According to this classification, application layer protocols can provide one of the following

services: source encoding, query processing, and network management. The source encoding service includes compression algorithms and operations. Examples of those protocols adapted to WSN can be found in [SAD06]. The query processing services is mainly concerned about the managing and aggregating of collected information from distributed nodes. WSN protocols that provides such services can be found in [MAD02] and [EHS11]. Finally, the network management services involve monitoring and managing network components. WSN protocols classified in this category are [RUI03] and [TOL05].

Main WSN characteristics in terms of energy and processing power could directly affect the implemented application protocol services. Thus, some WSN-oriented application protocol has been proposed such as Constrained Application Protocol (CoAP) [SHE14]. Standards bodies manufacturers also provide application protocol packages that fit into WSN specifications. ZigBee cluster library [ZIG12] and IEEE1451 [IEE07a] are examples of that.

#### *1.3.3.2.5. Routing protocols*

Routing is discovering the best available route for a data packet and using that path to route the packet. It is an Internet layer service. In this way, the role of routing protocols is to collect routing information from the nodes and calculate the best available route based on it. In this context, the word "best" is relative. It depends on the implemented routing protocol and the mechanism used to handle the topology. For example, some protocols determine routes based on distance, others use link parameters such as bit rate and reliability. In WSN, parameters related to energy or node's structure could also be included as a criterion in the calculation as well [HAO12].

Although routing is a function of the internet layer, routing protocols are not limited to this layer. To further discuss this point, a set of routing protocols is presented and categorised based on the layer that they function within: Routing Information Protocol (RIP) is an application layer protocol [HED88], Enhanced Interior Gateway Routing Protocol (EIGRP) is an internet layer protocol [SAV16], and Intermediate-System to Intermediate-System (IS-IS) is a link layer protocol [ORA90]. Thus, routing protocols cannot be categorised according to the network model, but rather as an independent item.

Based on the application, WSN can have topologies that are highly dynamic. Thus, routing protocols play a major role in this kind of network. In WSNs, routing protocols could be categorized according to the type of traffic they route into:

- Unicast routing protocols [AKK05] [HAO12] such as, for WSN, Optimized Link State Routing Protocol (OLSR) [JAC01] and Ad hoc On-Demand Distance Vector (AODV) [PER03]. On one hand, OLSR uses a link-state algorithm to determine all the available routes between two nodes. In this routing approach, proactively, a map that covers the whole topology is first created by a selection of nodes. Then, this map is diffused to all the nodes of the network to update their routing tables. On the other hand, AODV depends on a distance-vector algorithm to find the best route between two nodes. In this routing approach, the desired route is only calculated when it is needed. As stated in the protocol standard, based on the number of hops, every node in the topology makes its route calculation and routes unicast data packet according to that.
- Multicast routing protocols [JUN09] [FAR19]. Geographic Multicast Routing (GMR) [SAN07] and MUSTER [MOT11] are two examples of this kind of protocol used for WSN. GMR is computing the best multicast distribution tree by including geographical information, such as GPS coordinates, in the calculation. In another way, MUSTER performs energy-aware routing, which means energy-related information is included in the calculation of the best routes.

Many other routing protocols, who fit under one of the previously mentioned categories, were developed to include one or more WSN parameters in their route calculations. For example, Location Aided Routing (LAR) [YOU00] is a unicast routing protocol that considers nodes' position for the route calculation, whereas Power-Controlled Hybrid Multicast Routing (PCHMR) protocol [CHE06] takes into account energy parameters in their routing decisions.

### 1.3.4. Main challenges in WSN design

As previously discussed in this chapter, WSN applications are developed to answer requirements in several domains such as military, health, agriculture. The variation of domains where WSN can be applied lead to diversity in application purposes and in WSN design: network topology, protocols and node architecture.

In all WSN applications, nodes are used to collect information from a specific target, such as physical, physiological, location or movement data. The monitored targets could be natural, artificial or even human beings. Moreover, as demonstrated, WSN applications can be

categorised based on the mobility of these targets into tracking and monitoring applications. In the first category, the application objective is to generally collect data related to location and movement of mobile targets in the movement area. In the second category, the main goal is to capture physical or physiological information from a target that could be mobile or stationary. Therefore, applications have an impact on the number, the distribution, and the type of node deployed in WSN.

Table 1-2 provides a general but non-exclusive comparison between the two WSN applications' characteristics including differences in targets and data types as well as challenges facing each of which.

**Table 1-2: comparison between WSN applications main categories**

| | **Tracking application** | **Monitoring application** |
|---|---|---|
| **Target mobility** | Mobile | Mobile or stationary |
| **Target type** | Animals, vehicles or human beings | Production lines, elderly houses or volcanoes |
| **Data type** | Coordinates, speed ... etc. | Physical and physiological parameters: temperature, humidity, heart rate… |
| **Challenges** | Routing and mobilisation | Inaccessible, harsh environments, and autonomy |

Based on that, designing and implementing a WSN application have to face different challenges. Nowadays, to handle these challenges, generally, the design generally focuses on one of WSN aspects such as software (proposing new routing protocols) or hardware (develop specific circuits).

Nevertheless, addressing challenges like the energy consumption in WSN require to consider several trade-offs between specifications belong to different aspects: network topology, protocols and node architecture. For example, in some tracking applications, it is necessary to include position-related parameters in routing algorithms calculation in order to consider the mobilisation requirements. This could have an influence on the calculation’s complexity and, therefore, on the time and energy needed to perform them. On the other hand, several monitoring applications mainly focus on collect data in harsh environments as long as possible. Thus, the network lifetime has a significant priority whereas it needs also to be considered regarding the QoS required in this type of application.

Therefore, to make decisions related to these previous challenges, as early as possible in the design phase, developing WSN applications and new software or hardware solutions need tools

to evaluate them considering the application goals, different scenarios and their potential environments of deployment. In this way, simulation tools can provide a solution economising as much as possible time, cost and efforts. Indeed, these tools allow to simulate hundreds or thousands of nodes in their environment, without the need to physically implement or deploy instruments and real testbeds. Additionally, another interest of these tools is their capacity to repeat several times the simulation of different scenarios in order to analyse the obtained results and enhance the overall WSN performance.

## 1.4. Approach and energy-awareness in WSN simulation

As concluded previously, simulation is an important stage when designing and developing a WSN [MER09]. In this way, to be useful for designers, WSN simulators need to provide accurate and reliable results. A simulator is generally based on models to describe functionalities of the system to be simulated. In WSN simulator, the models are used to obtain the best possible time representation of the system behaviour, in terms of data exchange, energy consumption and other WSN properties. Therefore, WSN models and the approach, that is used to build them, plays a major role in WSN simulator accuracy and reliability [BAN10].

In this section, an overview on WSN modelling and simulation tools evolution is made with a specific focus on energy modelling. The goal of this work is to bring out main issues regarding WSN simulation in a context of energy-aware design.

### 1.4.1. WSN modelling approaches

Firstly, an overview of WSN modelling evolution is provided, aiming at defining the approaches on which WSN models are built. Based on these definitions, the progress in WSN simulation is traced to demonstrate that they follow the same approaches.

#### 1.4.1.1. Evolution of WSN modelling

In the last two decades, several modelling approaches were developed to answer the requirement of new technologies such as WSN [CHE17]. Consequently, many WSN models were built on approach. WSN modelling started from the stack approach adopted by the TCP/IP model [YIC08]. Following this approach, each communication system is modelled using a stack which, in turns, includes a set of layers that are separated according to their functionality. Each layer is designed and operates independently, with interfaces between the layers [SOC91].

However, as previously explained, while network tasks are performed, a problem within the stack approach appears when some parameters, in WSN, which belong to different and not

subsequent layers need to interact with each other. In models based on this approach, non-adjacent layers have no direct interfaces because the layers must respect the stack hierarchy. For example, modelling energy consumption in WSN requires interaction between parameters such as the duty cycle which is set on the application layer, and the hardware power levels which are configured on the link layer. Modelling of the routing service will lead to the same problem as well. As mentioned early in this chapter, routing occurs on the network level. However, routing information might be collected using protocols working on other, adjacent or non-adjacent, layers.

To address these modelling issues, changes to the traditional stack-layered model were needed. For example, in [GOL02], the authors propose a cross-layer modelling approach for ad-hoc network. In this approach, information can be exchanged across the model (Figure 1-10-b), and not only between adjacent layers regardless of the stack hierarchy (Figure 1-10-a). This is called a cross-layer interaction. It allows the protocols to globally adapt to both the application requirements at the top of the stack and the underlying network conditions at the bottom such as medium access. The idea was to keep the old structure, but to employ it in a different way. However, all the layers belong to the same modelling entity, which will be called, starting from now: a level. Thus, the two discussed models, stack or cross-layer, are included in one level only, which is the network level in these cases. That is why they are considered to follow the single-level modelling approach.

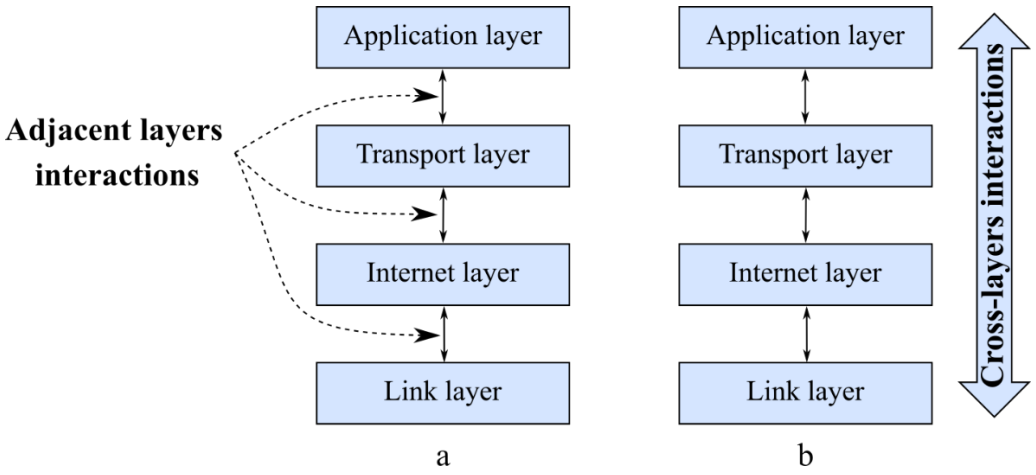



**Figure 1-10: Different interaction approaches between network stack layers (a) Traditional stack of layers TCP/IP model, (b) a cross-layer model for ad-hoc networks**

In [AKY02], the same single-level approach was adopted, and a cross-layered model was proposed based on it. This cross-layer concept is called a “management plane”. In this proposal, the authors suggested that tasks, such as mobility and power management, are better to be handled as a function in all the layers of the network model. As illustrated by Figure 1-11, these

tasks are placed in a second dimension that is orthogonal to the original model. This means that the tasks can now span across all the layers in the model. In [CHA09], this model was extended to include the security considerations from the cross-layer point of view.

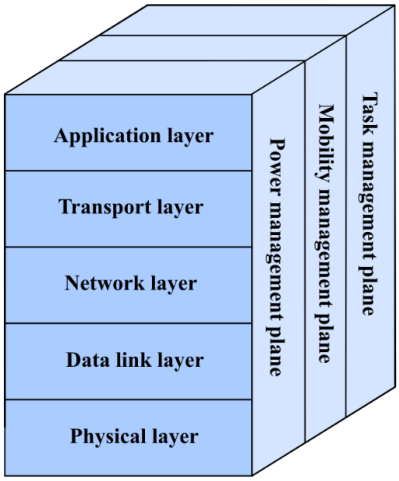



**Figure 1-11: WSN modelling approach proposed in [AKY02]**

Because WSN is a more complex system, it was necessary to extend the previous single-level model, and to include other entities, such as the communication channels and physical environment. Thus, the multilevel modelling approach emerged.

An attempt to introduce the multilevel modelling approach was proposed in [KUO05]. In this proposal, the layers of the stack were further divided into two main groups or levels, as illustrated in Figure 1-12. The first group contains non-physical parameters, mainly those related to the software and the application, and the second group is dedicated to the hardware where the protocols are implemented, as well sensors and RF unit parameters.

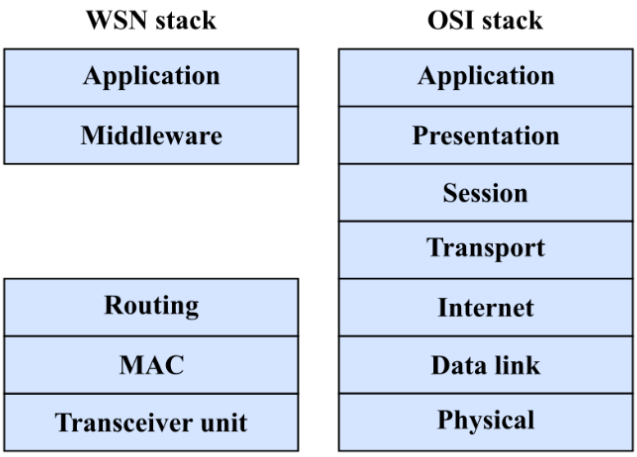



**Figure 1-12: Emerging to multilevel modelling approach, the WSN stack is two-level stack proposed in [KUO05]**

Another WSN model based on the multilevel modelling approach is proposed in [EGA05]. In this paper, the authors suggested the concept of "tier". As presented in Figure 1-13, a "tier" represents a group of parameters that can be assembled into what we consider after as a level. For example, parameters related to protocol stacks are grouped inside the protocol tier, while parameters related to the communication channel, such as radio channel characteristics are grouped to form the media tier. As a result, the WSN model was extended to include three hierarchical models instead of one, namely the system, the node and the media models.

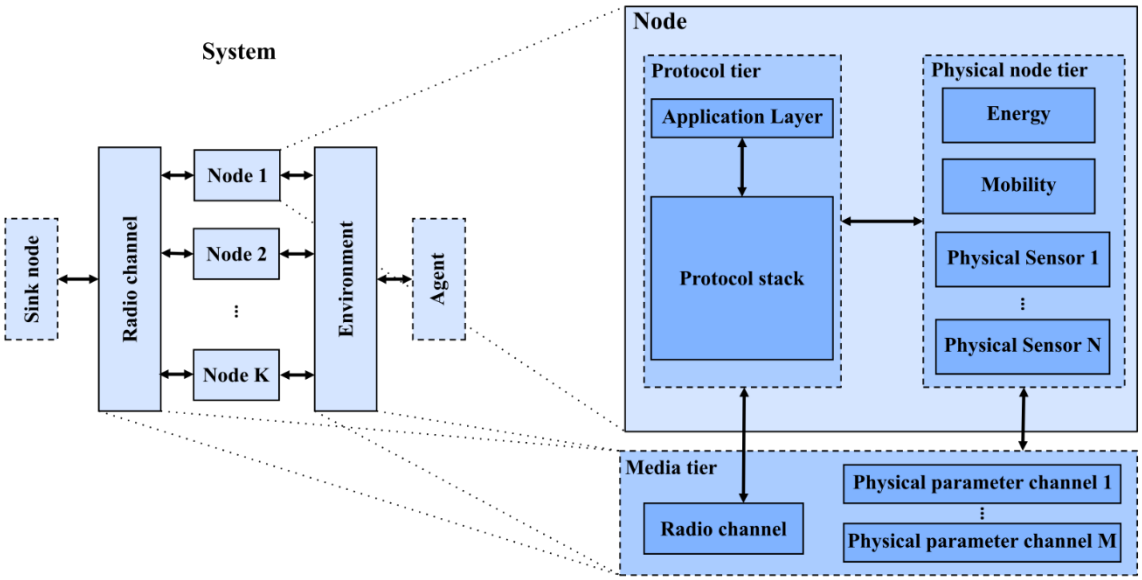



**Figure 1-13: The "tier" concept**

This model was widely used and improved. For example, the wireless medium was modelled and attached to the multilevel model [KOR09] [WAN10a]. This allows the wireless medium parameters, such as channels properties and distances between the nodes, to be included in the system model, as suggested by Figure 1-14.

The problem with the previous modelling approach is that entities from different levels cannot cross the boundary of the level and directly interact with each other, instead they have to follow the level hierarchy. To answer that, a new cross-level modelling approach emerged. Authors in [IMR10] demonstrate the importance of a vertical testing approach for WSN design. This paper introduces the use of the multiple-level modelling approach from a cross-level point of view. In the same way, examples of models that allow entities and parameters to interact regardless of the level boundary can be found in [HEF12] and [BER12]. However, the previous models do not fully adopt cross-level design approach. for example, in [BER12], the authors followed

the multilevel approach providing models for system and nodes. Although the two models are separated, parameters influence, is not limited to the level boundaries.

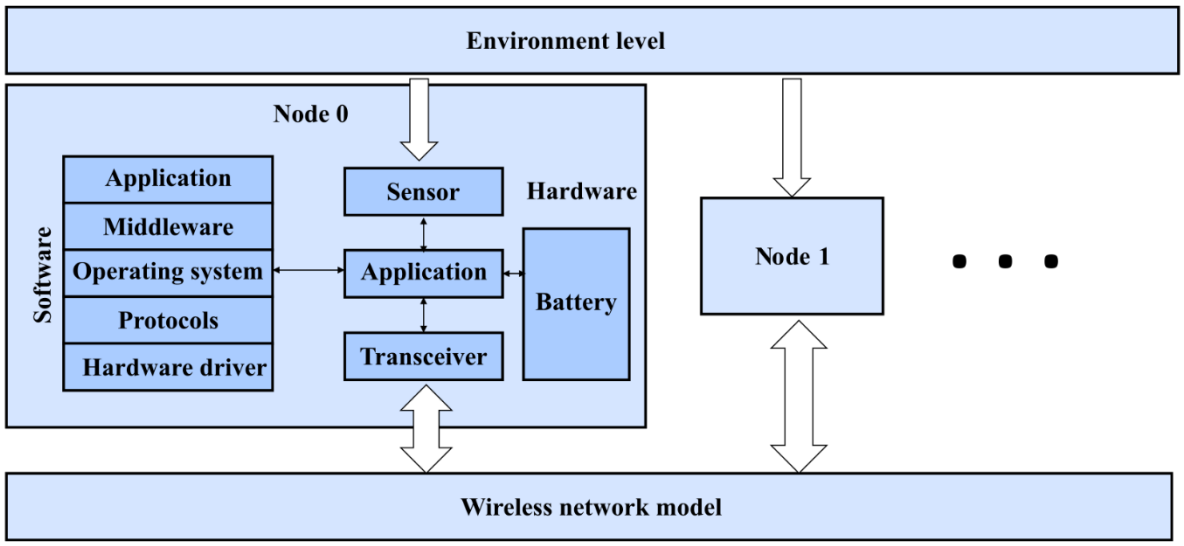



**Figure 1-14: The 3-level model for WSN proposed in [WAN10a]**

Based on the discussion above, three distinguished modelling approaches were developed. They can be briefly described as follows.

- Single-level modelling approach: all the entities reside in a single level, the entities can be layers, subsystems, or parameters, the definitions of each type is up to the model to be developed on the approach. The entities can interact with following the architecture of the model such as the stack-layered model or across the model.
- Multilevel modelling approach: the model can include different levels, each includes entities. Inside each level, entities can interact with each other directly. However, the relationship between the level is limited in ways that each level can interact with other adjacent levels only through interfaces. Entities belong two different levels can indirectly interact through the interfaces between the levels.
- Cross-level modelling approach: models build on this approach has the same structure as the multilevel models. However, entities belong to different levels and interact with each other across the model and they are not limited to the level's interfaces.

Figure 1-15 illustrates the models built on the previously mentioned modelling approaches. WSN simulators, which will be described in the next section, adopt those models and can be categorized in the same way.

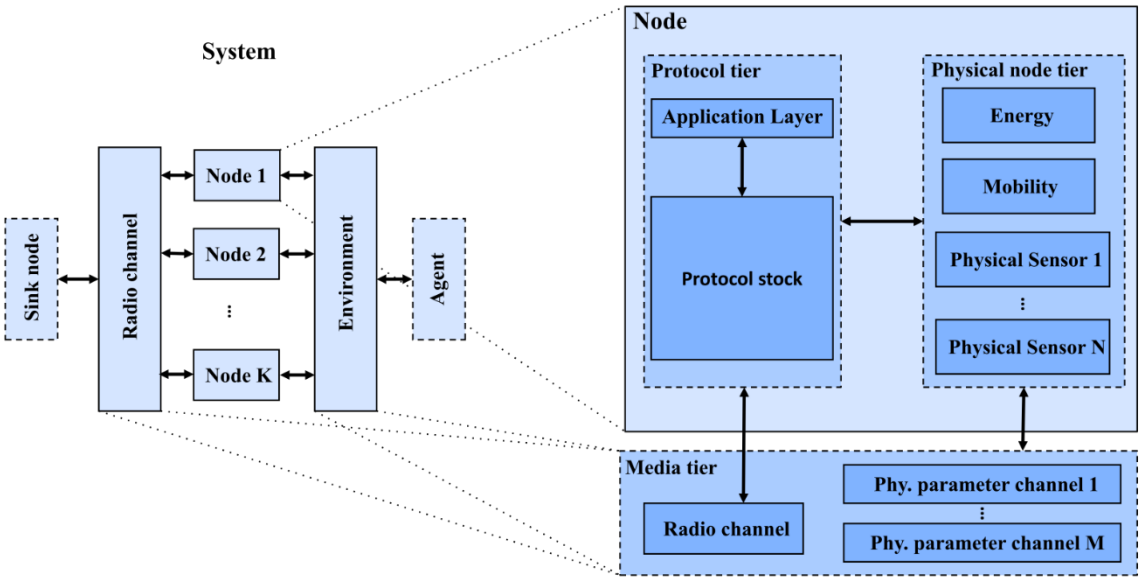



**Figure 1-15: Classification of WSN models based on the modelling approach**

### 1.4.1.2. WSN simulators: an overview

The imitation of an operation for a real-world process or system over time, with a particular level of accuracy, is called a simulation. The framework that hosts the simulations is called the simulator [BAN01] .In this section, we trace the development of WSN simulators in order to demonstrate that the simulation of WSN follow, in their approach, the same evolution as the WSN models.

When WSNs were emerged, efforts were first made to extend the computer networks simulators to involve this new kind of networks. Network Simulator 2 (NS2) [ISS09] is an example of simulator, primarily developed to simulate communication protocols for computer networks, which, from 2000, was expanded to include ad-hoc and Wireless Sensor Networks [YUA06]. This kind of simulation tools allows to focus on a major research topic related to WSN: routing and clustering in networks with no fixed topologies [BRI05] [CHI05] [GIO05] [LIO02].

While simulation tools, as NS2, are mainly oriented towards application and protocols, other single-level models dedicated to WSN are developed, mainly focusing on hardware. The main objective of these models was to provide hardware-oriented simulation, also called emulation. An emulation is generally used to show how a particular OS interacts with node hardware [LIU02] [GIO05]. An example of a well-known single-level emulator was developed under the name TOSSIM [LEV03].

Thereafter, as previously stated, the modelling of WSN tries to consider, at the same time, not only network and node levels but also environmental one. The simulation of this level is

important to study, for example, the effects of physical obstacles on the signal propagation. The environment level can also be used to create more realistic scenarios by generating values such as sound, light and temperature to be measured by each WSN node. Therefore, the development of simulators follows this multilevel approach. As a result, new simulators were created providing testing tools for the designers and researchers. These simulation tools permit to analyse collected results and make decisions considering different aspects of the WSN simultaneously. For example, SensorSim was an early attempt to build a multilevel simulator. It was built on models, wireless and sensor channels, that represent the environment. These models interact with other models within the node level to simulate the impact of surroundings on the communication and sensing parts of the node [PRA00]. Jsim was developed following the same modelling approach [SOB06]. In this simulator, as illustrated by Figure 1-16, WSN models, environmental, node and power, are separated resulting in multilevel architecture where each level can include different layers.

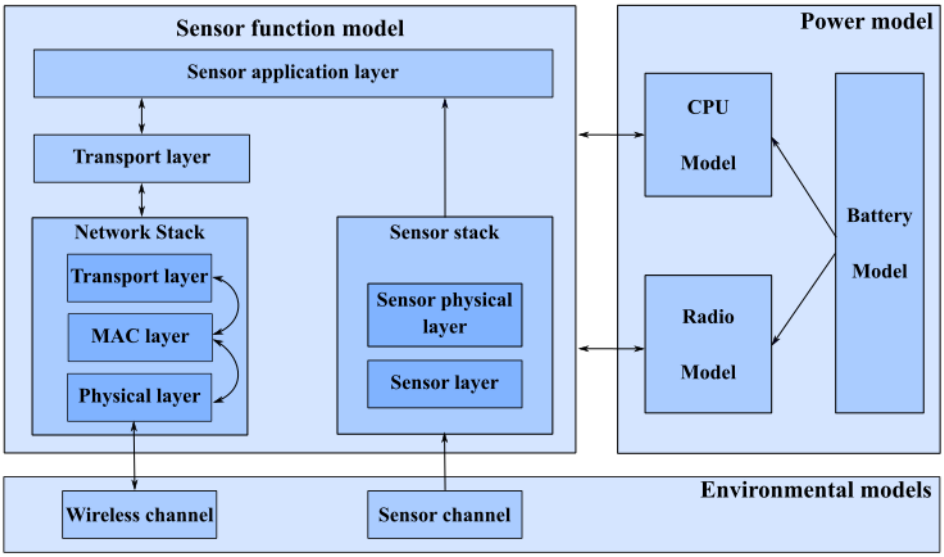



**Figure 1-16: Multilevel architecture adopted by Jsim, where environment parameters can affect the node through the wireless and sensor channels**

The multiple-level approach for WSNs modelling is widely used in WSN simulators and shows advantages over the traditional single-level approach. The major enhancement was allowing entities or parameters from different levels to interact with each other [WIT06]. However, as previously mentioned, the interactions were limited only to the interfaces between the levels. These limitations explain the development of cross-level models and simulators.

COOJA (Contiki OS Java) is an example of simulator from the cross-level category [OST06]. It was proposed in 2006 as the first simulator to adopt a cross-level model. COOJA includes

three levels: the network level, the Operating System level, and the machine code level. The first level involves the network topology and the environment configuration such as the radio parameters. The second one represents the node's software, and thus, the protocols stack, while the third is dedicated to the hardware. The simulator can handle events, such as the incoming radio traffic or a LED being lit, on different levels at the same time [OST06].

Considering the examples previously explained in this section, as WSN models [PHA07], simulators can be classified based on the modelling approach adopted by the developers into three categories: single-level, multilevel and cross-level.

### 1.4.2. Energy modelling in WSN simulators

As demonstrated early in this chapter, energy is a major research topic in WSN. Therefore, many WSN models tried to address the energy consumption in WSN in order to estimate node or system lifetime.

Considering this, authors in [CHE17] provide new classification criteria for simulators: energy-oriented and non-energy-oriented. Based on this work, an energy-aware level-based classification for modelling approach, illustrated in Figure 1-17, can be proposed.

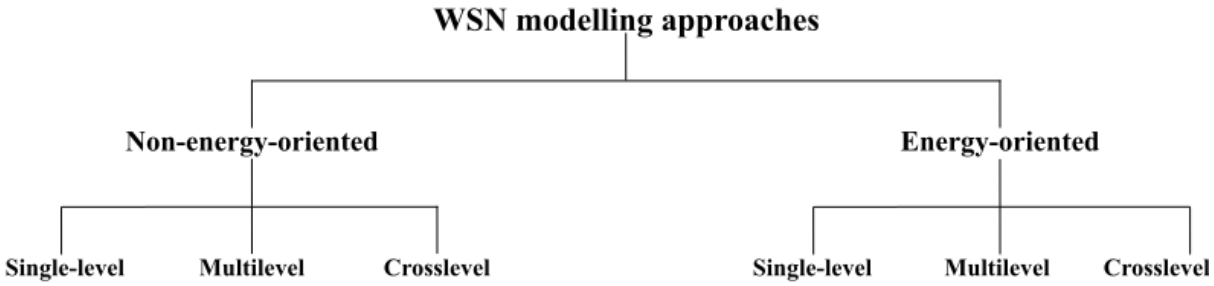



**Figure 1-17: Energy-aware cross-level classification for WSN modelling approaches**

As stated in [CHE17], none of the previously mentioned examples of WSN simulators are defined as energy-oriented simulators. Although some of these simulators includes simple energy models, assumptions are generally used to simplify the energy modelling process [JEV09] [SIN08]. Therefore, a need occurs for a detailed energy consumption model that can precisely describe the behaviour of the node and predict the lifetime of the system.

The cross-layered approach was firstly suggested to address problems related to energy modelling. For example, authors of [YIC08] suggested that the implementation of protocols at different layers in the stack can significantly improve the accuracy of the energy estimation. In parallel, early attempts extend WSN single-level models to include energy aspects. For

example, PowerTOSSIM, a power modelling extension of TOSSIM, models power consumed at the hardware level by TinyOS applications [PER08].

Following this single-level energy-oriented aspect, detailed multilevel models for energy consumption in WSN were introduced to trace the energy consumption on different levels [ZHO11] [HAN08]. Based on this approach, authors in [WAN10b] proposed an energy consumption model that enables the interaction between the software and the hardware levels. This model was then used to develop IDEA1 [WAN11], as presented in Figure 1-14.

Describing challenges of WSN energy-aware modelling, other works provide a cross-level scheme to handle energy consumption efficiently, by proposing a cross-layer energy-efficiency wireless link protocols [KIM19] [JAY07] [LIE09]. In [MER09], the cross-level approach was adapted to show only the interaction between hardware and environment models and their impact on the WSN.

Based on the bibliography, the energy modelling approaches evolved following the same concepts as WSN modelling: from the traditional stack of layers towards the cross-level approach. Nevertheless, none of WSN simulators completely adopt cross-level approach for energy-aware models. Considering all the WSN aspects from application to nodes, energy consumption modelling in WSN is a complex task and it is not a result from activities that are taking place inside one entity or level. Instead, it is rather a result of interactions between a set of parameters residing at different levels of the system [YAN13]. Thus, the use of the cross-level approach seems to be a relevant way to simulate WSN network considering their energy-awareness issue.

## 1.5. Conclusion

In this chapter, the variation of WSN was demonstrated not only in terms of applications but also hardware or software solutions and implementations. Each of these aspects is continuously subject to development where tools to test different scenarios, to compare configurations, and to analyse the obtained performance are desired. As stated previously, simulation tools can answer these requirements with advantages in terms of cost and time efforts. In this context, it has been shown that these simulation tools are based on a specific modelling approach which evolved from stack-layered to cross-level. The goal of this evolution is to provide more accurate and realistic models for WSN simulation.

In this way, the main modelling requirements for WSN is to:

- Allow models to be oriented to cover all levels of this kind of networks from application to node hardware and software by adopting a modelling approach that permits parameters to cross level's boundaries while interacting with each other.
- Be aware to the energy aspect in WSN in order to emphasise the autonomy of the node and the system lifetime.

Therefore, to answer these modelling requirements, a cross-level energy-aware approach is needed to provide models and simulation tools that accurately describe the interactions between WSN entities. For example, it has been demonstrated that energy-related issues are better addressed from a cross-level perspective because the overall energy consumption is impacting by complex interactions between entities that belong to different WSN levels. However, the above literature survey showed that cross-level energy-aware approach has not yet been completely adopted in WSN simulators. Based on these observations, the research question we are going to handle in this thesis is: **"How to provide an energy-aware model for WSN that follows the cross-level modelling approach?"**.

# Chapter 2. WIRELESS SENSOR NETWORKS SIMULATORS: MODELLING APPROACH AND ENERGY AWARENESS

2.1. Introduction ..... 37
2.2. Selection of WSN simulators ..... 38
2.2.1. Search and categorisation of papers ..... 38
2.2.2. Statistical analysis ..... 42
2.3. Evaluating WSN simulators ..... 43
2.3.1. Evaluation criteria ..... 43
2.3.2. Proposed methodology for criteria application ..... 46
2.3.3. Methodology implementation and obtained results ..... 46
2.4. Conclusion ..... 67

## 2.1. Introduction

As concluded in Chapter 1, simulation can reduce time, cost, and efforts needed to develop Wireless Sensor Networks and to deploy new applications based on it. Simulators can also handle a large number of applications and address many specific aspects of WSN. To efficiently cover this large diversity, several modelling aspects such as modelling approach and energy-awareness need to be enhanced.

To determine challenges related to these aspects, detailed study of existing simulators is needed. However, with the large diversity of WSN simulators, it is not possible to cover all of them. Thus, a set of representative simulators must be selected, and evaluation criteria had to be chosen.

In this chapter, both previous issues will be handled following the research approach illustrated in Figure 2-1.

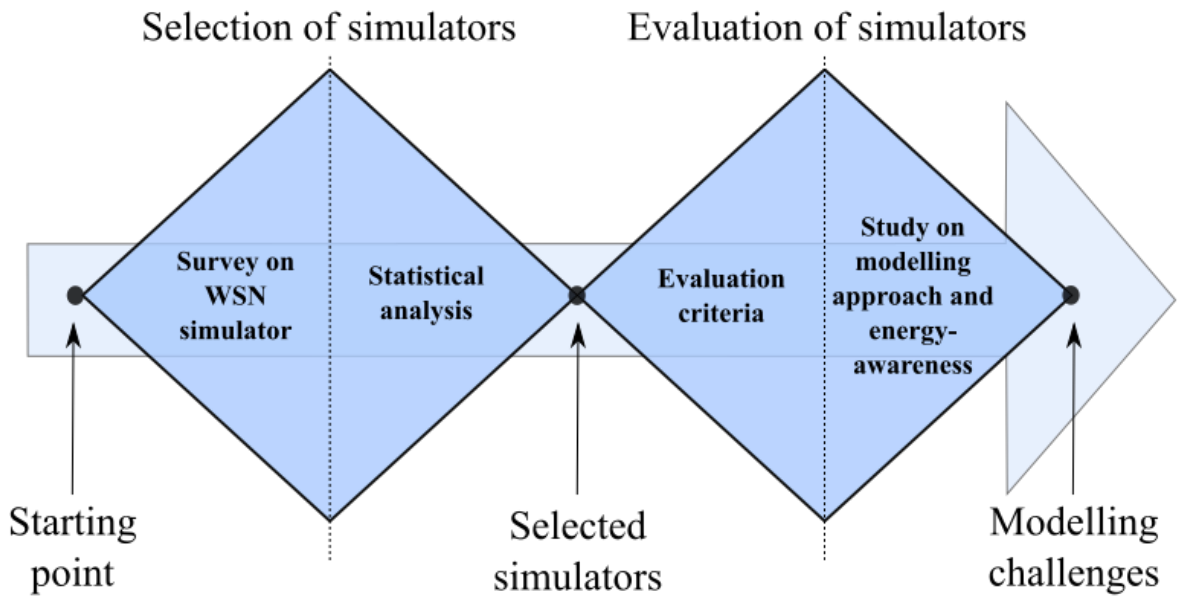



**Figure 2-1: Proposed research approach**

The starting point is the diversity of WSN simulators. The selection of the simulators is the first step, and it is divided into two phases. In the initial phase, a survey is carried out aiming to include as much as possible of WSN simulators. In the next phase, the results are statistically analysed based on the studied references, and a set of simulators is selected.

The evaluation of these simulators is composed of two phases: definition of the evaluation criteria and study of their specific aspects such as modelling approach used and implemented energy models.

The analysis of these evaluation results will allow us to highlight modelling challenges that we have to handle in order to provide an energy-aware cross-level model for WSN.

## 2.2. Selection of WSN simulators

In order to find, select, and analyse the most used WSN simulators, we followed, as illustrated by Figure 2-2, a systematic review consisting of four stages:

1) Search in online repositories of scientific works related to WSN simulators using keywords to reduce the obtained results. The result of this stage is the initial list of papers.
2) From the initial list, selection of relevant articles related to the classification and categorization of WSN simulators as well as surveys on WSN simulators; many papers might include the keywords we used, but they are oriented towards other subjects, such as energy consumption or the structure of WSN. These papers will not be selected at the end of this stage resulting in an abbreviated list of papers focused on the simulation of WSN.
3) Statistically, classify the simulators by number of citations in the selected papers. Based on the statistics obtained from the previous stage, organize the simulators into groups based of the citation numbers. Then, selection of the group with the most cited simulators.
4) Other simulators can be added to the group due to their interesting features.

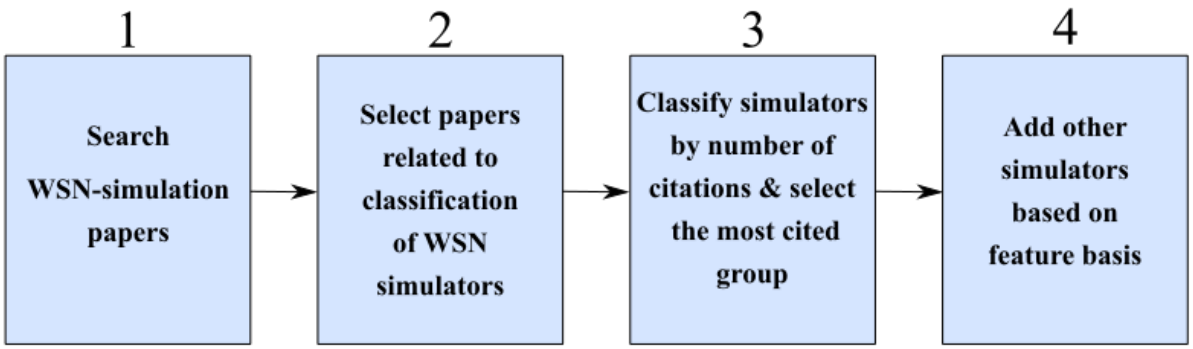



**Figure 2-2: Block diagram for the selection process**

### 2.2.1. Search and categorisation of papers

For the first phase, the search was done on the Google Scholar search engine, which provides links to scientific repositories such as IEEE Xplore, ACM, and Springer. The search was based on keywords within the targeted papers: "WSN "and "simulator", combined with tags related to

the type of the papers, such as "survey", "review", "comparison", and "evaluation". Results dated before 2005 were not included. After the execution of this phase, 47 articles were obtained (Appendix 1).

In the second phase, we have selected the most relevant articles related to WSN simulators evaluation, proposal, and comparison. The papers that do not focus on the targeted topics were excluded. For example, some papers handle issues related to the designing and evaluating WSNs, and not the simulators. The result was 34 relevant papers.

In the third phase, the selected papers were categorised according to their focus:

- Survey papers, in which authors present a general review of WSN simulators.
- Comparison papers, that evaluate and compare simulators.
- Trend papers, which are dedicated to specific research aspects and the evaluation of WSN simulators.

In the following a brief description for the selected papers that are categorised in each of the classification groups is presented:

- Survey papers. They describe WSN simulators in a general way, but there is no comparison among them, several criteria found in the surveys will be used later in the evaluation section:
  - For example, more than ten simulators are described in [SIN08] [MIS15] [KAL10] [SUN11b], in terms of type of simulator, API, languages supported, platforms supported, licenses, network support type, user interface.
  - In [EGA05], reusability, availability, performance, scalability, support for scripting languages, and GUI are the aspects considered to describe 15 simulators. In addition to these criteria, in [ABU16], hardware platforms are also considered in order to assessing different parameters required by WSN applications.
  - In [DU10] [FAH16], WSN simulators are described and classified according to their type: simulators or emulators. The work presented in [IMR10], describe twenty-one simulators considering simulation models, emulation, and testbeds.
  - More than thirty simulators are described in [MUS12], according to a classification presented by the authors, based on the target function of simulators: emulators, topology control simulators, environment and wireless medium simulators, network and application level simulators, cross-level simulators, NS2 based simulators, OMNeT++ based simulators, and PtolemyII based simulators.

  - In [RAH09], a review of network modelling and simulation tools is presented, including WSN simulators, such as NS2, OPNET, and GloMoSim.
  - Authors in [KHA11] present a review of several WSN simulation tools. They mostly focus on their suitability to simulate large-scale networks.
- Comparison papers. They include comparative studies of WSN simulators, based on self-defined criteria that evaluate the differences among simulators:
  - In [MAN13], authors make a review of some of the open source network simulators (i.e., NS2, NS3, OMNeT++, and JSIM), comparing them according to languages supported, platforms supported, licenses, network support type, user interface, and API.
  - In [KHA12], authors compare NS2, NS3, OMNeT++, and GloMoSiM. A unified scenario is applied, by simulating a MANET (Mobile Ad-hoc NETwork) routing protocol, in order to measure memory usage, computational time, and scalability, from which NS3 demonstrates the best performance.
  - Similarly, in [CHH13] [MIN16] [PES17] [SAR11] [VAS17] [MEK08] [KAT16], some of popular WSN simulators (NS2, NS3, TOSSIM, OMNeT++, JSIM, Castalia, Qual-Net, EmStar, ATEMU, Avrora, SENS, COOJA, etc.) are described and compared based on their general characteristics, their advantages, and their limitations.
  - In [CHE17], the authors evaluate more than 20 simulators and categorise them on the modelling approach and the energy awareness. In this survey, several “power-aware” simulators are highlighted such as IDEA1 and PowerTOSSIM.
  - In [GAR12], authors make a survey of available tools to evaluate WSN applications. They identify a set of models that are necessary to have in a WSN simulator: wireless propagation model, fine-grained energy expenditure model, non-linear battery model, and application model.
  - In [HEL16], authors compare Castalia, TOSSIM and NS3 based on results obtained for specific scenario. The goal is to test dynamic network reconfiguration protocols. To achieve that, the authors evaluated the energy consumption model of the simulators. They conclude that the ability to model the Telecommunication Unit states of the sensor nodes is important to estimate the energy consumption in nodes.

  - In [LAH12], authors compare NS2, NS3, TOSSIM and OMNeT++, focusing on the modelling of the energy consumption. They described how each component of the WSN nodes consume energy and show how the studied simulators model it.
  - In [JEV09], researchers evaluate four WSN simulators: NS2, Castalia, TOSSIM, and COOJA. The evaluation is made by following criteria that they defined in the paper. One of the aspects they used to make the evaluation is the energy consumption model and the ability to model non-linear batteries.
  - In [KOR09], authors review and compare the following simulators: NS2, OMNeT++, Prowler, OPNET, and TOSSIM. They highlight the features of each simulator in MAC and routing support, energy modelling, and implemented radio frequencies models.
  - In [STE11], authors compare Castalia, MiXiM, TOSSIM, and WSNet, based on topology, antenna, radio propagation, noise, Telecommunication Unit, medium access control, and energy consumption modelling.
- Trends papers. These papers focus on studying proposed approaches to evaluate WSN simulators and research trends:
  - In [PAP16], authors compile a large set of papers of wireless communication-related conferences and review the statistics about the tools (i.e., testbeds and simulators) the researchers use to evaluate their experiments. Additionally, they address the issues and challenges facing the proper use of WSN simulators such as the number of simulated node and the modelling approach.
  - In [CHI18], authors discuss topics to consider when addressing IoT issues. They present the research trends on IoT simulators in the last years, for example, the orientation towards eco-friendly systems and the support of wireless technologies such as RFID. To achieve that, authors describe existing tools that are used by researchers to prove and evaluate their findings on IoT research. They claim that more work is needed to conduct large-scale, robust and effective IoT simulation, and prototype evaluations.

### 2.2.2. Statistical analysis

In total, the selected papers include 355 citations of WSN simulators distributed among more than 80 simulators. The simulators were sorted by the number of citations, as presented in Figure 2-3.

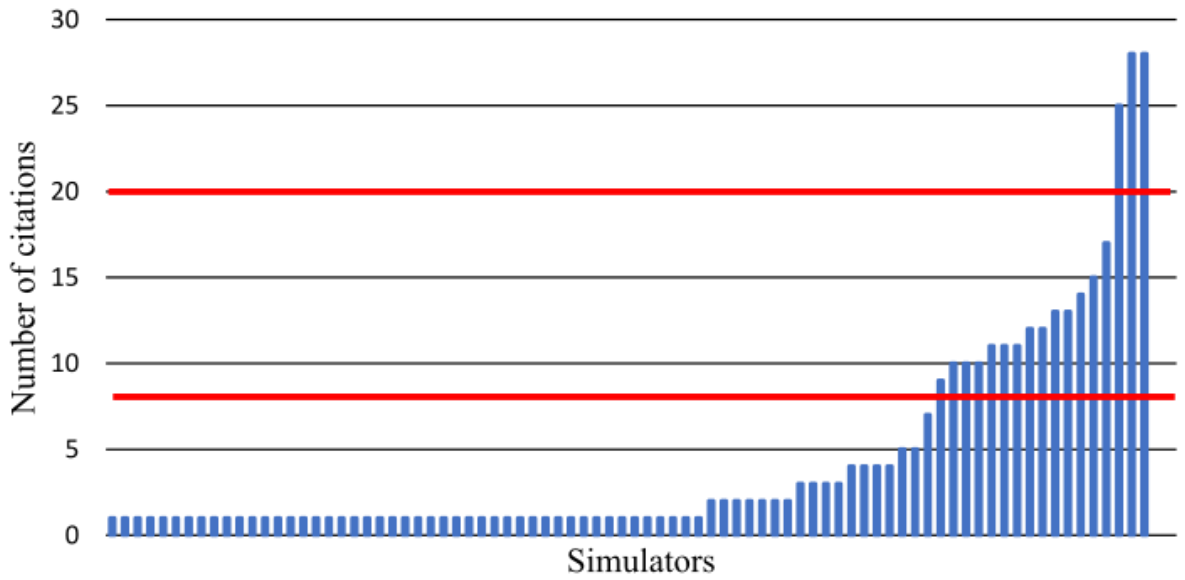



**Figure 2-3: Distribution of citations among the simulators**

According to the number of citations, simulators are categorised into three groups:

- Group 1 covers all simulators that are cited less than 8 times, the group has 68 simulators.
- Group 2 involves all simulators that have citations between 8 to 19, it contains 14 simulators: Shawn, QualNet, COOJA, Emstar, SENSE, Prowler, ATEMU, OPNET, NS3, Castalia, SENS, JSIM, AVRORA and GloMoSim.
- Group 3 includes simulators that have more than 20 citations. This group includes 3 simulators: TOSSIM, NS2 an OMNeT++.

In this chapter, only simulators included in Group 3 will be highlighted. As illustrated by Figure 2-4, NS2 and TOSSIM have the higher number of citations. They were cited in 28 papers. Furthermore, IDEA1 is added to this group due to the advanced design model, presented previously in Chapter 1, that the simulator follows. Studying this model can be useful to understand the multi-level modelling approach.

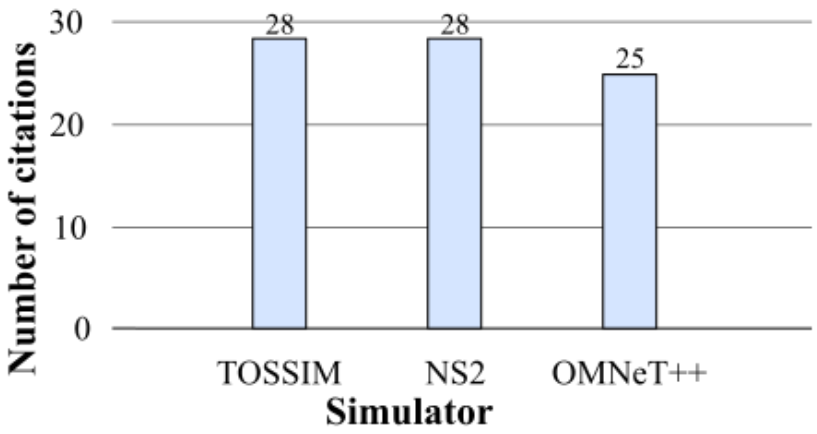



**Figure 2-4: Citations in Group 3**

## 2.3. Evaluating WSN simulators

In this section, an evaluation of the previously selected simulators is provided. The goal of this evaluation is to highlight the adopted modelling approach and how the energy aspect is addressed in each simulator. Firstly, a methodology based on the proposition of criteria is described. After that, this methodology is applied to the four simulators selected.

### 2.3.1. Evaluation criteria

In order to understand the capabilities of the selected simulators, WSN evaluation criteria, based on the survey we did in the previous section and on the WSN characteristics highlighted in Chapter 1, are proposed and defined. It includes items that can be either quantitative or qualitative.

1. **Nature of the simulator** is an assessment of how the simulation is performed. On one hand, there are simulators, which are frameworks that run system models as a function of time. On the other hand, emulators are tools that use firmware as well as hardware ressources to perform the simulation. This approach can combine both software and hardware implementation. In addition to that, physical testbeds are also used to evaluate WSN development. These testbeds are pure hardware implementation of the targeted system [IMR10]. Thus, they are not included in this criterion.
2. **Type of simulator.** Network simulators create a set of subsequent activities that imitate the studied system behaviour, they are chronologically related and usually called events. The simulators produce events following one of two approaches: discrete-event or trace-driven.

   In the discrete-event case, an initial set of events is generated, representing the primary setting of the given scenario. After the simulation starts, the initial events generate

another set of events located in the future, which in turn, when executed, will generate future-located events. The process is cyclically repeated until the end of the simulation. This philosophy is highly used in WSNs since it allows to simulate several tasks running on different nodes, that is why it is used to test new protocols and algorithms. On the other hand, in the trace-driven simulation all the events are known from the beginning of the simulation. Thus, the focuses are oriented towards the response of the modelled system under given situation [NUR09].

3. **License.** From a legal aspect, simulators can license under free (open) or non-free licenses. Definitions of the licenses can be provided as follows:
   - Free licenses, which include:
     - Public domain licenses where user can display, copy modify, distribute and sublicence the original software.
     - Permissive licences where all the rights are granted except for the right to distribute the software that is restricted to the same licence. Examples of licence from this category are Berkeley Software Distribution (BSD) and Academic Free License (AFL).
     - Protective licenses, also called copyleft, include all the rights except for the right to sublicence, as well as the restriction on the right to distribute the software. It is limited to the same licence as the original software. GNU General Public License (GPL) is an example of licence from this category.
   - Non free licences, which can be categorized into:
     - Non-commercial licenses where rights to display and copy for non-profit activities are granted. Campus-Wide License that is adapted by software like MATLAB belongs to this category.
     - Proprietary licenses which means that the software is copyrighted, and the user has restricted right based on the licence terms.
4. **User interface** answers the following question: how can users interact with the software? This item includes two aspects:
   - Graphical User Interface (GUI) is an integral part of the simulator? What is the level of details it can show?
   - Which are the supported programming languages used to interact with the simulator? Can users develop software to communicate with the simulator?

5. **Supported platforms** evaluate the usability of the simulator source code on different platforms and Operating Systems. The item covers hardware requirements such as memory size and software restrictions such as a specific distribution of an Operating System.
6. **Heterogeneity** is an evaluation of the simulator's ability to support different types of nodes simultaneously. If the simulator can run and trace interactions between different types of nodes such as sensing nodes, relays and gateways, it is heterogeneous. Otherwise, it is considered to be homogeneous [ROM04].
7. **Support for model modification** represents the capacity to modify existing models in the simulator or to implement and test new ones.
8. **Mobility model** indicates the support of the WSN simulators for modelling mobility of sensor nodes.
9. **Wireless medium model** is the ability to model the wireless medium used to estimate the radio signal strength (path loss), wave propagation (obstacles and fading), and delay between transmission and reception.
10. **Supported protocols and technologies.** This item follows the WSN model proposed in Chapter 1 including the routing protocols included. The criterion is used to compare the protocols, features, services, and technologies that are supported by the simulator.
11. **Modelling approach.** Definitions related to this item were presented in Chapter 1. As presented, simulators are built on models, which, in turn, follow a modelling approach. It corresponds to methods and techniques employed in the modelling process. Consequently, as WSNs are being modelled, the resulting models follow a specific approach and inherit the approaches properties.
12. **Energy storage and consumption modelling** at the sensor node level. As stated in Chapter 1, a sensor node is typically composed by four major components: Sensing Unit, Power Unit, Processing Unit, and Telecommunication Unit. The aspects considered to evaluate this criterion are:
    - Storage modelling consists of describing battery models (linear or non-linear) implemented in the simulator;
    - Generation modelling that handle energy harvesting devices;
    - Consumption modelling that represents the capacity to consider and simulate node units' states such as idle and sleep or specific Telecommunication Unit states: receiving and transmitting, for example.

## 2.3.2. Proposed methodology for criteria application

To apply this evaluation criteria on the selected simulators, a methodology is proposed. This methodology is based on two main methods:

1) **Analysing simulator documentation and source code**. For NS2, OMNeT++ and TOSSIM, the developers of these simulators provide user manuals, or a webpage dedicated to the documentation of these simulators. These include descriptive information of the simulator's structure as well as log files that help understanding the evolution of the simulator functionalities. For all the simulators, because they are open-source simulators, the source code is available and could be analysed.
2) **Analysing scientific literature.** As stated in the scientific literature, many researchers have used the considered simulators as part of their research projects. Thus, the information collected from scientific papers and PhD thesis is used in the evaluation process.

For all the four selected simulators, the two methods were used. The obtained results are presented using tables. However, as a focus is needed on “modelling approach” and “energy modelling” criteria, these two items are not included in the tables, but they are addressed separately in dedicated sub-sections.

## 2.3.3. Methodology implementation and obtained results

In this section, the results obtained from implementing the proposed criteria on each of the selected simulators are presented.

### 2.3.3.1. NS2

Network Simulator version 2 (NS2) is an object-oriented general-purpose discrete-event simulator built using C++ with an object-oriented Tool command language (OTcl) interpreter as a front-end [ISS09]. It was originally developed as an open source simulator for wired networks. Later, it was modified to support wireless networks. Although its developing has been stopped since 2011, it is still used today as a powerful tool to study networks, because it is one of the simulators that includes a large library in the terms of protocols and network technologies.

Network functionalities can be tracked through a file tracing. The output of the simulator is log files which contain all generated events, and it is possible to customize the file and its content as needed. By default, the simulator supports a text command interface, but a graphical

interface, called Network AniMator (NAM), was added [CAN03]. The simulator can be installed on Linux, UNIX and Mac OS but not on Windows.

NS2 is not WSN-specific. However, its capabilities were expanded to support them later. However, this simulator supports a plenty of wireless network standards (802.11 [CHE07], 802.15.4 [PET06] etc, ...) and simulates many of its features such as the energy consumption and the node's movement.

Table 2-1 provides detailed information on NS2 obtained applying the considered criteria.

**Table 2-1: Results obtained for NS2**

| **Criterion name** | **Criterion value** |
|---|---|
| Nature of the simulator | Simulator |
| Type of simulator | Discrete-event |
| License | Free: Permissive (GNU GPLv2) |
| User interface | **GUI:** yes, through NAM.<br>**Supported Languages:** C++ and OTcl |
| Supported platforms | Linux, MacOs and UNIX |
| Heterogeneity | No |
| Support for model modification | Yes |
| Mobility model | Yes |
| Wireless medium model | **Propagation models:** shadowing, 2-ray ground and free space. |
| Supported technologies and protocols | **Application layer:** DHCP, TELNET, FTP and HTTP**.**<br>**Transport layer:** TCP, UDP, SCTP, XCP, TFRC, RAP and RTP<br>**Network layer:** IPv4 and IPv6:<br>**Link layer:** HDLC, GAF, MPLS, LDP, Diffserv, Droptail, RED, RIO, WFQ, SRR, Semantic Packet Queue, REM, CSMA, 802.11, 802.15.4, and Satellite Aloha.<br>**Routing:** RIP, AODV, DSDV, DSR, NixVectorRouting, and OLSR. |

***Modelling approach***

The design process of NS2 started in 1995 and stopped in 2011. Throughout these years, different actors were involved in the development, including the Defence Advanced Research Projects Agency (DARPA) and the National Science Foundation (NSF), as well as a set of universities including the University of California and Cornell University. Without following a unified developing standard, this led to the existence of a large set of contributors that were

working on the project simultaneously. It was one of the major drawbacks that impact the evolution of this simulator.

By analysing the simulator documentation [FAL11] and the code [TAH12], we noticed that the simulator needs an input file that is an OTcl script representing the initial configurations of the desired scenario. The simulator shell converts the file to a form compatible with the core of the simulator that is written using C++. The shell is connected to following objects:

- Simulator related. The main functionality of objects and models in this set is to maintain and execute events. It includes timing objects such as the scheduler and events objects as well as the event handler model.
- Network related. It represents models and objects that serve the networking functionalities. An example of these models is the routing model, while the packet object is an example of the objects. This category is further divided into two subcategories: nodes and links related models and objects.
- Helper objects: this category include auxiliary objects that provide functionalities that are not related to timing or networking. An example of these objects is the null agent which handle the received messages by removing them from the simulation.

The shell is linked to the output utilities that is used to show time or topological representation of the results. Figure 2-5 provides a schematic overview of the structure of NS2, demonstrating the adopted modelling approach in this simulator.

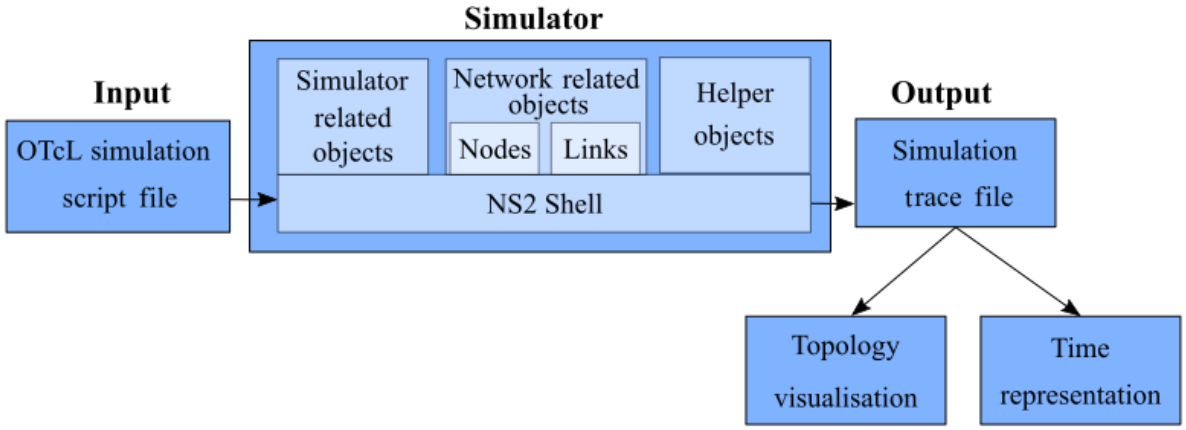



**Figure 2-5: Modelling approach in NS2**

NS2 can simulate network protocols, such as TCP and UDP, different traffic source behaviour like FTP, and MAC protocols, for examples, IEEE 802.11 and IEEE 802.15.4. Additionally, it can be used for testing routing algorithms such as Dijkstra [DIJ59]. All these mentioned themes

are network topics and considering the proposed level-based definitions in Chapter 1, NS2 can be considered as a single-level simulator.

***Energy modelling***

In NS2, the energy model is designed to address the energy consumption in mobile or stationary nodes. The model can be configured using Otcl objects, as illustrated in Figure 2-6. It includes variables to store the initial energy of the node and the total energy consumed in TRANSMISSION, RECV, IDLE and SLEEP modes, as well as the functions intended to handle energy issues such as "energy consumers". The measurement unit used in all the energy calculations is Joule.

As observed in the code and the documentation, the energy model consists of a basic class and two subclasses. The first subclass is called "AdaptiveFidelityEntity", it manages the energy consumption chronologically at the node level. The second subclass is "SoftNeighborHandler", it is used to control the node relationship with its neighbours. The two subclasses are intended to handle events generated in the simulation environment. Thus, they both inherit the Handler class, which is the base class developed for this purpose in the simulator.

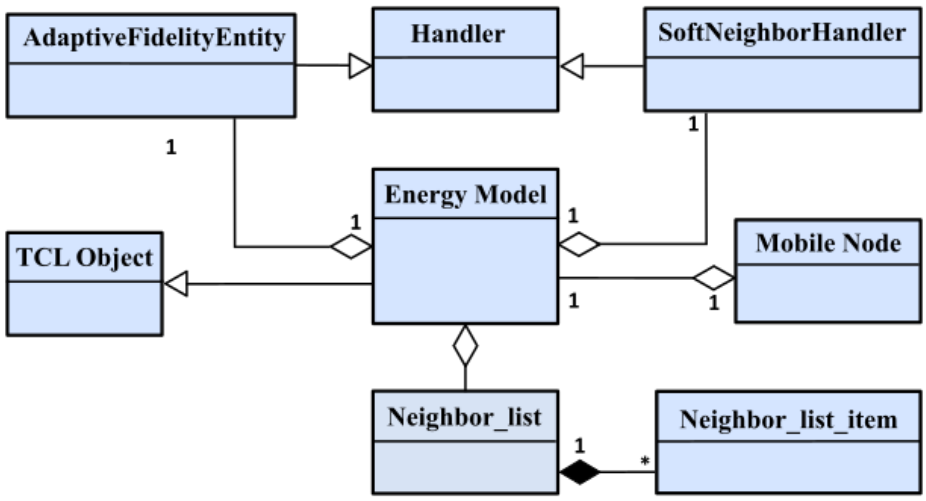



**Figure 2-6: UML Class diagram for the energy consumption model for a node in NS2**

Two structures were written for managing telecommunication channels: "Neighbor_list" and "Neighbor_list_item". The first one is a linked list used to describe neighbours. Each item represents a neighbour with all the characteristics included.

The model supports four functioning modes:

- Waiting mode represents the active state. The node is ON, but it is not moving. This mode is activated when the node is powered up.
- Sleep mode is a power saving mode with all the functionalities stopped and the consumption at its minimum.
- Idle mode is also a power saving mode, but the node is ON and waiting for commands.
- InRoute mode describes the node while it is moving.

The state transition scheme is provided in Figure 2-7. The process begins when the node is switched on and ends when it is switched off. In the waiting mode, the nodes patients until a timer that govern this state is expired. The timer value is set in the input parameters. After the timer is expired the node goes to one of the power saving states: Sleep or Idle. The state transition scheme is provided in Figure 2-7. The process begins when the node is switched on and ends when it is switched off. In the waiting mode, the nodes patients until a timer that govern this state is expired. The timer value is set in the input parameters. After the timer is expired the node goes to one of the power saving states: Sleep or Idle. From the Sleep state, the node can only wake up whereas from the Idle state, the node can pass to InRoute, OFF or Sleep state based on the configuration commands. During InRoute state, the node moves towards its new positions. When it arrives at this new position, the node state automatically changes to Waiting.

The simulator does not include a storage model such as a battery model, that is dedicated to handling energy issues from the consumption point of view. As a result, the NS2 energy model cannot estimate the system lifetime for a specific scenario. The consumed energy is computed for each state based on the power level and the time spent in that state. The power levels can be set as configuration parameters, and the time spent in the state is computed based on the simulation events.

One of the disadvantages of this model is that it is written by several programmers without unified formatting or programming rules. For example, the names of the variables and methods do not follow a certain pattern. It is obvious that they were added sequentially when needed, not based on pre-planned work.

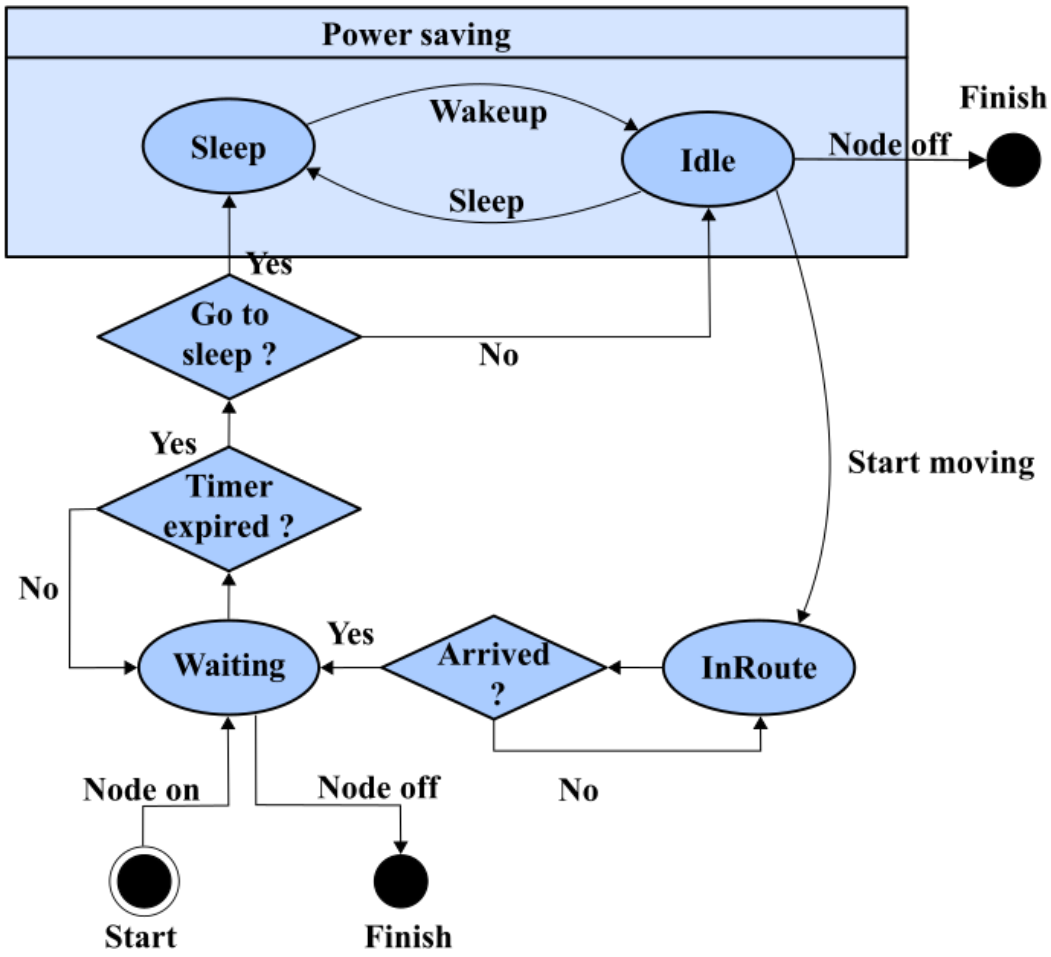



**Figure 2-7: The machine state diagram for NS2 energy model**

Moreover, the main drawback of NS2 is that other hardware components such as microcontrollers or sensors are not included in the energy model. Thus, the impact of their activities into energy consumption is not considered. In addition to that, positions of nodes are known, but inter-nodes distances do not have an influence on the consumed energy although these distances could have an impact on the Transmission Unit setting such as transmit power. Finally, NS2 energy consumption model does not integrate a battery model and support energy generation through energy harvesting solutions.

#### 2.3.3.2. OMNeT++

Objective Modular Network Testbed in C++ (OMNeT++) is an open-source component-based discrete-event network simulation platform [VAR16]. It is developed under GNU GPLv2 License. A commercial version for enterprises called OMNEST also exists. This simulator can be installed on Windows, Linux and Mac OS. The project started in 2001, and it is still ongoing with version 6 released in December 2020.

OMNeT++'s essential advantage is the theoretically infinite-level modelling ability. A hierarchical complex multi-level system can be easily modelled using a built-in domain-specific network descriptive language called “NED” [WEH10]. Based on OMNeT++, researchers have built their own simulators, for example Castalia [LIN03] and INET [LIN04]. This simulator has also a built-in Graphical User Interface, as well as the possibility to trace the simulation result through log files.

In addition to that, the simulator has been WSN-oriented from the first release. OMNET++ supports 802.11 [MAU09], 802.15.4 [CHE07] standards and a set of routing protocols such as AODV [ARI08] [SOM09], as well as various models to simulate the harvesting, the storage, and the consumption of the energy [CHE09].

Table 2-2 provides detailed information on OMNeT++ obtained from the considered criteria of our proposed evaluation methodology.

**Table 2-2: Results obtained from OMNeT++**

| **Criterion** | **Criterion value** |
|---|---|
| Nature of the simulator | Simulator |
| Type of simulator | Discrete-event |
| License | Free: Permissive (AFL) |
| User interface | **GUI:** yes, a built-in GUI is available.<br>**Supported Languages:** C++ and NED |
| Supported platforms | Windows, Linux and MacOS |
| Heterogeneity | Yes |
| Support for model modification | Yes |
| Mobility model | Yes |
| Wireless medium model | **Propagation models:** Free-space, log-normal shadowing, Rayleigh fading, 2-ray ground, Rician and Nakagami fading.<br>**Other models:** Background noise, obstacle loss and propagation. |
| Supported technologies and protocols | **Application layer:** HTTP, DHCP, BitTorrent, P2P video and voice Streaming.<br>**Transport layer:** TCP, UDP, SCTP, RTP and RTCP.<br>**Network layer:** ARP, HIP, IGMPv2, IGMPv3, IPv4, IPv6, MCoA, MIPv6.<br>**Link layer:** 802.11, 802.11p; 802.1e, 802.15.4 WiMAX, LDP, LTE and PPP.<br>**Routing protocols:** AODV, BGP, GPSR, link-state routing, OSPF, PIM, and RIP. |

***Modelling approach***

OMNeT++ is built using a modular approach called "component architecture". In this approach, each component is considered as a basic unit. A component can connect with each other using connections over which they can exchange messages.

Based on the simulator documentation [VAR16], components can be classified, based on their architecture, as follows:

- Simple component: a single and basic design unit. It cannot include other components.
- Compound component: a complex component that includes other simple or compound components.

There is no restriction on the connections between simple or compound components.

As suggested in Figure 2-8, all components are included inside an entity called the network which represents the whole system that is being modelled. Additionally, the network has a simulation library that provides messages and events handling mechanisms such as event scheduling and message exchange.

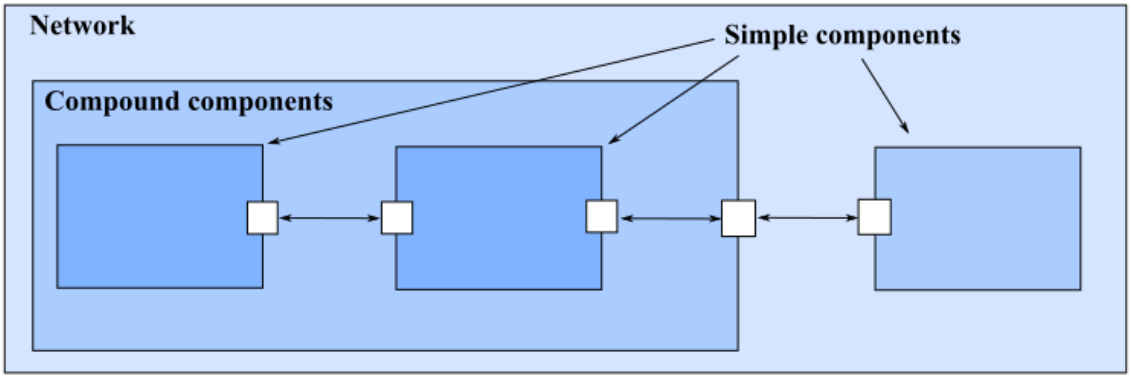



**Figure 2-8: OMNeT++ Design concept**

For example, by analysing simulator code [VAR20], in OMNeT ++, both the network and link layers are compound components. The link-layer includes 802.11 and 802.15.4 compound components. The internet layer has compound components as well, such as IPv4, IPv6 and ICMP. IPv4 is linked to both the ICMP and the link-layer using connections.

Application designers and researchers map the network elements, such as nodes or protocols, into a hierarchal architecture of components. From the design point of view, there are no limits to the number of layers that can be created to precisely simulate elements' behaviour.

As a result, considering the level-based classification provided in Chapter 1, we conclude that OMNeT++ has a flexible design architecture. The simulator can represent multiple-layered topology inside the network level only. Thus, it can be considered as a single-level simulator.

***Energy modelling***

The first version of OMNeT++ did not include an energy model. However, in the following versions, an energy-oriented hierarchical structure has been developed. The energy model in OMNeT++ supports the representation of energy in two forms: charge/current (CC) and energy/power (EP). The four quantities are considered using units such as Coulomb, Ampere, Joule and Watt respectively. Separated models are provided for the two forms. Thus, the user can select one of these forms based on their needs.

By analysing both [VAR16] and [VAR20], we noticed that the model is divided into submodels according to their functionalities. Based on that, as presented in Figure 2-9, submodels are sorted out into three groups: energy consumption, power generation and energy storage.

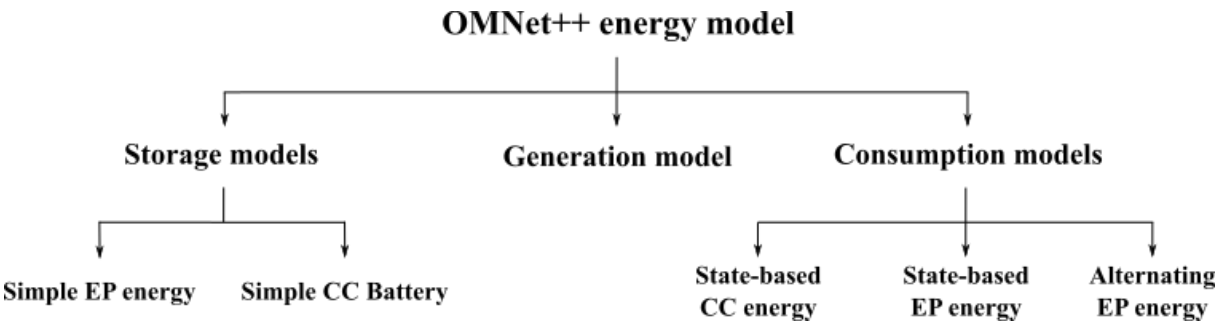



**Figure 2-9: Energy model structure in OMNeT++**

In the next sections, starting from the storage models, the energy submodels in OMNeT++ will be briefly explained. Then, the generation and consumers submodels will be also presented. Finally, a global overview is added to clarify how the submodels work together.

***2.3.3.2.1. Storage models***

The simulator supports storing energy inside the node using models developed especially for this goal. Consistent with the two available representations related to the energy purpose in the simulator, there are two energy storage models. The first one called “Simple CC Battery” model is related to CC representation. The second one deals with the EP representation and is called the “Simple EP Energy Storage” model.

The Simple CC Battery model has been developed to provide a mechanism for describing how the energy stored in the node can be consumed. The model describes the energy stored in the node in terms of capacity. It is only compatible with the CC consumption model that use the

current drawn from the battery, as described below using Figure 2-12. The battery model defines the initial value of the stored energy and nominal capacity, nominal value of the voltage and nominal internal resistance of the considered battery. Other dynamic variables are used to trace changes in the energy stored inside the battery such as residual stored charge.

The Simple Energy EP Storage model is also used to simulate the storage of energy within the node. It is compatible with the EP consumption models, as shown below in Figure 2-12. This model defines a maximum and an initial amount of energy stored in the node, and a threshold of the minimum amount of energy changes to be considered for the log file.

Figure 2-10 shows a class diagram of the energy storage models in OMNeT++. We note that the two models are connected to other modules that handle state's changes of the nodes such as "Node Status" and "LifeCycle controller" classes. Additionally, these two models are linked to other modules in the global energy model through two "Energy Storage Base" modules: the EP and CC.

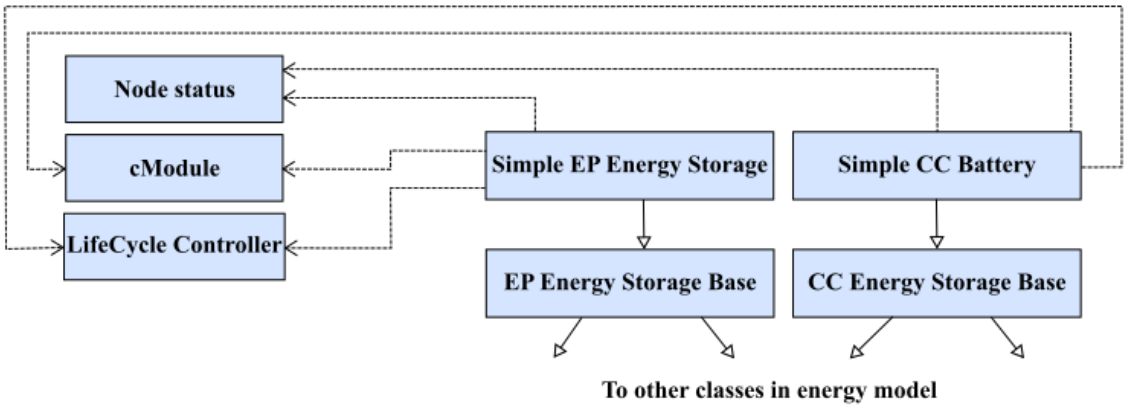



**Figure 2-10: Energy storage models UML class diagram in OMNeT++**

#### *2.3.3.2.2. Generation models*

OMNeT++ supports an energy model that allows simulate solar panels, or any power sources dedicated to energy harvesting solutions.

As illustrated by Figure 2-11, OMNeT++ supports only one energy generation model called "Alternating EP Energy generator". It is used as a linear source of energy and is compatible with the EP representation. The energy generation model supports CC representation as well, but there is no actual CC model that is implemented. However, it is possible to extend the existing model, which is called "ICC Energy Generator", to cover a CC implementation.

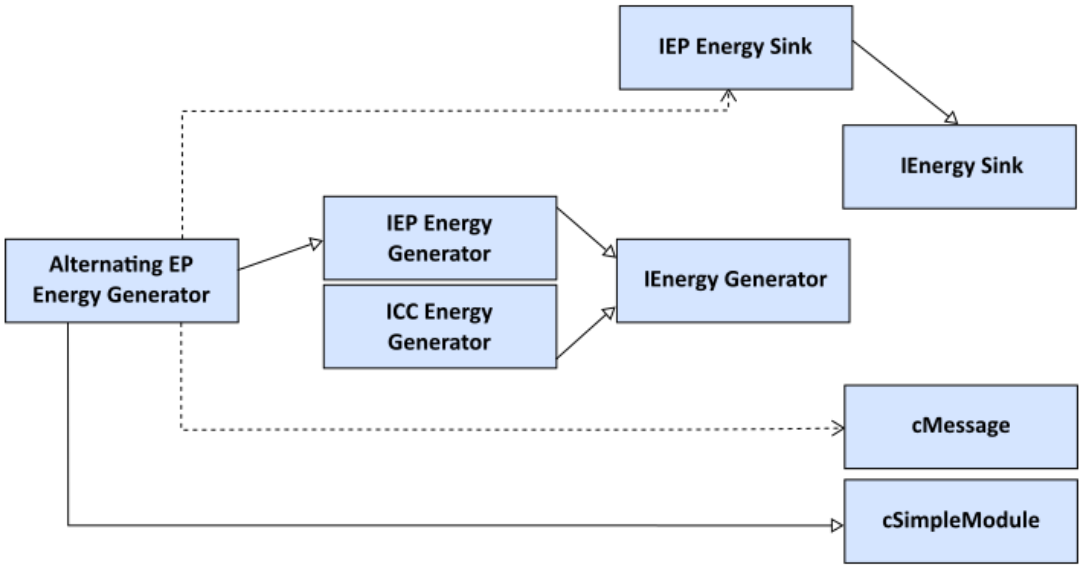



**Figure 2-11: Energy Generation class diagram in OMNeT++**

The Energy Generator model is connected to the Sink models to provide a way to transfer the generated energy in order to exploit by the Storage models. In addition to that, the Energy Generator model is connected to other auxiliary models that provide basic services. For example, "cMessage" module provides the means to exchange messages, that have a standard format, between simulators modules. This module could be used to connect Generation modules to Storage modules exchanging information about the energy generated.

The model distinguishes between two modes for the node: the generation and the sleep modes. In the generation mode, a configuration is needed to provide the model with a set of power values. Thus, the model randomly selects constant power for a random time interval to simulate the energy generation with an unpredictable behaviour. In sleep mode, the generated energy is zero for another random time interval. Periodically, the node is switching from one mode to another based on predefined configuration of two adjustable timers.

#### *2.3.3.2.3. Consumption models*

In addition to energy storage and generation, OMNeT++ supports three models of energy consumption. Two of these models are dedicated for the consumption related to the Telecommunication Unit, while the third is a general model for energy consumption, without considering the cause of the consumption. The two models dedicated to the Telecommunication Unit energy consumption are called the "State-based CC Energy Consumer" and the "State-based EP Energy Consumer" because they respectively deal with the CC and EP representations proposed in OMNeT++. In these two models, the consumption process depends on the state of

the radio link to determine the current or power level of the Telecommunication Unit. There is no state machine for the nodes, the states are imposed by the communication channel.

The “State-based CC Energy Consumer” is dedicated to calculating the current consumption due to the radio activities in the node. This model includes the current levels of the transmit, receive and idle states. Additionally, it provides the ability to include link layer parameters such as bit rate, header and preamble lengths in the calculation. Moreover, the model can calculate the consumption of the node in three cases without radio activities. The first case is the no-operation mode and it covers the consumption when the node is supplied, this is equivalent to the self-discharge of the battery. The second case is the sleep mode, and the third is the transient state when the node’s state is changing among any of the previous cases. When Telecommunication Unit is active, there are three modes for the receiver: idle, listening (busy) and reception. The idle mode is the state of the receiver in which the circuit is ON, but it is not receiving. The busy mode is a state of the receiver when it is ready to receive, but reception has not been yet started. The reception mode is the state of the receiver when it is receiving data from the network. The transmitter has only two modes: the idle and the transmission modes. The first is a state on which the transmitter is ON, but it is not transmitting, and the second is the transmission state. The consumption is calculated based on the current drawn in the state and the time spent in it.

The “State-based EP Energy Consumer” is similar in terms of structure and mechanisms to the previous model, but it calculates the consumption in a given node in Watts instead of Amperes. Thus, the energy consumed is calculated based on the power level of the state and the time spent in this state.

The “Alternating EP Energy Consumer” is designed to simulate the consumption of the node in general, without focusing on what causes the consumption. Thus, it could be used to consider energy consumption of the node Processing and Sensing Units. This model supports two node's mode, namely consumption and idle mode. The node is switched into the consumption mode periodically, and the switching is repeated at fixed intervals that can be adjusted in the configuration file. The consumption mode lasts for a specified period that could also be adjusted. In this mode, the node consumes energy at a constant rate (constant power). Otherwise, the node is automatically turned into the idle state and does not consume any energy.

The class diagram for the energy consuming models in OMNeT++, illustrated in Figure 2-12, is obtained by analysing the simulator code. The class diagram shows the hierarchical structure

where three Energy Consumer modules inherit an abstract energy consumer class called "IEnergy Consumer" which resides at the higher level of the structure. The inheriting is implemented through an intermediate level of the hierarchy which contains two modules that differ in the calculation method (EP or CC).

Four other modules are attached to the consuming model: cListener, iRadio, cSimpleModule, and cMessage. They are used to connect the consumption models with other energy modules such as the storage models or radio channel but also to provide services such as information exchanging.

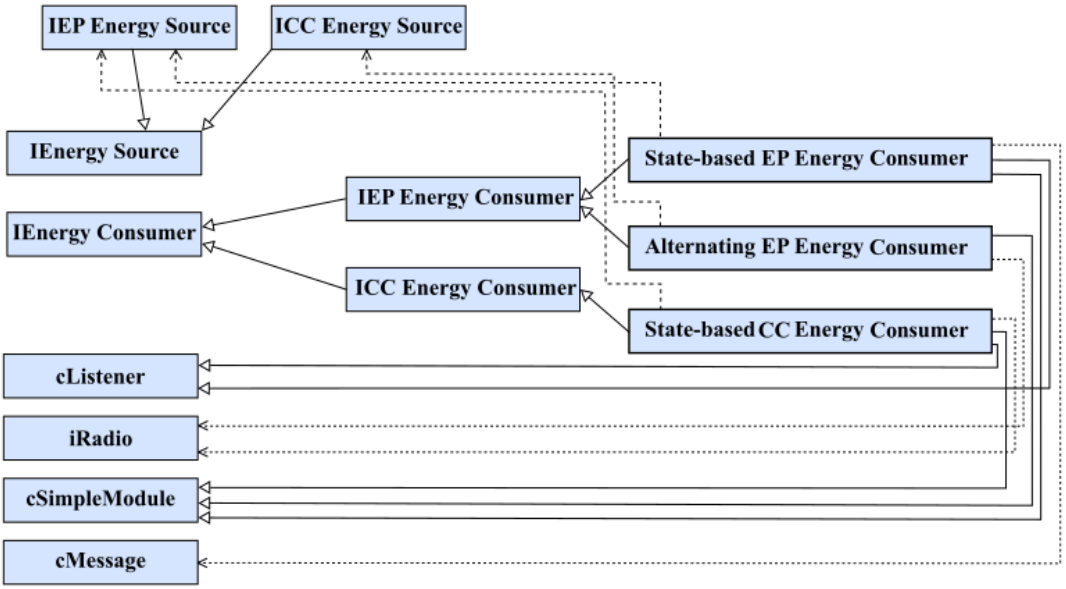



**Figure 2-12: Energy consuming model class diagram in OMNeT++**

#### *2.3.3.2.4. Global overview of energy model*

The energy model in OMNeT++ consists of three submodels: storage, generation and consumption models. In a WSN node, energy is generated, stored and consumed. It explains why, in OMNET++, the three types of models should not function separately, but they rather operate simultaneously. To achieve this, these submodels are connected together through a three-level hierarchical structure shown in Figure 2-13.

The energy model is structured as follows: at the top level, there are four base abstract classes: "IEnergy Sink", "IEnergy Consumer", "IEnergy Generator" and "IEnergy Source". These classes include properties that are used by all other classes in the lower hierarchical levels. In the middle level, there is a set of abstract energy classes that connect the base abstract classes from the top level with the energy models in the bottom level. The bottom level classes contain energy model classes. They are related to a number of simulator utility classes that provide a

set of non-energetic services and functions such as information exchange and message formatting.

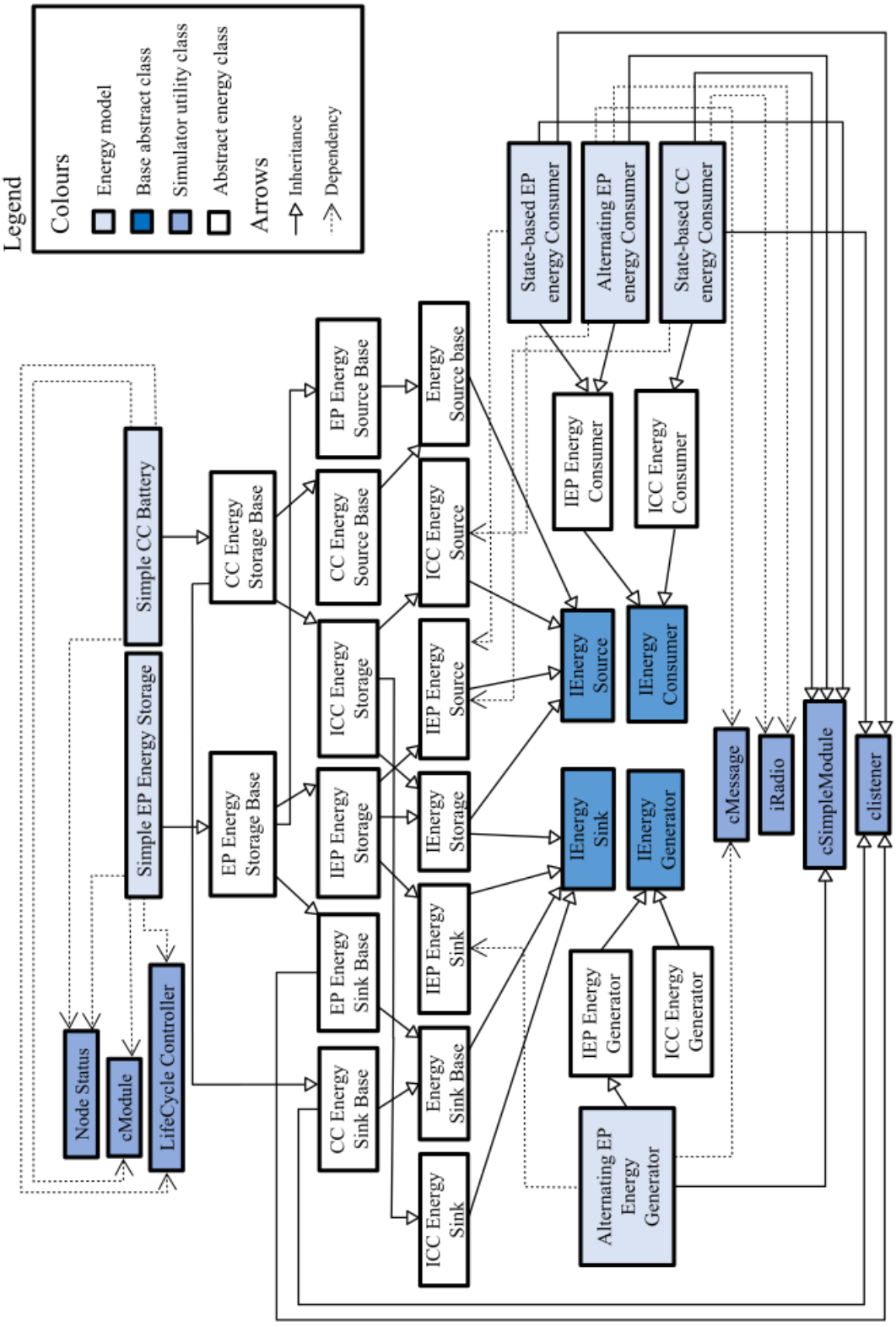



**Figure 2-13: Full energy model for OMNeT++**

#### 2.3.3.3. TOSSIM

Tiny Operating System Simulator (TOSSIM) is a specific-purpose component-based discrete-event emulator developed at University of California, Berkeley [LEV03]. The main purpose for developing TOSSIM was the need to provide a way to test TinyOS applications in an environment as close as possible to the real nodes [LIN01]. TOSSIM supports different programming languages such as Python, C++ and NesC. NesC is a component-based, event-driven, extension of C language, used to build applications for the TinyOS platform.

Several test platforms were proposed, all contain the same CPU (ATmega128L) and different Telecommunication or Sensing Units. For example, regarding the Telecommunication Unit, Mica2 uses CC1100 (including a proprietary protocol), while Micaz employs CC2420 (802.15.4 - ZigBee). Although this emulator was mainly designed to support wireless networks, it does not include the ability to track energy consumption or the mobility of the nodes. Later, some scenarios combined with additional hardware was proposed to highlight the consumption of each component such as CPU, EEPROM, Telecommunication Unit, sensors and LEDs [SHN04] [PER08]. In addition to that, an extension called MOB-TOSSIM was introduced to handle the mobility of the nodes [DER12].

TOSSIM first releases supported a graphical user interface, called TinyViz [SAF11], as well as the ability to track the output through log files. In later versions, the graphical interface was not supported, and the output could only be traced by means of log files.

Table 2-3 provides detailed information on TOSSIM obtained from the considered criteria.

**Table 2-3: Results obtained from TOSSIM**

| **Criterion** | **Criterion value** |
|---|---|
| Nature of the simulator | Emulator |
| Type of simulator | Discrete-event |
| License | Free: Permissive (BSD) |
| User interface | **GUI:** yes, through TinyViz.<br>**Supported Languages:** Python, C++ and NesC. |
| Supported platforms | Linux and Windows |
| Heterogeneity | Yes |
| Support for model modification | Yes |
| Mobility model | Yes, through MOB-TOSSIM |
| Wireless medium model | **Propagation models:** lognormal shadowing.<br>**Other models:** noise model. |
| Supported technologies and protocols | TinyOS applications are supported only. |

***Modelling approach***

The major difference, when comparing with the two simulators previously studied, is that TOSSIM is an emulator. TOSSIM is TinyOS-oriented, thus, emulations are limited to hardware compatible with this Operating System. TOSSIM is installed on a host and it runs TinyOS which allocate the host's resources to create emulated nodes. TinyOS compatible hardware platforms such as Mica, Mica2 and MicaZ are available in TOSSIM. In order to understand the emulator characteristics, the documentation [TOS20] and the code was analysed [TIN13].

In TOSSIM, the WSN topology is represented as a graph: vertices stand for the nodes, while wireless channels are the edges. As presented in Figure 2-14, each node is represented using a TinyOS component graph that handles a queue of discrete events. The node has a small number of emulated hardware and mechanisms for ADC models, radio modules, and communication services. EEPROMs are also emulated and memory space is allocated for each node.

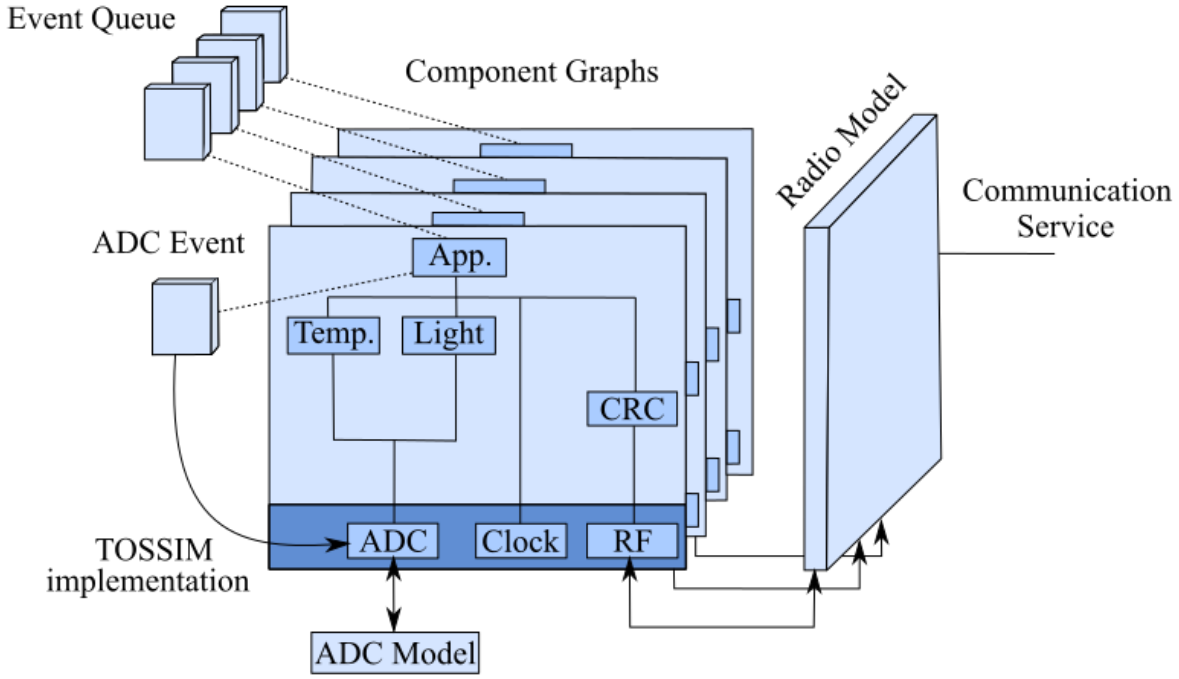



**Figure 2-14: TOSSIM modelling approach**

TOSSIM considers event duration such as the time necessary for each analog-to-digital conversion, as well as all periodicity of events such as interrupts created in the system. However, TOSSIM does not consider CPU execution time.

As the simulator focuses only on emulation of node hardware, applications are simple and support for network protocols is limited. Thus, TOSSIM handles only the circuit level, and because of that we consider it as single-level simulator.

***Energy Modelling***

TOSSIM does not model energy consumption nor power draw by the WSN node. Thus, it does not include energy consumption, battery or energy generation models. However, the user can trace activities of components such as CPU, LEDs and RF module. For example, when these components are turned on or off, the collected transitions can be considered later by an energy consumption model. To achieve that, a central "struct" object called "NodeEnergy" is used, as illustrated in Figure 2-15.

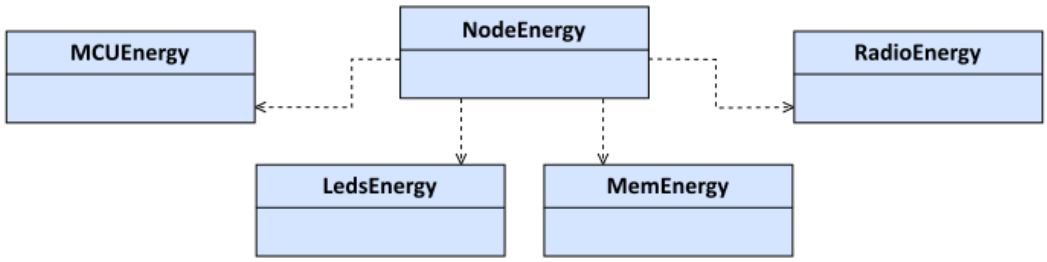



**Figure 2-15: Class diagram for objects related to energy consumption in TOSSIM**

The main structure is linked to child objects that represent node components as follows:

- MCUEnergy is a structure dedicated to trace the energy consumed by the Processing Unit. It includes variables to store the consumed energy for each of the microcontroller states: idle, standby, extended standby, energy-saving, on, and down states. Additionally, the structure includes a variable to trace the energy consumed by the Analog-to-Digital Converter.
- LedsEnergy is a structure that traces the energy consumed by the LEDs, where each node can have up to three LEDs.
- RadioEnergy is a structure used to trace and store energy consumed by the Telecommunication Unit. It follows the state of the circuit: sending, receiving and synchronize.
- MemEnergy is a structure dedicated to trace the energy consumed by the memory-related operations, this includes reading and writing.

Every emulated node has an instance of "NodeEnergy" object attached to it, which means that TOSSIM can only ran homogenous scenario where all nodes has the same structure.

The use of the energy model is simple. Every time an activity is detected, the function related to the circuit and the corresponding activity is called. As a result, an amount of energy is consumed, and the corresponding energy consumption register is updated.

#### 2.3.3.4. IDEA1

The hIerarchical DEsign plAtform for sensor networks exploration (IDEA1) is a SystemC-based simulator for WSN design, where systemC is a C-based hardware description language. IDEA1 provides a possibility to model heterogenous networks from system and node point of views [NAV12]. According to the simulator's website, it was released in 2011.

The simulator was built in Qt framework. It uses C++ for the wireless network model, as well as SystemC for hardware modelling. However, the simulator is only supported on a distribution of Linux called CentOS [WAN11a]. The simulator uses XML configuration files as well as a Graphical User Interface [NAV11].

In IDEA1, hardware and software are separately modelled. For example, IEEE 802.15.4 is modelled as a software to represent a link protocol for the wireless connection, it can be used in beacon and non-beacon modes. On the other hand, MRF24J40 and CC2420 circuits are modelled as Telecommunication Units, both supporting 802.15.4. For the Processing Units, models for PIC16LF88 and MSP430 are available.

Table 2-4 provides a detailed description of the simulator, based on the evaluation methodology.

**Table 2-4: Results obtained from IDEA1**

| **Criterion** | **Criterion value** |
|---|---|
| Nature of the simulator | Simulator |
| Type of simulator | Discrete-event |
| License | Free: Permissive (AFL) |
| User interface | **GUI:** yes<br>**Supported Languages:** C++ and SystemC |
| Supported platforms | Linux (CentOS) |
| Heterogeneity | Yes |
| Support for model modification | Yes |
| Mobility model | No |
| Wireless medium model | Yes; two propagation models were implemented:<br>1) free space model<br>2) ITU (International Telecommunication Union) indoor model |
| Supported technologies and protocols | **Link layer:** 802.15.4<br>**Hardware:**<br>• **Microcontroller:** PIC16LF88 and MSP430<br>• **Telecommunication Unit:** MRF24J40 and CC2420 |

***Modelling approach***

Based on the definitions provided in Chapter 1, IDEA1 adopts a multi-level approach to model WSN. The study of IDEA1 is based on the simulator documentation [IDE14a] [WAN11b] and analysis of its code [IDE14b]. According to that, a WSN model includes three levels: the node, the network, and the physical environment.

Figure 2-16 shows a schematic diagram for IDEA1 modelling approach. The node level is composed of two parts: hardware and software. On one hand, hardware includes models for the node's components: sensors, transceiver, CPU, and battery. Sensors are controlled using a stimuli generator configured by means of an interface specifying how the physical parameters in the environment vary in spatial and temporal terms. On the other hand, software covers the Operating System, applications and the network protocol stack. The network model serves as the wireless communication medium and is used by nodes to exchange data and control frames.

Software and hardware can interact through the CPU, it is an intra-level interaction. The environment and the wireless medium levels can only interact with the hardware components of the node level, this is the only cross-level interaction implement in IDEA1. The sensors can read environment parameters and the Telecommunication Unit is connected to the wireless medium. So environmental parameters cannot affect the wireless network model or interact with its parameters directly. There are no other cross-level interactions and all interaction must follow the hierarchy of the model levels.

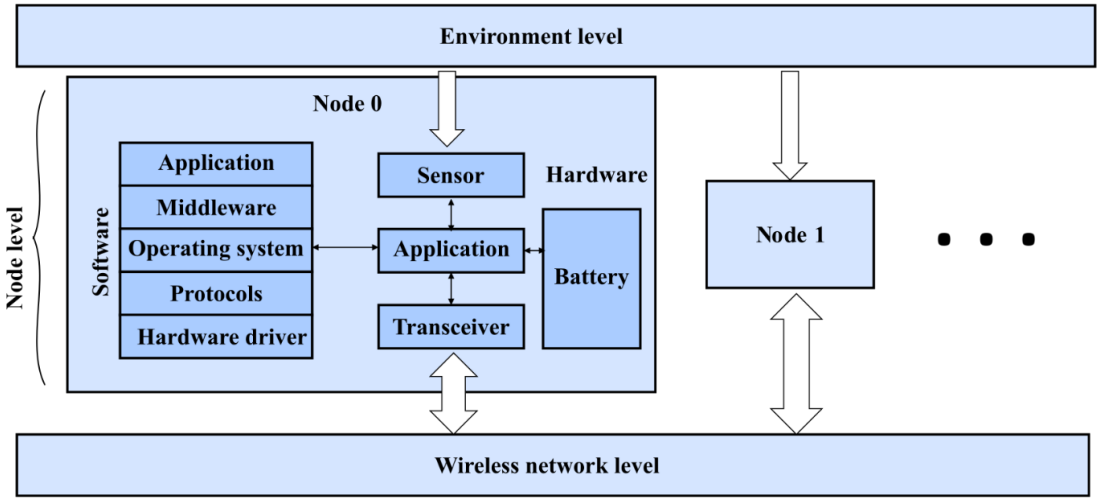



**Figure 2-16: IDEA1 modelling approach**

***Energy modelling***

The energy model in IDEA1 includes a battery model associated with two machine-state diagrams for the Processing and the Telecommunication Units. Based on state-machine attached to each unit, the model can trace the units' energy consumed and estimate system lifetime.

Figure 2-17 provides an overview of the IDEA1 schematic structure. The microcontroller is the core of the model, it collects the sensors data that is programmatically generated through the Stimulus block. Moreover, it is connected to the transceiver which can send data to the network through the proxy block or receive data from it. As illustrated by this schematic structure, the sensors are not directly modelled, and their energy consumption is not considered. On the contrary, the microcontroller and transceiver are modelled, and they have associated machine-states. For each state, energy parameters can be configured. During the simulation, the consumed energy is traced considering time spent in each state.

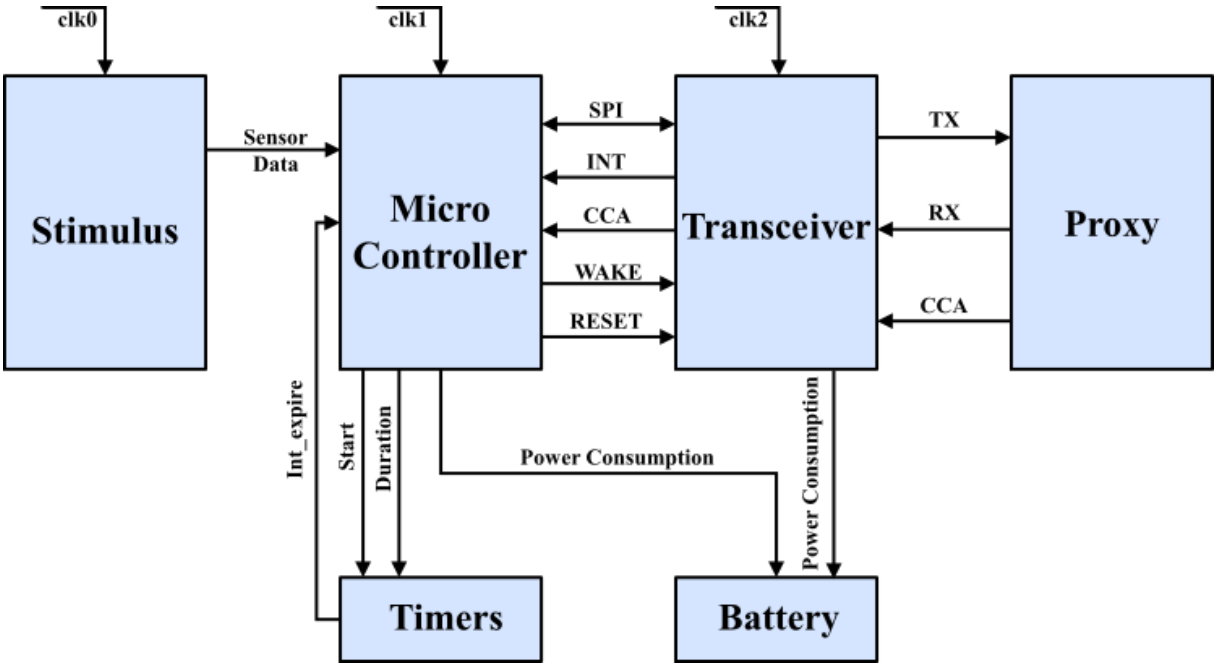



**Figure 2-17: Schematic diagram for the structure of IDEA1**

The Processing Unit can be in one of the five following states: SLEEP, IDLE, SENSING, receiving (RX) and sending (TX). The SLEEP state is energy state with the minimal consumption. In the IDLE state, the unit is awake, but without any activities. The other three states represent activities managed by the processing unit, each of which is corresponding to the state name. The transitions between the states are represented as sequential events and set using the scenario configuration file.

The node can be configured to send data messages which consists of one or several data segments that are collected from the sensors. Both the lengths of the message and data segment can be configured. SENSING state can be achieved from SLEEP or IDLE states, as shown in Figure 2-18. The transition between each state is triggered by a configurable timer. If the total length of the segments reaches the message length, the state is automatically transited into TX. Otherwise, and based on the configuration file, the transition can be toward SLEEP or IDLE states. From TX state, IDLE, SLEEP or SENSING are accessible. Transition to RX state is only possible from IDLE if an interrupt event arrives from the transceiver. From RX, all the other states are available.

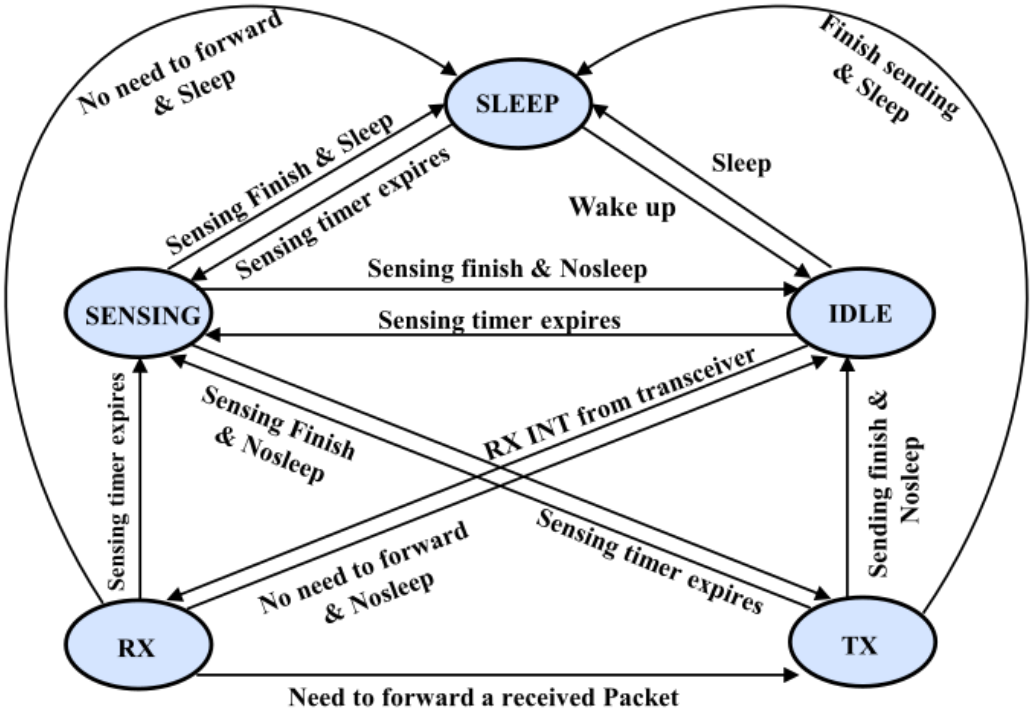



**Figure 2-18: State-machine diagram for Processing Unit in IDEA1**

As illustrated by Figure 2-19, Telecommunication Unit state-machine contains four states. The SLEEP state represents the case when the Telecommunication Unit is off, and the IDLE represents the case when it is on but neither sending nor receiving. SLEEP state is only accessible as an initial state or through the IDLE state. Two more states are dedicated to radio activities, they are sending and receiving states, called TX and RX respectively. The transition between these two states is possible as well as for the IDLE state.

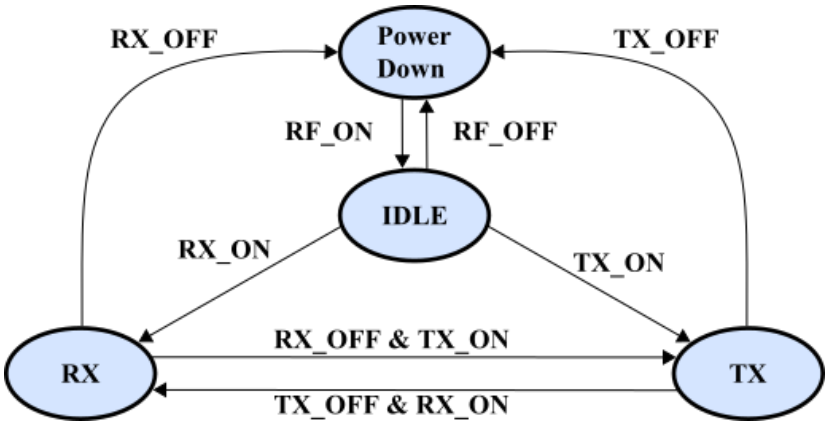



**Figure 2-19: State-machine diagram for Telecommunication Unit in IDEA1**

## 2.4. Conclusion

In this chapter, we first presented a methodology for evaluating Wireless Sensor Network simulators using a set of defined criteria. Then, based on this evaluation methodology, we evaluated four selected WSN simulators focusing on their adopted modelling approach and the description of the energy model implemented in each of which. In the following, we present a synthesis of this evaluation in order to highlight the strengths and weaknesses of each considered simulator:

- NS2 is a single level simulator focusing on network algorithms and protocols, there is no mean to add node level components. Consequently, hardware modelling is not supported, thus, heterogeneity cannot be achieved. From an energy perspective, NS2 has an energy model that supports energy storage and consumption but not generation. Moreover, because hardware is not completely modelled, the consumption of Processing Unit and Sensing Units cannot be included in the energy calculation, resulting in tracing the consumption of Telecommunication Unit only.
- OMNeT++ is a single-level simulator that follows a modular modelling approach. It is oriented to the network level with simple models for node level. From an energy point of view, OMNeT++ supports two forms of calculation, either using charge and current, which uses Coulomb and Ampere respectively, or power and energy, using units Watt and Joule. This adds complexity to the simulator causing a repetition of the structure and calculations in the code with only different units used. In addition, there is no modelling for the hardware of the nodes except for the Telecommunication Unit. Thus, energy consumption resulted from Sensing Units or Processing Unit activities is not considered.

- TOSSIM is a single-level emulator limited to hardware compatible with TinyOS implementation. This emulator does not include an energy model. However, using PowerTOSSIM, as an extension, allows tracing changes in component states that are used as inputs for the available energy model. Nevertheless, although TOSSIM focuses on hardware, sensors are not implemented. Thus, the sensors' energy consumptions are not included. Additionally, the emulator lacks a battery model excluding the possibility to estimate the system lifetime.
- IDEA1 is a multilevel simulator that support models for both hardware and software of the node. The simulator includes three levels: environment, node and wireless medium. As demonstrated, when parameters interact with each other, they need to respect the sequence of the levels because the cross-level approach is not completely supported. IDEA1 includes an energy model that can trace consumption of the Processing and Telecommunication Units but not the Sensing ones.

Results obtained from this evaluation process confirms that cross-level modelling approach is not always implemented in the studied simulators which are, excepted IDEA1, the most cited in the literature. Moreover, concerning the energy-awareness purpose, it has been highlighted, as shown in Table 2-5 where heterogeneity is added next to the energy-awareness, that none of the evaluated simulation tools allows to simulate completely energy consumption in a WSN from different level of abstractions at the same time.

**Table 2-5: Comparation between selected simulators based on the modelling approach, energy-awareness and Heterogeneity**

| **Item** | | **NS2** | **OMNet++** | **TOSSIM** | **IDEA1** |
|---|---|---|---|---|---|
| **Modelling approach** | | Single level | | | Multi-level |
| **Energy-awareness** | **Consumption** | Yes | Yes | No Energy Model | Yes |
| | **Generation** | No | Yes | | No |
| | **Battery** | Yes | Yes | | Yes |
| **Heterogeneity** | | No support for hardware | | Yes | |

Therefore, a set of issues related to the modelling approach and to energy awareness is identified to address the need, in the WSN design stages, to provide a cross-level energy-aware model for this kind of network.

**Objective 1: Define the requirements for WSN cross-level modelling approach**

**Question 1.1: How to define cross-level interactions between parameters residing in different levels of the model?**

The transition from multi-level toward cross-level modelling approach requires defining a new type of interactions that can pass over the level's boundaries to create new relationships between parameters that describe WSN properties.

**Question 1.2: How to integrate cross-level interactions in multilevel architecture of WSN?**

In the adopted approach, different levels must be considered in order to address both high and low abstraction levels of a WSN and to evaluate the impact of each level on specific WSN components. These levels could be, for example: application needs, environment, network and node's software and hardware.

**Question 1.3: How to model the concepts belong to different levels simultaneously?**

Cross-level modelling approach should be able to reflect different aspects of the simulated system at the same time. For example, at the highest level, the user should be able to define different scenario based on the application requirements like duty cycle. At the lowest level, heterogeneity should be supported allowing the user to accurately define circuits characteristics and providing the ability to model several types of nodes.

**Objective 2: Include the energy awareness capacity in the modelling approach from cross-level perspective**

**Question 2.1: How to handle energy from cross-level perspective?**

The model shall be able to handle energy-related operations such as consumption for different degrees of details. From the system point of view, the model should be able to predict the lifetime (low detail degree). In order to trace the energy consumptions of different nodes, higher detail degrees are necessary as well. This includes consumption of different circuits in a node and different activities in circuits. Thus, interactions between entities from these three levels (system, node and circuits) need to be considered.

**Question 2.2: How to increase the model's accuracy including energy-aware perspective (energy conversion)?**

The model should be able to handle models of auxiliary hardware within the node such as regulators and energy harvesting devices. Each circuit has specific supply voltage level and can be linked to the battery using different configurations. The model should be flexible to include these configurations and reflect their effects on the energy calculation. Auxiliary hardware has no direct effects on the functionality of WSN neither from the telecommunication perspective nor on system-environment interaction. However, design choices related to these components can significantly impact the energetic performance.

In the next chapter, we define and provide a cross-level energy-aware model for WSN design, aiming at addressing the problematic issues mentioned above.

# Chapter 3. CROSS-LEVEL ENERGY-AWARE MODELLING FOR WSN

3.1. Introduction ..... 72
3.2. Cross-level modelling approach ..... 72
3.2.1. Concept definitions ..... 73
3.2.2. Cross-level model overview ..... 74
3.2.3. Cross-level interactions in and between the nodes ..... 76
3.2.4. Model description ..... 80
3.3. Energy-awareness modelling ..... 90
3.3.1. Regulator modelling ..... 90
3.3.2. Node lifetime modelling ..... 92
3.4. Conclusion ..... 95

## 3.1. Introduction

Chapter 1 showed that cross-level modelling approach and energy-awareness are required to provide accurate and realistic WSN models. After that, in Chapter 2, based on a proposed methodology, a set of selected simulators were analysed. Using the obtained results, challenges related to these two previously mentioned aspects were identified considering the limitations of the studied simulators.

As concluded in Chapter 1 and demonstrated in Chapter 2, it has been demonstrated that the cross-level modelling approach is not always implemented in the studied simulators which are the most cited in the literature. This could be explained by the fact that WSN, as a system, is complicated to model entirely. Therefore, existing simulators generally adopt only a level or a multi-level approach because they were developed to focus on one aspect of the WSN, such as the evaluation of new protocols for NS2 or platform emulation in TOSSIM. However, as stated in this conclusion, the cross-level modelling approach is essential, in an energy-aware context, to precisely model energy aspects that are result of features and functionalities that are distributed among different levels. Examples of these aspects are the influence of the voltage regulators and the power levels of each circuit phases. As a result, extended issues as the estimation of node lifetime become possible.

To address these issues, we propose, in this chapter, a cross-level energy-aware model for Wireless Sensor Networks. To present our model, this chapter is divided as follows. First, our cross-level modelling approach and its implementations are introduced. After that, a detailed description related to the energy-aware aspects of this model is provided.

## 3.2. Cross-level modelling approach

In this section, first, definitions and level descriptions, we developed for our cross-level modelling approach, are introduced. Then, a detailed node component-based demonstration of cross-level and intra-level interactions between WSN parameters is provided. Finally, a detailed model design is proposed associated with the mathematical description of the specified interactions.

### 3.2.1. Concept definitions

As stated previously, the step one of our modelling process consists on providing considered definitions that are going to be used during the implementation of the proposed model.

Firstly, the definitions regarding the structural concepts of our model, shown in Figure 3-1, are proposed:

- *Object*. An object is a part of the modelled system that provide one or more specific functions. Each object includes a set of parameters that describe its properties.
- *Parameter*. A parameter is a configurable value that represents one property of the object which it belongs to.
- *Level*. A level is a subdivision of the model hierarchy focusing on a dimensional perspective of the WSN such as topology or node. A level contains a set of objects providing similar functionalities.
- *Interaction*. An interaction is defined as a relationship between two parameters that could mutually affect each other. If the parameters belong to the same level, the interaction is called *intra-level*. If the two parameters reside at different levels, it is a *cross-level* interaction.

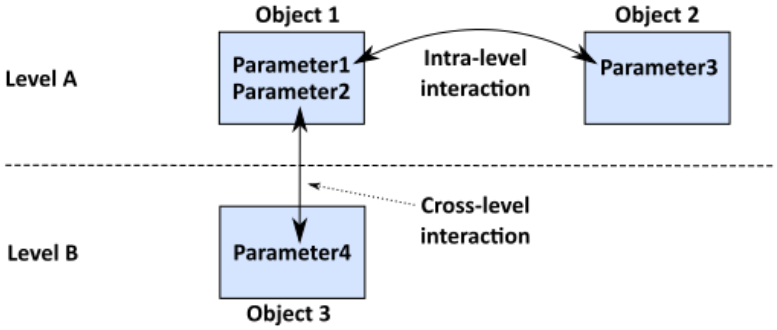



**Figure 3-1: Structural concepts used in the proposed model**

Secondly, we provide specific definitions to address chronological and energetic aspects of the proposed model:

- *Phase*. As illustrated in Figure 3-2, a phase is characterised by a time period and one or more current/power levels of an object. Each current/power level corresponds to one operational state of this object. For some objects that represents hardware components, the operational state is identical to the circuit state (sleep, wake-up, idle…), and the consumed power level is constant during the phase. However, for other objects, the phase might include several circuit states. For example, the exchange phase of the

Telecommunication Unit might include sending and receiving operations, each of which has a distinct power level.

- *Pattern.* In the proposed model, we assume that, from energetic point of view, the node activities are periodic. Thus, as shown in Figure 3-2, they can be described using a pattern which is defined as a set of subsequent phases for all the circuits of a node. Therefore, each pattern could be used to describe the behaviour of the node in terms of energy consumption.
- *Pattern frequency* ($F_p$). The pattern frequency represents the number of pattern occurrences in the time unit. For example, in Figure 3-2, $F_p$ is equal to $t/N$ Hz.

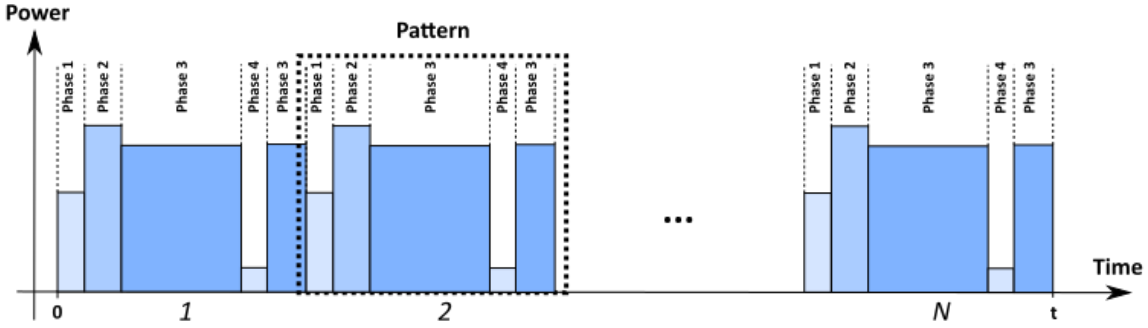



**Figure 3-2: Concept of the pattern used in the proposed model**

## 3.2.2. Cross-level model overview

As illustrated by Figure 3-3, to provide a comprehensive WSN model, our proposal is composed of four levels:

- The Use case Level (UL) is the higher level of the model. UL is related to the scenario description which depends on the considered WSN application. This level is mainly described using parameters such as the pattern frequency $F_p$ or the payload length.
- The Topology Level (TL) focuses on the network organisation, such as node's distribution, roles and positions. Parameters related to the network topology, like the distance between nodes, or the high-level protocols headers belong to this level. Therefore, TL describes how high-level protocols and algorithms affect WSNs performances in terms of energy.
- The Node Level (NL) is used to model the node behaviour (phases order, activity sequences, …) and node hardware structure. This level defines the node types and patterns. By defining different type of nodes, heterogeneity is handled in this level. Additionally, energy consumption of each node is modelled in this level based on the node's type and pattern.

- The Circuit Level (CL) is particularly used to describe hardware circuits that composed each type of node. This level covers the modelling of specific Telecommunication Units, Processing Units as well as Sensing Units and batteries. Selection of these units depends on the considered application. Therefore, hardware-oriented parameters related to the selected unit references, such as power level or supply voltage, reside in this level. Moreover, configurable link protocol parameters, which are hardware-implemented in the Telecommunication Unit, such as the bit rate and the link layer header fit in this level.

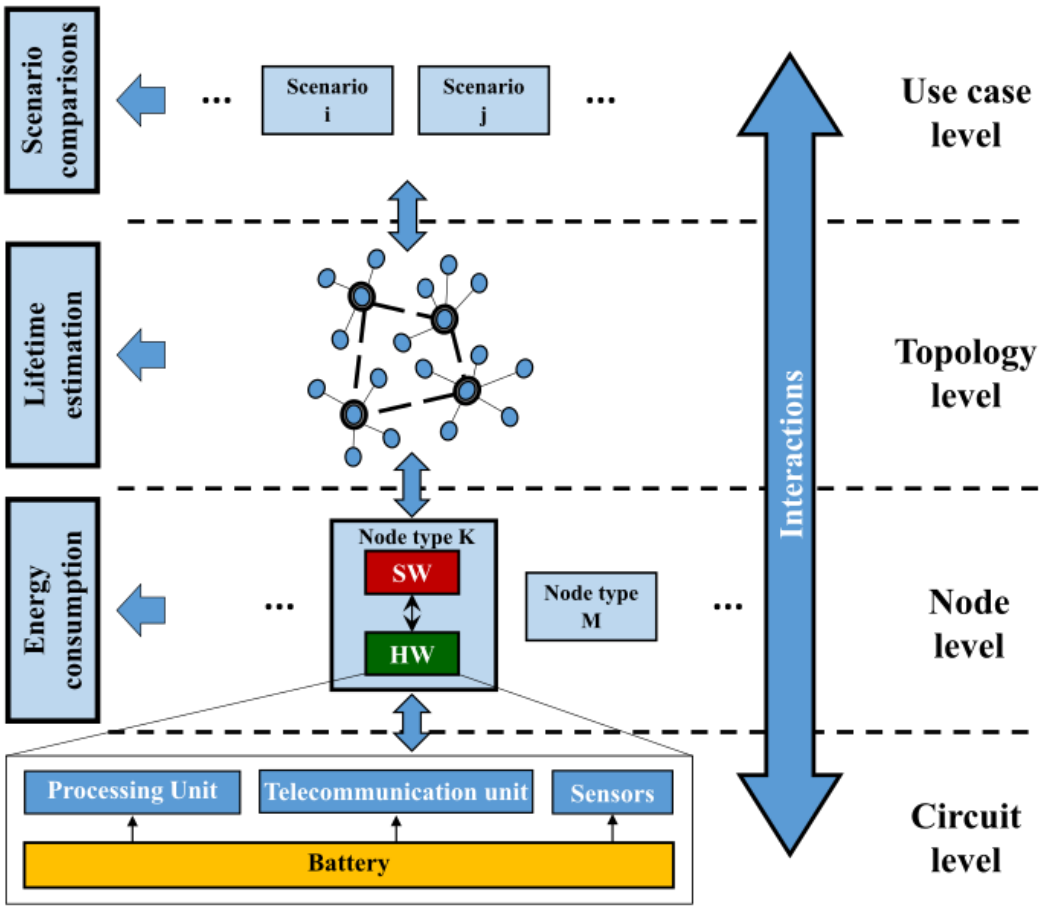



**Figure 3-3: The Proposed Cross-level model for Wireless Sensor Networks**

As presented in the previous figure, the first main objectives, in an energy-awareness context, of this cross-level model are to:

- Predict energy consumption at the Node level;
- Estimate system lifetime at the Topology level;
- Compare different scenarios at the Use case level.

### 3.2.3. Cross-level interactions in and between the nodes

In order to determine the overall architecture of our model and its description, not only parameters, and the levels they belong, but also the interactions between these parameters needs to be revealed. To achieve that, parameters and their interactions, both intra-level and cross-level, will be highlighted, in this section, for each unit: Processing Unit, Telecommunication Unit and Sensing Unit. As a result, these descriptions could be used to better understand how WSN parameters interact between each other showing the interest of the cross-level overview adopted in our model.

#### 3.2.3.1. Processing Unit

The Processing Unit activities are defined as node-based. It means that the interactions will only occur with other onboard units. There will be no direct interactions with the physical environment, such as the telecommunication channel or physical phenomena to be sensed. In a real system, the Processing Unit controls the Telecommunication Unit activities in terms of actions (wake-up, transmit, receive…) and sleep times. Moreover, it handles sensors in the same way. Thus, in our model, in the active phase, we have to include the time required for the achievement of those missions.

Thus, managing the tasks of each node unit is the major activity of Processing Unit when it is in active phase. Figure 3-4 illustrates a general time sequence of the Processing Unit in which three different phases are identified. Only at the beginning, the Processing Unit is in an initial sleep phase, in which, all node units are sleeping. Then, the time sequence starts when the Processing Unit wakes up and enters the active phase where it manages all other components activities. Finally, it turns back to the sleep state.

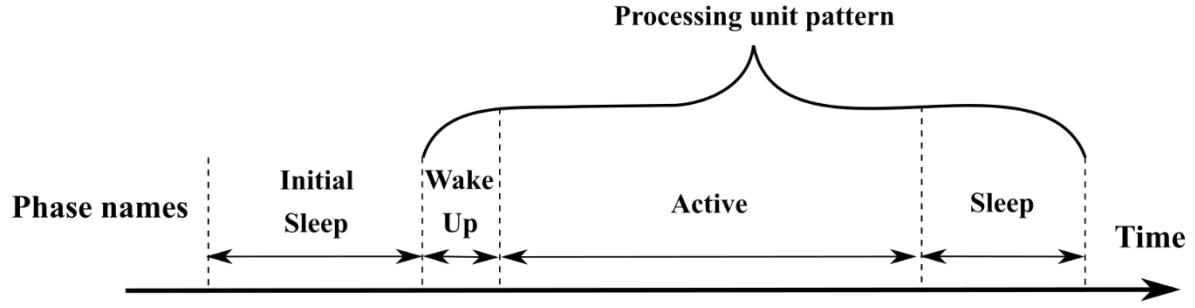



**Figure 3-4: General time sequence considered to define Processing Unit activities**

As illustrated by Figure 3-5, the construction of the time sequence is impacted by parameters that belong to different levels. For example, activities sequence (UL) and phases order (NL) are

parameters related to other units and govern the duration of the active phase. This is the first step in the construction of the Processing unit time sequence.

In the next step, the corresponding current levels for each phase are derived from the datasheet of the selected Processing Unit, and these are CL parameters. As a result, the instantaneous electrical current consumption in one pattern for the Processing Unit could be computed. After that, the supply voltage of this unit, which is also a CL parameter, is included to calculate its power consumption. Finally, pattern frequency (UL), that defines the periodicity of each Processing Unit time sequence, representing the concept of node duty-cycle, and battery energy (NL) are added to calculate the energy consumption, enabling the estimation of the Processing Unit effects on the node lifetime.

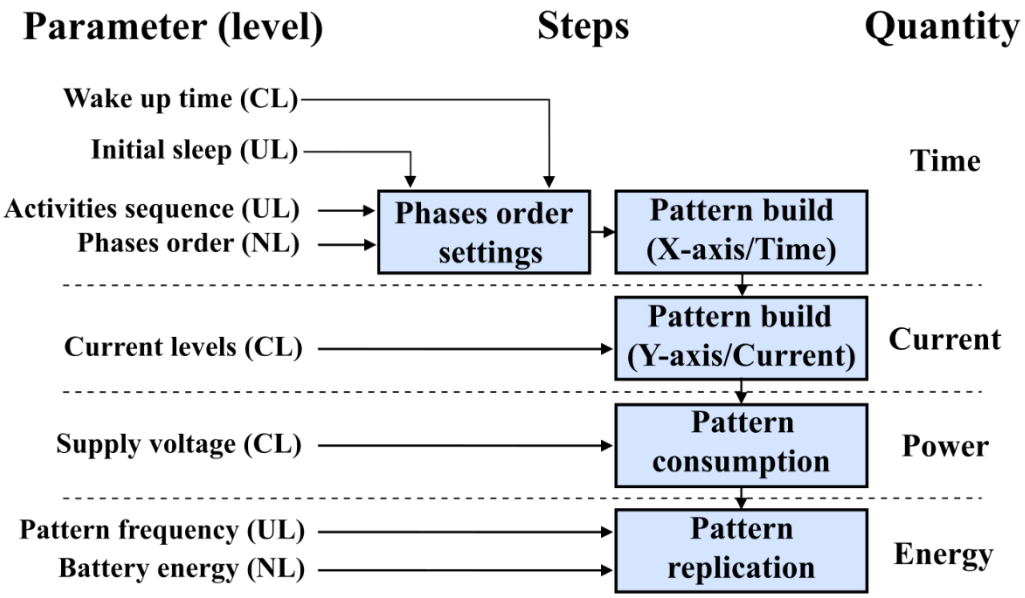



**Figure 3-5: Parameter-based pattern construction for the Processing Unit**

### 3.2.3.2. Telecommunication Unit

As stated in Chapter 1, Telecommunication Unit is connected to the communication channel where the data is being sent or received. Therefore, Telecommunication Unit activities are both topology-based and node-based, which means that the interactions take place between nodes as well as inside each node. In our model the Telecommunication Unit is governed by the Processing Unit in terms of duty cycle.

In general, the main objective of the Telecommunication Unit is exchanging data with other nodes to serve the application of the WSN. Figure 3-6 illustrates a time sequence of the Telecommunication Unit in which four different phases are identified.

First, the Telecommunication Unit is in the sleep phase. Second, the Processing Unit wakes the Telecommunication Unit, and the nodes need to access the communication channel. Then, when the node is ready to send or receive data, the exchange phase starts, and it is followed by the acknowledgment phase to guarantee the reception of the exchanged data. Finally, it turns back to the sleep state.

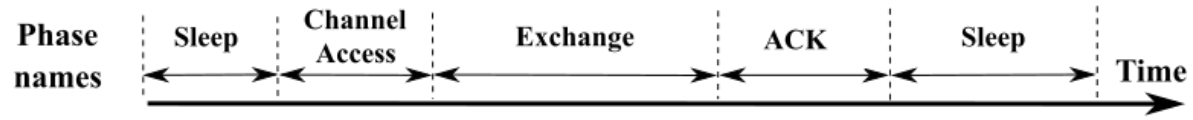



**Figure 3-6: General time sequence considered to define Telecommunication Unit activities**

One telecommunication consumption pattern can be built as illustrated in Figure 3-7. First, activities sequence (UL) and phases order (NL) are to be considered. The activity sequence helps to specify the actions taking place, i.e. sending or receiving frames, while the link protocol defines the phases and their order within each activity. For example, in the sending activity, the order of phases is accessing the channel, exchanging data frame, and then, wait for the acknowledgment.

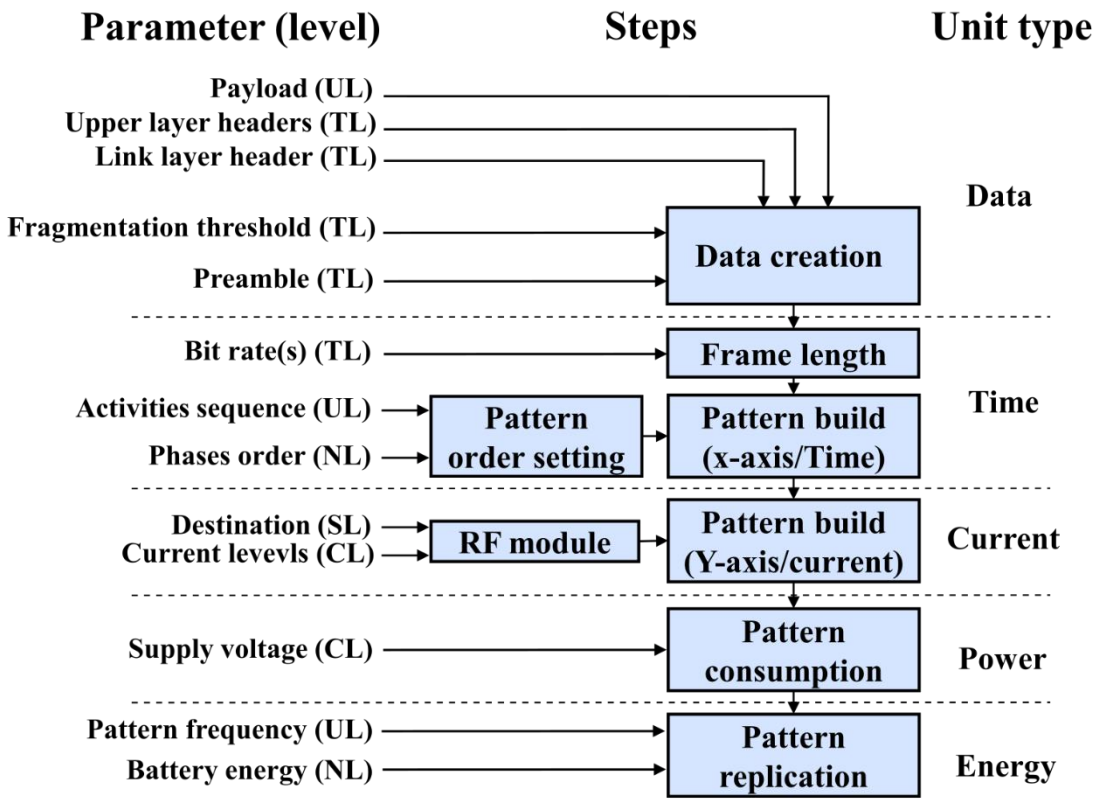



**Figure 3-7: Parameter-based pattern construction for the Telecommunication Unit**

After that, the sequence of activities is to be matched with the power levels (CL) provided by the telecommunication unit datasheet. This includes the consideration of the distance (TL) that

has a direct impact on the power level of the sending activity. Then, the energy consumed is calculated. In the last stage, the frequency of the pattern $F_p$ (UL), as well as the initial amount of the energy stored in the battery (CL) are useful to estimate the system lifetime.

#### 3.2.3.3. Sensing Unit

The Sensing Unit activities are both node-based and topology-based. In one hand, Sensing Unit has interactions with physical environment, particularly when measuring phenomena such as light temperature or humidity. However, Sensing Unit has no direct interactions with other nodes. On the other hand, this unit is controlled by the Processing Unit in terms of duty cycle and activity sequence.

In our model, we considered that Sensing Unit has only two phases related to its operation: sleep and active. These phases are led by the Processing Unit. When the Sensing Unit is awake and measuring, it is in the active phase. Otherwise, it is in the sleep phase. Figure 3-8 shows a typical time sequence of a Sensing Unit where two measurement activities are taking place.

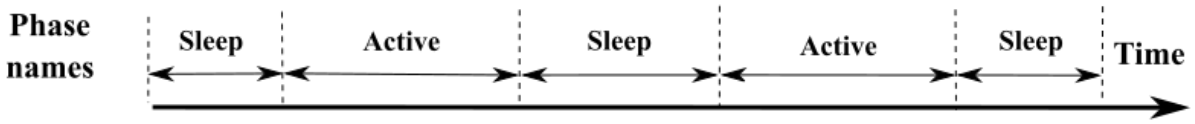



**Figure 3-8: General time sequence considered to define Sensing Unit activities**

When sensing a physical phenomenon, a sensor measures a physical quantity and converts the measured value into an electric signal. The conversion time (CL) is specified in the sensor datasheet provided by the manufacturer, and it is the time when the energy consumed by the sensor is the most. The result of this stage is a set of bits captured from the sensor environment.

The activity sequence (UL) of the sensor is then to be added, this includes the number occurrences in which the measurements are taking place in one pattern, as well as intervals between measurements. When the sensor is not sensing, it is in the sleep mode. The result of this stage is a set of time values represent the phases of the pattern.

Next, the power level (CL) derived from the datasheet is included. For each time value, there will be a corresponding current consumption. After that, the battery voltage will be added, and energy consumption for one pattern of sensor activities is calculated. Finally, simulation time, pattern frequency (UL) and the battery energy level (NL) are to be included. Thus, an estimation of the Sensing Unit activities impact on the node lifetime can be generated. Figure 3-9 presents the parameters related to the Sensing Unit and their interactions.

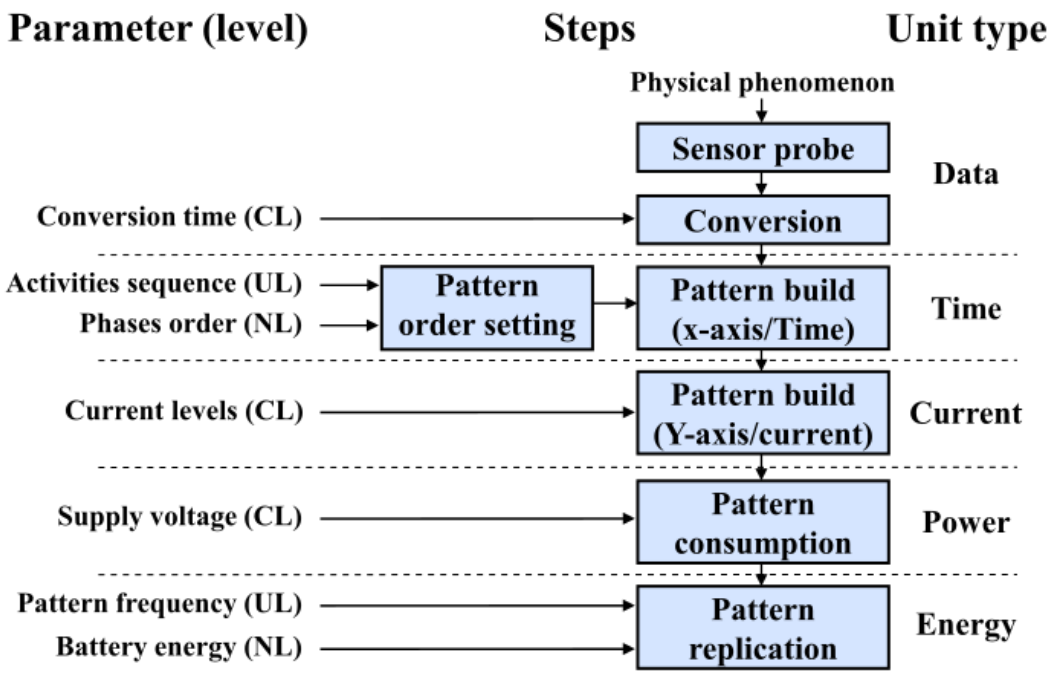



**Figure 3-9: Parameter-based pattern construction for the Sensing Unit**

## 3.2.4. Model description

Based on the interactions previously described, we propose, in this section, a level-based model that allows for cross-level interaction to take place. The structure of this model is firstly presented before providing the detail description of its implementation.

### 3.2.4.1. Structural model description

The proposed model is divided into four levels of abstraction based on the overview described early in this chapter in Figure 3-3. Using the structural concepts defined previously and presented in Figure 3-1, each level includes a set of objects, represented in a rectangular shape in Figure 3-10, that are characterised using a set of parameters and the associated behaviour description. There are two types of connection between objects:

- Derivation connection. This kind of connection links two objects: the derivative and the source. The derivative object inherits all source attributes and behaviour, plus it has extensions that serve the derivation purpose.
- Child-parent connection. It couples two objects: the child and the parent. The child object is part of the parent which can have more than one child object from the same type. This kind of connection is used, in the proposed model, to reflect the hierarchical aspect of its structure. For each connection, a parent could have one or more children. If the number of is more than one, the corresponding connection is tagged with "*" in Figure 3-10.

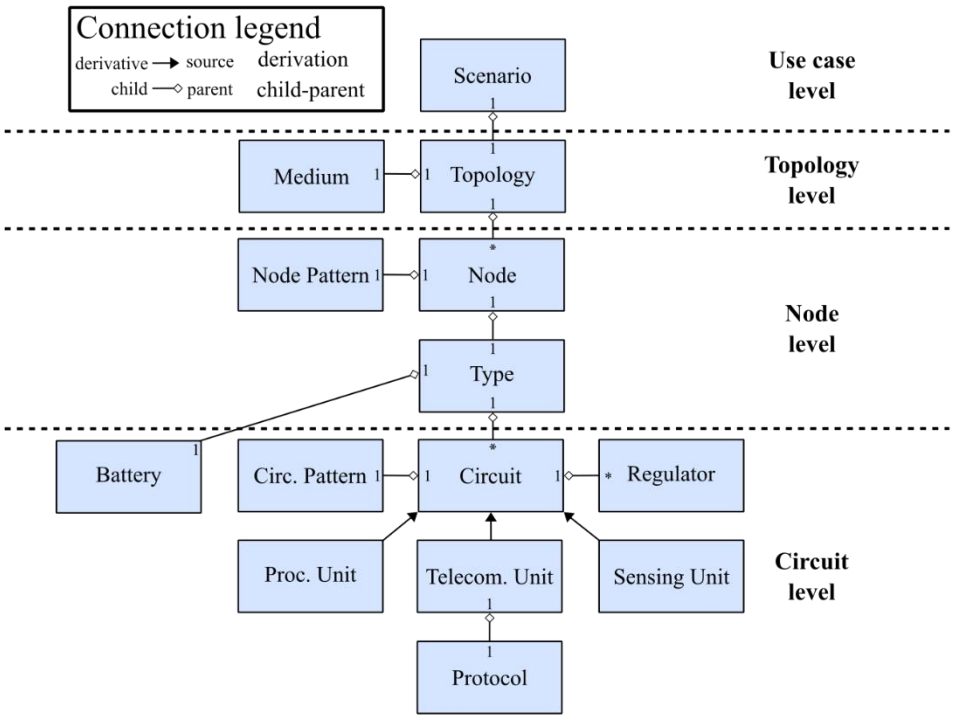



**Figure 3-10: Object diagram for the proposed model**

In the following, we provide a level-based description of each object in the model, starting from the lowest level.

### *a) Circuit level*

The circuit level is the lowest level in the proposed model. Objects residing at this level answer the hardware modelling requirements.

- Circuit object. This object is used to model any node component that consumes energy. As mentioned in Chapter 1, each node includes multiple energy consumers, precisely a Processing Unit, a Telecommunication Unit, and an optional number of Sensing Units. To simplify the structure of our model, the Circuit object is defined as a source object. Several derivative objects, representing units previously enumerated, are connected to this source because these units have different functionalities but common attributes. For example, each unit has a supply voltage parameter that is defined only one time in the circuit object. Then, the value of this parameter is to be set in objects derived from this source. Moreover, circuit pattern and regulator objects are defined as children of the circuit object. As a result, each object derived from the circuit one will inherit these two objects.
- Processing Unit object. It is an object derived from the circuit object. It is used to describe the behaviour of the node's Processing Unit based on specific CL parameters,

as illustrated in Figure 3-5, such as the wake-up time as well as the power levels for each state (sleep and active) of the circuit.

- Telecommunication Unit object. This object, also derived from the circuit one, is used to model the behaviour of the Telecommunication unit of the node. Therefore, it includes parameters obtained from the unit's datasheet such as nominal power levels, as well as information related to the protocols used, such as the lengths of upper layer headers.
- Protocol object. Each Telecommunication Unit object includes a protocol object. It is used to model the wireless link protocol behaviour adopted in this unit. This object describes the protocol specifications such as frame lengths (acknowledgment, beacon, data, etc.) and format.
- Sensing Unit object. It describes the behaviour of a sensor circuit using parameters such as the length data generated, power level and conversion time. As mentioned previously, the use of Sensing Unit objects is optional. For example, a sensing node could have one sensor or more. However, some types of node, such as relays, could be sensor-less.
- Circuit Pattern object. As illustrated above, unit's behaviour could be described combining sequential phases creating a consumption pattern. Therefore, this object is used to model unit's activities by combining a time sequence with the associated current consumption levels.
- Regulator object. The Regulator element is an electronic circuit, part of the Power Unit, that sits in-between the battery and a given circuit aiming to control its voltage level. Thus, this object plays a major role in energy modelling. In our model, each circuit must be attached at least to one regulator. In the case where supply voltage level does not need to be adapted, virtual regulator will be added and configured to reflect the fact that it does not impact the circuit (the same input and output voltages and an efficiency equal to 100%). In very specific cases, some Circuits can require two regulators.
- Battery object. Each node must have a battery object. This object, which is also a part of Power Unit, represents the node energy-storing unit. It includes information that describes battery specifications such as the nominal values (nominal voltage and capacity) derived from the datasheet.

***b) Node level***

There are three objects at the node level. At this level, the main contribution of our proposition is to support the heterogeneity requirement.

- Node object. This object is a parent object used to model each node of the WSN. Thus, it stores the node's relative position in the topology. Additionally, it has two children: type and node pattern objects that are used to describe node structure and behaviour.
- Type object. This object serves the implementation of the WSN heterogeneity feature in our model. As mentioned earlier, different types of node can exist in a same WSN topology such as sensing or relay. Thus, each of these node variants is defined by a Type object. This object is used to describe the node structure that is composed of a set of circuits depending on the WSN application and topology: a battery, Processing Unit and Telecommunication Unit, and optional number of Sensing Units. Therefore, in our model, a Type object can be common to many nodes and nodes that have different types can co-exist in the same topology.
- Node Pattern object. Each node must include one Node Pattern object that models the node behaviour. It is implemented as a log of all activities that consume energy in a given node. Thus, the Node Pattern object combines all the Circuit Pattern objects and positions them in relative phases to each other in order to represent the real time sequence from the node point of view.

***c) Topology level***

The Topology level is the third level in our model hierarchy. It is useful to provide a description of the whole WSN topology. Objects at this level tend to represent the relationships between nodes. Two objects can be found at this level, they are:

- Topology object. It is used to describe the real topology of the considered WSN. Each topology must have at least one node. The Topology object collects position information, defined in Node object, from all the nodes in order to create the network topology. As well, it is used to model the shared medium through a child called the Medium object. Additionally, Topology object is associated with a Scenario object which resides at the use case level.
- Medium object. This object is used to represent the shared wireless medium. Although this is not included in the first version of our model, the Medium object will manage issues related to time or frequency division multiplexing, that is vital for channel accessing in environment where resources are shared. Moreover, signal propagation

models and routing algorithms will be associated with this object or implemented separately at this level.

***d) Use case level***

This level, the highest level of the model, includes only the Scenario object. This object describes a simulation scenario, related to a given WSN application, using parameters such as the simulation time and duration type which can be seconds, hours, days or years to provide the user with a way to set the level of accuracy to correspond to the selected time unit. The model can include several scenarios at the same time for the sake of comparison.

### 3.2.4.2. Behavioural model description

As explained in the structural description of our model, depending on its type, every node in the topology has a Processing Unit, a Telecommunication Unit and one or more Sensing Units. Moreover, it has been demonstrated in Section 3.2.3 that the behaviour of each unit, which composed a node, could be described using a succession of phases called a pattern. Therefore, this section is dedicated to the description of how the cross-level modelling approach is implemented for each unit in our model.

***a) Processing Unit***

As presented previously, the Processing Unit time sequence consists of 3 phases: sleep, wake-up and active. The unit starts in a sleep state and stays in this state for a specific time. The duration of this first phase is set based on the scenario setting, as illustrated by Figure 3-11. Then, the Processing Unit wakes up and enters the active state allowing other units' activities to take place. Finally, it goes for sleep until the end of the sequence.

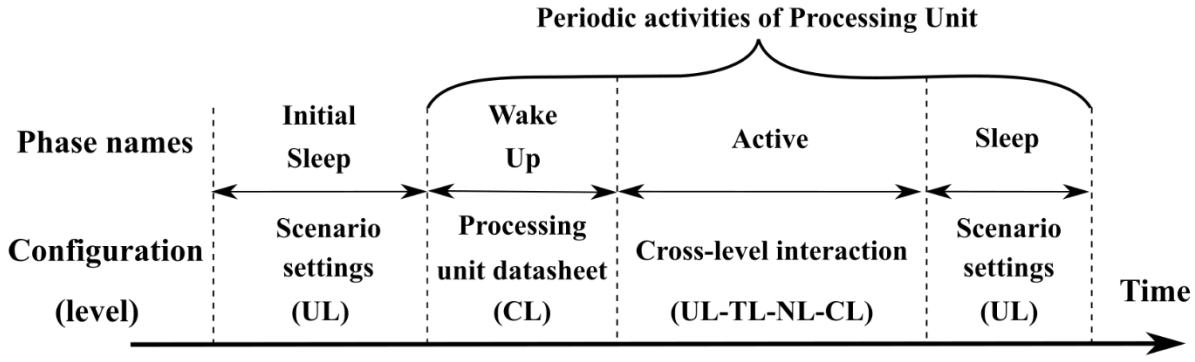



**Figure 3-11: Typical time sequence for the Processing Unit in the proposed model**

Based on the scenario that is being implemented, the Circuit pattern object associated with the Processing Unit stores a sequence of time values and their corresponding power levels. The time values for each state are defined as follow:

- Initial sleep ($t_{CPU\_islp}$): a user case level parameter, it can be 0.
- Wake up ($t_{CPU\_wk}$): the time spent in this state is a Circuit Level (CL) parameter. The value is derived from the circuit datasheet. For the Processing Unit, this phase is usually transitory and thus, it is very short.
- Active ($t_{CPU\_act}$): the length of the time spent in this state is not a constant parameter, but it depends on the activities of other node units (Eq. (3-1)). The active state starts when the wake-up state ends, and it spans, through cross-level interactions, to give enough time for the activities of the node's components to take place. For example, if the sensing unit is going to measure two times separated by a small interval, the processing unit active duration will automatically extend to cover the activities time of the Sensing Unit. The Processing Unit leaves the active state when there are no more activities to perform in the implemented scenario.

$$t_{CPU\_act} = \sum_{i} t_{act_i} \tag{3-1}$$

  Where $i$ is the activities index for the other circuits considering that if two or more activities intersect, the corresponding interval will be considered once.

- Sleep ($t_{CPU\_slp}$): considering the pattern frequency $F_p$, the time spent in this state is calculated using Equation (3-2).

$$t_{CPU\_slp} = {}^{1}/_{F_p} - (t_{CPU\_act} + t_{CPU\_wk}) \tag{3-2}$$

The current consumption levels corresponding at each phase is derived from the Processing Unit datasheet.

***b) Telecommunication Unit***

As shown earlier in Figure 3-7, each node includes a Telecommunication Unit which implement a full network protocol stack (Figure 3-12). In this regard, modelling the Telecommunication Unit is not limited to specific set of protocols, different combinations can be selected as a part of the scenario settings.

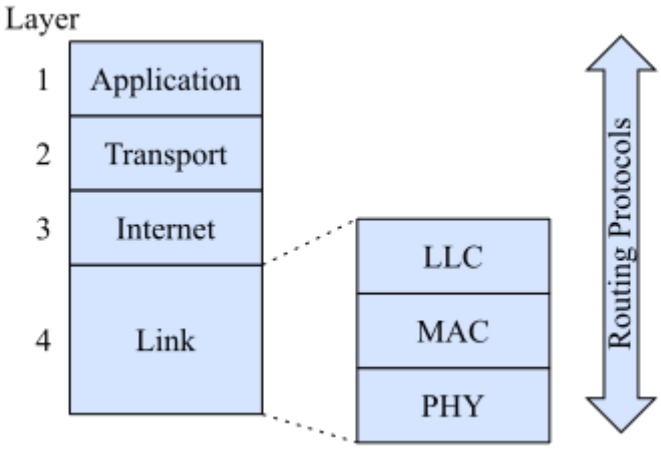



**Figure 3-12: Proposed protocol stack for the Telecommunication Unit**

The length of the payload, in bytes, is called $B_{DATA}$. Based on the selected protocol, Layer 1 adds its header ($B_{L1H}$ bytes) to the payload, creating its Protocol Data Unit (PDU) $B_{L1PDU}$ (Eq. (3-3)). Then, Layer 1 passes its PDU to the lower layer, Layer 2, where the same operation is repeated and the corresponding header ($B_{L2H}$ bytes) is added, creating L2PDU ($B_{L2PDU}$ bytes) (Eq. (3-4)). After that, L2PDU is passed to the Layer 3 that follows the same operation passing L3PDU ($B_{L3PDU}$ bytes) to the next layer (Eq. (3-5)).

As illustrated by Figure 3-12, Layer 4 has several sublayers: LLC, MAC and PHY. The LLC operations has no impact on the L4PDU length. The MAC sublayer adds not only a header ($B_{L4H}$ bytes) but also a trailer ($B_{L4T}$ bytes) as well, the length of the header and trailer are specified by the wireless link protocol selected from the scenario setting (Eq. (3-6)). The PHY sublayer, specified by the wireless link protocol, does not add a header to the data frame, but it rather focuses on how to access the radio channel and in which bit rate the data is to be sent.

$$B_{L1PDU} = B_{DATA} + B_{L1H} \quad (3\text{-}3)$$

$$B_{L2PDU} = B_{L1PDU} + B_{L2H} \quad (3\text{-}4)$$

$$B_{L3PDU} = B_{L2PDU} + B_{L3H} \quad (3\text{-}5)$$

$$B_{L4PDU} = B_{L4H} + B_{L3PDU} + B_{L4T} \quad (3\text{-}6)$$

Following the previous equations, two specific cases might appear for the considered Layers:

1) If there is no protocol working in one specific layer, then the number of the bytes added at that layer is set to 0.

2) The case where there are more than a single protocol operating in the layer. For example, the IP is an internetwork protocol, it can be combined with a routing protocol or an auxiliary protocol such as Internet Control Message Protocol (ICMP). Thus, if $N_p$ protocols are working at a specific layer, Layer $X$ (with $1 \leq X \leq 4$), the length of the header is calculated as a sum of the protocols headers ($B_{LXHi}$) using Equation (3-7).

$$B_{LXH} = \sum_{i=1}^{N_p} B_{LXH_i} \tag{3-7}$$

Furthermore, the preamble, used at the PHY sublayer, is a mechanism implemented before sending any data frame. Using the preamble means sending a set of predefined bits. The length of the preamble is $b_{PRE}$ bits. It is specified by the wireless link protocol, and it is to be sent before the data frame. Therefore, the total number of bits sent over the radio channel when sending a data frame ($b_{TOTDATA}$) is given using Equation (3-8).

$$b_{TOTDATA} = b_{PRE} + B_{L4PDU} * 8 \tag{3-8}$$

The time needed to send this number of bits ($t_{DATA}$) is calculated based on the bit rate ($BR$) which is specified by the wireless link protocol. Equation (3-9) shows how $t_{DATA}$ is calculated in our model. $t_{DATA}$ is expressed in seconds and $BR$ unit is bit per second.

$$t_{DATA} = \frac{b_{TOTDATA}}{BR} \tag{3-9}$$

Several bit rates ($BR_1, BR_1$ ... $BR_{N_{BR}}$) might be used to send the total payload. In this case, Equation (3-10) is used, where ($b_{TOTDATA_i}$) is the amount of data sent using the bit rate $BR_i$.

$$t_{DATA} = \sum_{i=1}^{N_{BR}} \frac{b_{TOTDATA_i}}{BR_i} \tag{3-10}$$

If the amount of data to be sent does not fit into a single frame, multiple frames can be created using fragmentation. In this case, as illustrated by Figure 3-13, there is a need for sending data frames sequentially and a guarding period to separate the sequential frames is used. It is called Short InterFrame Spacing (SIFS). This time is obtained from the protocol specifications.

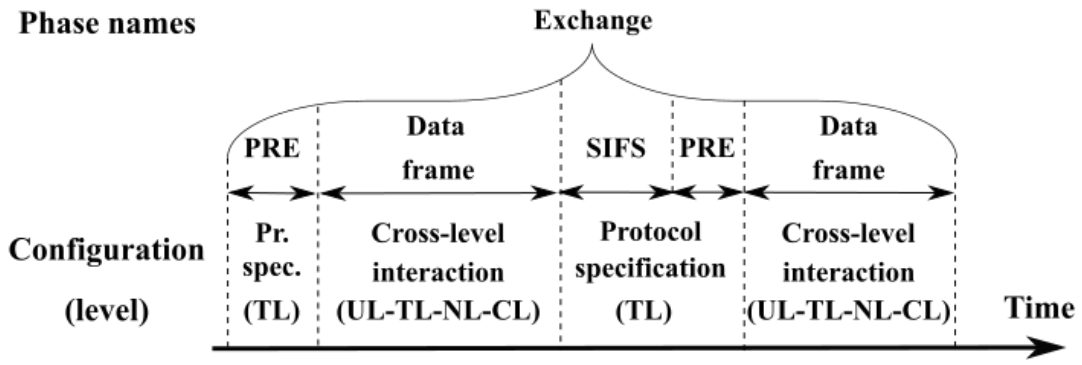



**Figure 3-13: Time sequence for the transmission without acknowledgment**

The acknowledgment process can also be included in the modelling of the Telecommunication Unit. If it is enabled, the unit that received the data will wait for a specific period ($t_{w\_ack}$), then, it will send an acknowledgment frame to the data source. The length of the acknowledgment frame ($B_{ACK}$) can be obtained from the protocol specification and it is usually given in bytes. The time needed to send or receive an acknowledge frame ($B_{ack}$) is calculated using Equation (3-11).

$$t_{ack} = \frac{B_{ack} * 8}{BR} \tag{3-11}$$

As shown in Figure 3-14, in the acknowledged transmission, SIFS separate the acknowledgment frame period from the next data frame period.

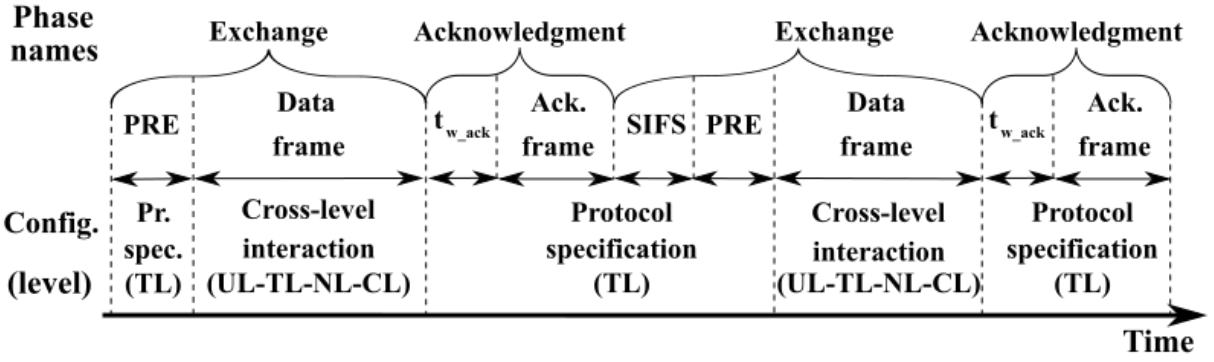



**Figure 3-14: Time sequence for the transmission of two sequential frames using acknowledge**

When nodes are competing for the radio resources, the channel access phase is necessary to avoid collisions within the wireless medium, and this phase can also be included in our model, as presented in Figure 3-15. However, there is no specific algorithm followed by all the wireless link protocols. Some, such as 802.11 family, uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) algorithm [IEE16a]. Others, like 802.15.4, use the Channel Clear Assessment (CCA) algorithm [IEE16c]. Thus, there will be no strict definition for the access

phase. We assume, in our model, that it occupies a time period, and details related to the algorithm used must be configured. In the Figure 3-15, only one exchange phase is considered as well as a sleep phase that is attached at the end of the sequence.

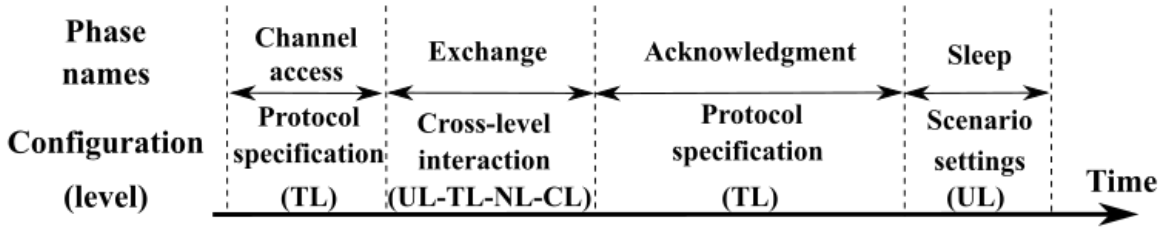



**Figure 3-15: Simple time sequence for a Telecommunication Unit**

In this regard, the Telecommunication Unit time sequence consists of the following phases: sleep, access, exchange, and acknowledgment. Each of these phases includes a set of activities, and each is associated with a current consumption level derived from the Telecommunication Unit datasheet. At the beginning of the time sequence, the Telecommunication Unit is in the sleep state by default, after that, when the Processing Unit pass into the active state, the Telecommunication Unit enters automatically the idle state.

***c) Sensing Unit***

The Sensing Unit time sequence consists of two phases: sleep and active. In the simplest case, the unit starts in the sleep phase, and it remains at this state for a duration determined in the scenario setting. Then, the unit become active in order to measure a specific physical phenomenon, resulting in its conversion into an electrical signal. The time needed for one measurement is obtained from the Sensing Unit datasheet. After that, this unit return to its sleep state and remain at this state until the next needing measurement (Figure 3-16).

In general, the time sequence can be more complex as the Sensing Unit can concatenate more than one measurement separated by waiting times. The number of the measurements and the length of the separating period are specified from the scenario settings. Each activity is associated with a current consumption level. The level values are derived from the Sensing Unit datasheet.

In the case presented in Figure 3-16, the sensor makes two measurements separated by the spacing period, the measurement periods are referred to as active activities.

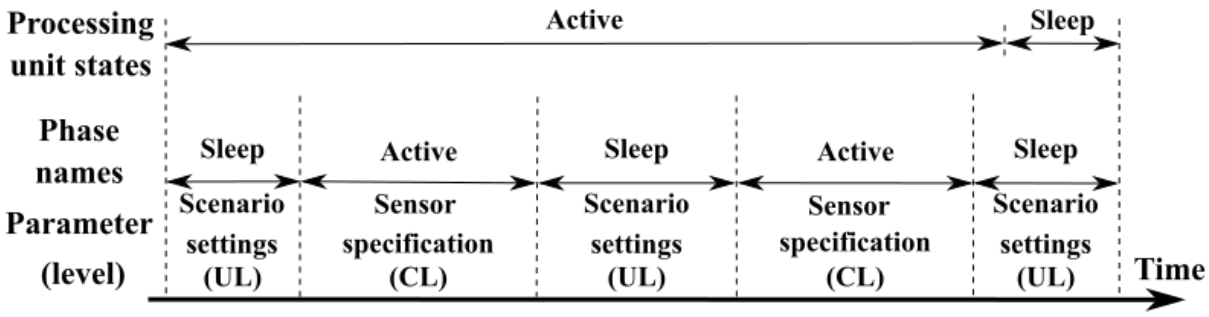



**Figure 3-16: Time sequence for Sensing Unit with two measurement activities**

## 3.3. Energy-awareness modelling

In this section, the energy awareness aspect of the proposed model is introduced. First, the energy conversion is covered. Secondly, a detail description is provided to show how the node lifetime can be estimated using our model.

### 3.3.1. Regulator modelling

As for all electronic systems, wireless sensor nodes need to be supplied with electrical energy. This energy can be obtained from a source that provide alternating (from electrical grid) or direct (from battery) currents. Considering that, in WSN, when the application requires autonomy and mobility of the nodes, many solutions can be implemented. Integrating an on-board battery is the simplest one and already the widely used solution. In this thesis, this is the only power option that is considered.

Based on that, in our model, all nodes are equipped with a battery. However, node circuits have very specific needs in terms of supplying voltage. It explains why a conversion step is generally required to adapt the available power source, here providing by a battery, to these circuits. Because batteries are DC power source which provides a voltage that varies depending the state of charge, that is why a DC/DC conversion is enough to satisfy the circuit requirements previously mentioned.

Devices that balance the variation of input voltage to provide a constant level for supplying node circuits are called voltage regulators. As illustrated by Figure 3-17, on one hand, a voltage regulator receives, on its input, the voltage $V_{bat}$ provided by the battery and supply the circuit with the voltage $V_{cir}$. On the other hand, the circuit current $I_{cir}$ is drawn based on the circuit and its current state (sleep, active…). For each circuit, the consumption level for each possible state is derived from the circuit datasheet. Therefore, the current $I_{cir}$ depends on the circuit time sequence that is defined in the Circuit pattern object, based on the scenario settings and cross-

level and intra-level interactions existing between parameters. The relationship between battery current $I_{bat}$ and $I_{cir}$ is determined by the regulator type.

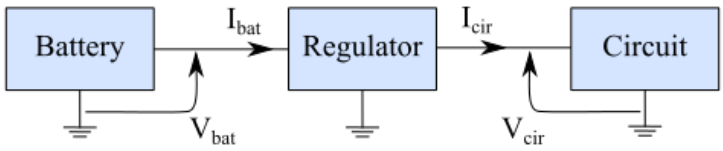



**Figure 3-17: Schematic illustration of the regulator connections**

Therefore, to accurately calculate the lifetime of the system, the regulator efficiency $\eta$ should be considered. It is calculated using the Equation (3-12):

$$\eta = \frac{P_o}{P_i} \, x \, 100 \tag{3-12}$$

where $P_o$ and $P_i$ are the regulator's input and output power respectively.

Based on that, two categories of voltage regulators can be distinguished:

1) Linear regulator. This kind of regulator supports the differences between input and output voltages ($V_{bat}$ - $V_{cir}$) and the input current of the regulator is equal to the output current, as shown in Equation (3-13).

$$I_{bat} = I_{cir} \tag{3-13}$$

The weak efficiency is a drawback in this family of regulators, especially if the ($V_{bat}$ - $V_{cir}$) has a significant value. Based on the Equations (3-12) and (3-13) the efficiency of this type of regulator can be calculated using Equation (3-14).

$$\eta = \frac{V_{cir}}{V_{bat}} \, x \, 100 \tag{3-14}$$

2) Chopper regulator is a device based on switching semiconductors. The switching technology allows to enhance up the efficiency of the regulator to a value close to 100%. Chopper regulators can be step-up or step-down. Based on that, they are positioned in parallel or in series respectively with the target circuit. The input current $I_{bat}$ of the regulator must be calculated based on Equation (3-15), and the relations between the power, voltage and current ($P = VI$).

$$I_{bat} = \frac{V_{cir} * I_{cir}}{V_{bat} * \eta} \quad (3\text{-}15)$$

In our model, we consider the two cases by implementing Equation (3-16):

$$I_{bat} = k_1 \frac{V_{cir} * I_{cir}}{V_{bat} * \eta} + (1 - k_1) I_{cir} \quad (3\text{-}16)$$

where $k_1$ is a coefficient that is set to "0" if the regulator is linear, and set to "1" if it is chopper regulator.

### 3.3.2. Node lifetime modelling

The suggested model considers a node to include a Processing Unit, a Telecommunication Unit, and a number of Sensing Units that can be set to zero, if needed. In addition to that, each node has an energy source: a battery which supply all circuits of the node.

As illustrated in Figure 3-17, the voltage regulator is the interface between the battery and a circuit, and it provides a specific voltage level. Based on circuit behaviour, there are several configurations for regulators in the real implementation. The three configurations presented in Figure 3-18, are the following:

A. Circuit with an internal regulator. This is a special case, where the circuit includes a built-in regulator, as in the transceiver CC2420, and there is no need for an external one. If it is not mentioned in the circuit datasheet, it is not possible to know what type of regulator is used. In this case, the model assumes that it is a linear regulator. As a result, the current drawn from the battery is equal to the current that feeds the circuit ($I_{bat} = I_{cir}$).
B. Circuit associated with one external regulator. In this connection, one regulator supplies the circuit. Thus, only one current is being drawn from the battery. Because the regulator could be chopper or linear, the equation that governs the relationship between $I_{bat}$ and $I_{cir}$ can be (3-13) or (3-15) respectively. MSP430 is an example of a Processing Unit that is fed with energy using this type of connection.
C. Circuit associated with two external regulators. In case the circuit requires two different levels of voltage supply, two regulators are necessary to serve this purpose. Consequently, two currents are drawn from the battery, one for each regulator. HDG204 is a Telecommunication Unit that require to be supplied by two different voltages (3.3 and 1.2 V), thus, two regulators are needed [HDG11].

The proposed model implements the three types of connection between the battery and the circuit. Note that the use of the three types together is not mandatory, and the node shown in Figure 3-18 adopts this structure only for illustration purposes. Additionally, this figure shows that current drawn from the battery can be calculated using Equation (3-17):

$$I_{bat} = \sum_{i=0}^{M} I_{bat_i} \tag{3-17}$$

where *M* is the number of regulators onboard the nodes.

The battery is a mean to store electrical energy, but its output voltage varies in a function of the state of charge that is related to the drawn current. Thus, it is difficult to compute the stored energy at a given moment. To simplify the model, we consider the battery voltage $V_{bat}$ to be constant. Based on that, the energy consumed by one pattern of the circuit *k* can be calculated using Equation (3-18):

$$E_{cir_k} = V_{bat} \sum_{j=0}^{N} \left[ \left( \sum_{i=0}^{Z} I_{bat_{ijk}} \right) * T_{seq_{jk}} \right] \tag{3-18}$$

where *i* and *j* are the indexes of the regulator and the phase (obtained from the pattern of corresponding circuit) respectively, *Z* is the number of regulators associated with the circuit *k*, *N* is the number of phases, $T_{seq_{jk}}$ is the duration of the phase *j* associated with the circuit *k*, $I_{bat_{ijk}}$ is the current drawn from the battery by the regulator *i*, at the moment specified by the phase *j* in the time sequence of the circuit *k*.

Finally, the total energy consumed in one node pattern $E_{pat}$ is calculated using the Equation (3-21):

$$E_{pat} = \sum_{i=0}^{X} E_{cir_i} \tag{3-19}$$

Where *X* is the number of the circuits onboard the node.

Equation (3-18) concerns all the phases except of the first one that might be different because of the initial sleep phase. In this case, the structure of the first pattern will be different from the other patterns, and thus, the energy consumed in it is different. Anyway, the concerning energy is very low, and that is why it is negligible.

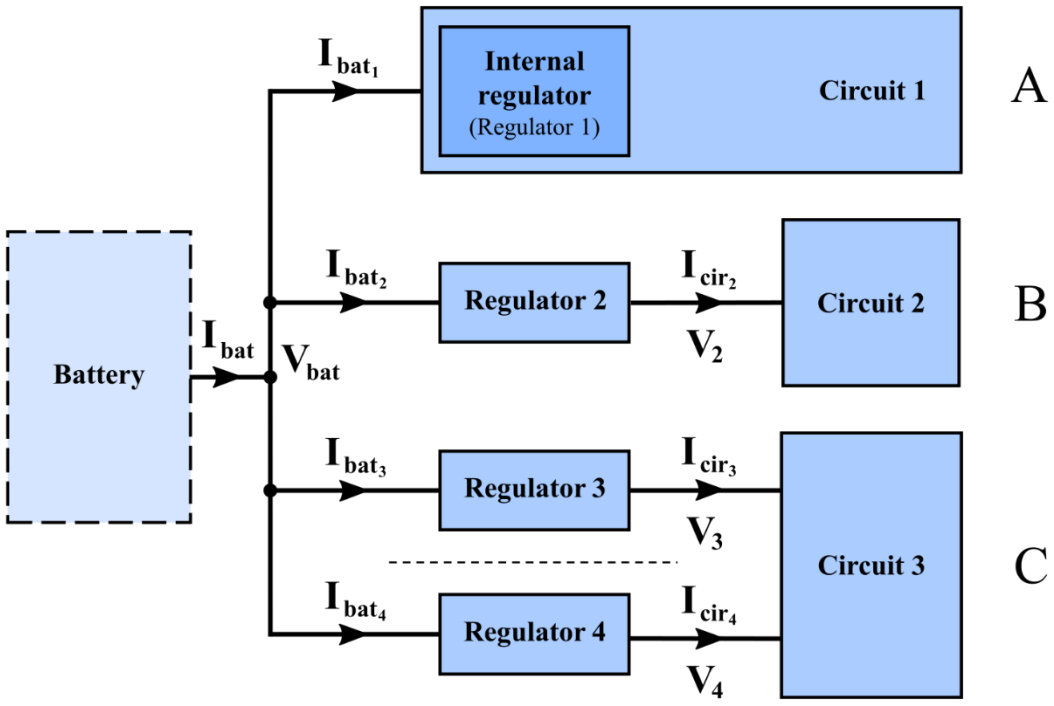



**Figure 3-18: Schematic model for the node with different types of regulators**

To estimate its lifetime, a node is considered dead when it has no more energy stored in its battery, corresponding to $E_{bat}(t) = 0$. In our model, the energy consumed can be expressed as a multiple of the node pattern energy. A mathematical form of this relationship is shown in Equation (3-20)

$$E_{bat}(t) = E_{bat}(kT_{pat}) = E_{bat}[k] = E_{bat}(0) - kE_{pat} \quad (3\text{-}20)$$

Where $E_{bat}(0)$ is the initial amount of energy stored in the battery and $k$ is the number of occurrences of the node pattern. Considering that the battery become empty when $k = k_{pat}$, the number of occurrences is calculated using Equation (3-21).

$$k_{pat} = \frac{E_{bat}(0)}{E_{pat}} \quad (3\text{-}21)$$

Thus, the estimated lifetime of the node, called $T_{nodelife}$, can be calculated using the pattern duration derived from its frequency as show in Equation (3-22):

$$T_{nodelife} = \frac{k_{pat}}{F_p} = k_{pat}T_{pat} \quad (3\text{-}22)$$

where $F_p$ is the frequency of the pattern and $T_{pat}$ is the pattern duration.

## 3.4. Conclusion

This chapter was dedicated to explain the modelling process of the suggested WSN model. First, the adopted modelling approach, based on a cross-level technique, was presented. After that, the energy-awareness aspect of the model was exposed. As a result, the energy consumed by each circuit onboard the node, as well as the prediction of the system lifetime are estimated.

In the cross-level approach, the studied system is divided into a set of levels of abstraction (circuit, node, topology and use case), each of which includes several objects where parameters are located. The parameters interact with each other, and if the parameters are located in different levels, the interactions will cross the level boundary (cross-level interactions). Otherwise, the interactions will occur within the level, and in this case, we speak about intra-level interactions.

The circuit and node patterns are concepts introduced throughout the modelling process. The pattern is a time sequence associated with a sequence of current consumption levels. The circuit pattern describes the activities of a node's component in the circuit level from energetic perspective. The node pattern is composed by the aggregation of the corresponding circuit patterns. In the proposed model, it is assumed that the activities take place periodically, and thus, to simplify the modelling process, the node pattern will be repeated until the end of the node lifetime or the end of the simulation.

As it will be shown in the next chapter, the cross-level approach allows effective, accurate and energy-aware modelling for energy storage and consumption.

In terms of energy, the WSN nodes are usually autonomous, and that imply the use of onboard storage units such as batteries. Although the battery object resides in the circuit level, the energy consumption imposed by the node pattern is the result of cross-level interactions. In practice, usually the battery is not directly compatible with circuit requirements, and the voltage levels need to be adapted. In the proposed model, we have considered linear regulators and DC/DC switching converters because they have different characteristics from the energetic point of view, and they are the most commonly used in WSN applications.

In the next chapter, we will show how proposed model was implemented, and different scenarios will be introduced and tested. In order to test the precision of the model, the obtained results will be compared with results from a well-known simulator (NS2) as well as results measured on real WSN nodes.

# Chapter 4. MODEL VALIDATION WITH SIMULATORS AND REAL NODES

4.1. Introduction ..... 97
4.2. Simulator description and capabilities ..... 98
4.2.1. General overview ..... 98
4.2.2. Model implementation ..... 99
4.2.3. Scenario creation ..... 104
4.2.4. Simulation and analysis stages ..... 107
4.2.5. Other functionalities of the simulator ..... 111
.4.3 Comparison with NS2 ..... 116
4.3.1. General settings ..... 117
4.3.2. Simulations using Telecommunication Unit only ..... 118
4.4. Validation with real nodes ..... 122
4.4.1. Description of the circuits ..... 122
4.4.2. Results ..... 126
4.5. Conclusion ..... 129

## 4.1. Introduction

Modelling is a method that is used to mathematically describe the behaviour of a large spectrum of entities that might range from technical and living systems. Different models can be created to cover aspects of the studied system. In Chapter 3, we provided a detailed cross-level energy-aware model for WSN. However, this model needs to be validated and a simulator can provide a framework to test and verify it.

As a general concept, the implementation of the model in a simulator allows to study the temporal evolution of the system. The simulation can be used, for examples, to enhance system performance or to detect design errors before the implementation. In practice, a model can never be a perfect representation of a real system, some assumptions are always necessary. Although these assumptions help to reduce system complexity in order to have realistic duration of simulation, the cost is to be paid in the accuracy of the obtained results. This is the reason why comparing the proposed simulator with other simulators and validating it with a real implementation of the studied system are two important steps of our work.

In the first step, the results obtained from our simulator will be compared with those taken from NS2, a simulator described in Chapter 2, when the same scenario is implemented. The goal of this comparison is to determine the differences between the obtained results and try to explain them. However, comparing two simulators is not enough for the validation of our proposal because the differences depend necessarily on the model implemented in NS2. Those implementations might have their own errors when compared to the original system. That is why, in the second step, the validating phase, where results will be obtained from physical implementation of WSN, will be used. Thus, the results obtained with our simulator can be experimentally validated. In physical implementations, measurements of the physical phenomena are needed. However, in simulations, the results are provided by a user interface attached to the software after the execution of the implemented models. The output can be in textual or graphic form. In practice, some unpredictable phenomena, such as electromagnetic noise, will affect the results, whereas these factors are generally excluded from the simulation via a set of assumptions adopted during the model development.

In this chapter, a WSN simulator, based on the model suggested in Chapter 3, is presented. This simulator is developed using MATLAB. Based on WSN scenarios, this simulator reproduces the network behaviour at different levels: topology, node, and circuits. Chapter 4 starts by showing how to implement the model proposed in Chapter 3 to create a cross-level energy-

aware simulator. After that, a set of scenarios are created to compare the simulator with NS2 and to validate the obtained results, and the proposed approach, with real nodes.

## 4.2. Simulator description and capabilities

This section is dedicated to describing the implementation of a WSN simulator based on the cross-level energy-aware model proposed in Chapter 3. We start presenting the structure of the simulator. Then, an application example is provided to show how to create a scenario and trace results.

### 4.2.1. General overview

MATLAB was the platform used to create our simulator because it supports an object-oriented environment and allows to integrate easily graphical interfaces. Moreover, MATLAB is a generic platform that is widely used, in research and development domain, to model and simulate complex systems such as WSN [PAP16].

As illustrated by Figure 4-1, the proposed simulator supports a process consists of three stages:

- Configuration stage. Scenarios can be created, modified and stored in a repository as XML files. In this version of the simulator, the scenarios can only be configured through a command-line interface. This textual interface accepts script commands, that we specifically created, to configure the whole scenario.
- Simulation stage. It is achieved by the simulator core that execute commands taking into account XML configuration files. The obtained results are sent to the graphical interface for further analysis.
- Analysis stage. At the end of the simulation, it allows the visualisation, using the built-in graphical interface, of the obtained results based on the user choices. The output is displayed according with the cross-level design of our model. Thus, results are categorised based on the level they belong to. For example, energy consumption for a complete node can be viewed as a node-level result, while energy consumption for each phase for a given circuit is a circuit-level result. Additionally, several simulations can be run one after another for the sake of comparison.

XML files are created for each component and linked together using the hierarchical structure of our model previously presented in Figure 3-10. These links provide a means for the cross-level interactions to be implemented. For example, the XML file that represent the node Type object, which reside at the node level, has cross-level links to the XML files that represents the

Circuit objects that are within the Circuit level. These links allow the Node type object to integrate the circuit objects parameters in its own structure.

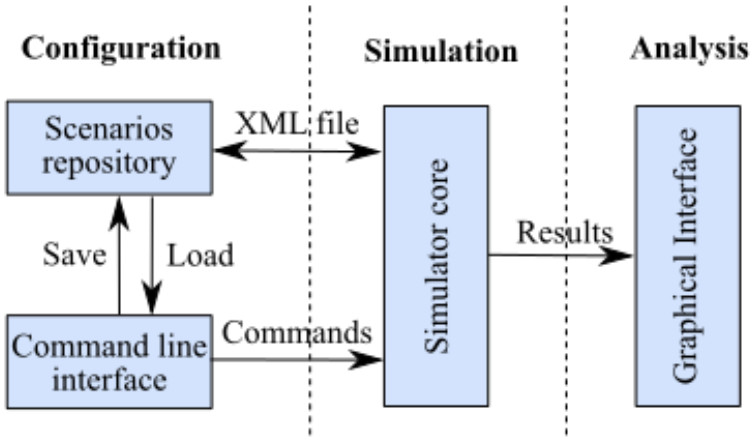



**Figure 4-1: Three stages process of the proposed simulator**

## 4.2.2. Model implementation

The simulation can be built based on two approaches, as illustrated in Figure 4-2: bottom-to-top and top-to-bottom. In the bottom-to-top approach, the circuits are created firstly, then nodes are assembled and put together to create a topology. Finally, the topology is to be attached to a scenario. On the contrary, the construction order is inversed in the top-to-bottom approach.

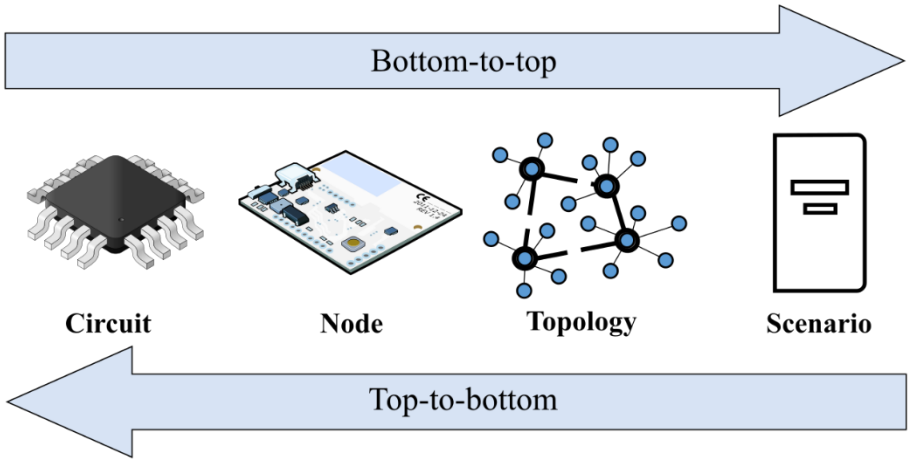



**Figure 4-2: Scenario construction approaches**

For the sake of explaining how to construct a simulation, the bottom-to-top approach will be adopted in the next section (4.2.4).

The XML trees will be presented to clarify the structure of the implemented model, providing a way to clear up how the interconnections are established inside and across the levels. For better understanding the model structure, the XML trees will be presented following top-to-bottom approach starting from the scenario until the auxiliaries attached to the Circuit objects.

Figure 4-3 represents the XML tree of the scenario object which resides at the Use case Level (UL) of the model structure. This object includes two parameters: the simulation time and its scale, as well as a link to the topology XML file.

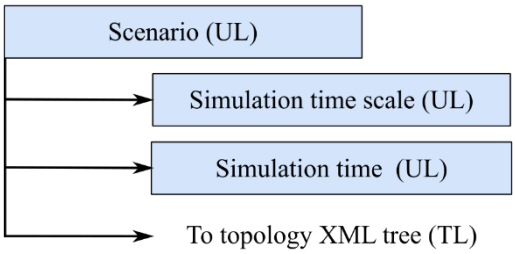



**Figure 4-3: XML tree of parameters for the Scenario object**

The Topology XML tree, shown in Figure 4-4, includes medium characteristics and the size of area considered in the simulation. The specifications of the nodes included in the topology are read form the Nodes XML trees which are located in the Node Level (NL).

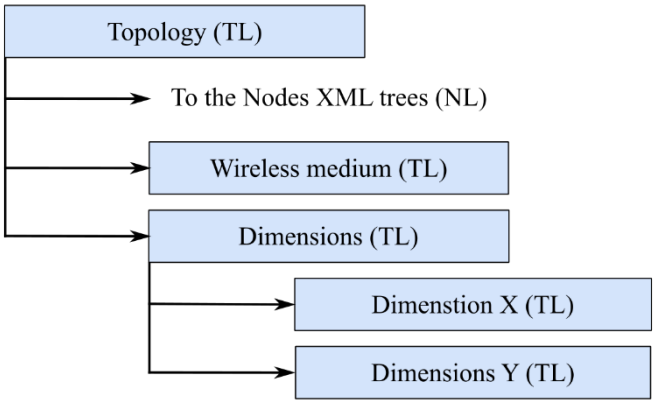



**Figure 4-4: XML tree of parameters for the Topology object**

Figure 4-5 displays the XML tree of a Node object. It includes parameters for the initial status that describe the state of the node at the start of the simulation, and the node position which is given in Cartesian coordinates where the origin (0,0) is located at the left down corner of the simulation area. Each node has its own XML file, and each file includes a link to a Node type XML file that is used to describe the components of the node.

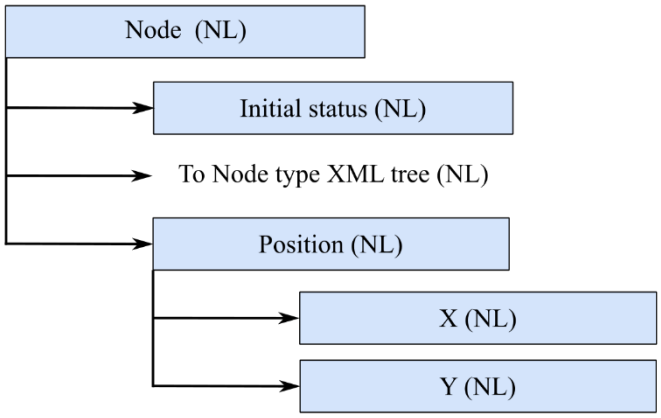



**Figure 4-5: XML tree of parameters for the Node object**

The Node Type XML tree, presented in Figure 4-6, has only links to other XML files located at the Circuit Level (CL) of our model. A node type object has a link to a Processing unit, Telecommunication Unit, battery, and an optional number of Sensing Units. A Node Type file could be used by a set of nodes that have identical characteristics. In order to answer the heterogeneity requirement, several Node types could exist in the same scenario, providing a way to simulate different types of nodes at the same time.

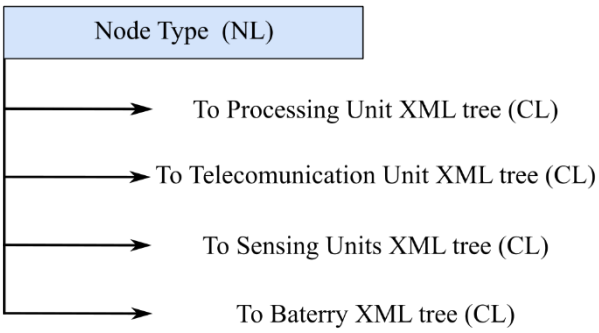



**Figure 4-6: XML tree of parameters for the Node Type object**

As illustrated in Figure 3-10, all these units are derived from an abstract object, called Circuit object, allowing polymorphism. In this technique, the units share two attributes from a common source (consumption levels and supplying voltage), but at the same time, they have their own attributes that enable creating distinguishable variants.

The Processing Unit object is derived from the Circuit object using polymorphism, its XML tree is shown in Figure 4-7. Several parameters are inherited from the source object: the supply voltage and the consumption levels as well as a set of parameters that belongs only to the Processing unit: operating frequency, modes and wakeup duration.

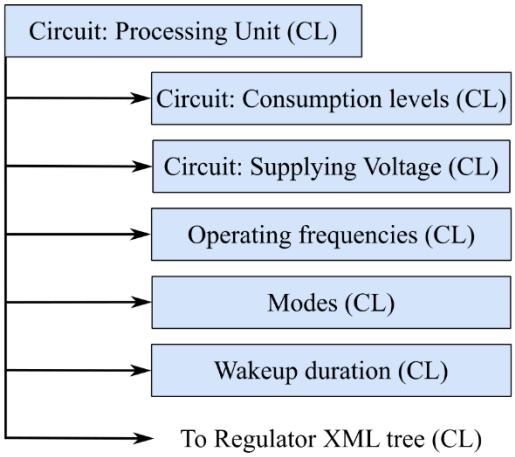



**Figure 4-7: XML tree of parameters for the Processing Unit object**

In the proposed implementation, the consumption levels are a 3-values vector that includes the current consumptions values for sleep, idle and active phases respectively. Those levels depend on other parameters such as supplying voltage, operating frequency and oscillator mode (High-Speed Crystal with or without Phase-Locked Loop - PLL, external Resistor/Capacitor, internal oscillator, etc.). In the case where several combinations of operating frequencies, modes or supplying voltages need to be simulated, an XML file will be created for each one.

The regulator object is attached to the Circuit object, this means that all the objects derived from the circuit will have one or two regulator children. As a result, when the model is implemented, every circuit XML file will have one or two links to regulator XML files.

Each Regulator XML tree, as shown in Figure 4-8, includes two parameters: the type that can be linear or chopper, and efficiency given as a percentage rate for the chopped regulator (value set to -1 if the regulator is linear).

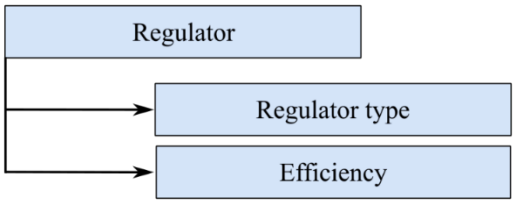



**Figure 4-8: XML tree of parameters for the Regulator object**

The Telecommunication Unit object is derived from the Circuit object, which means that it inherited the common parameters from this source object. These parameters, consumptions levels and supplying voltage, are to be configured using the data obtained from unit datasheet. Additionally, because it is a circuit, the Telecommunication Unit has a child regulator object.

Finally, the XML tree, provided in Figure 4-9, includes parameters related to the protocol stack as well as the payload lengths that are used to build the PDU based on Equations (3-3) to (3-6).

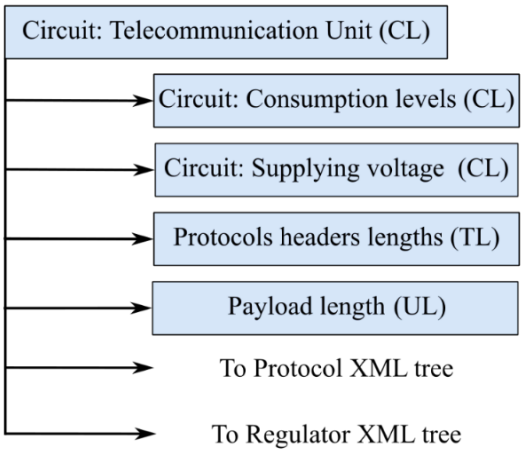



**Figure 4-9: XML tree of parameters for the Telecommunication Unit object**

The Telecommunication Unit object has a second child object: the protocol. As illustrated in Figure 4-10, it includes parameters related to the link layer protocol such as the length of the preamble and the acknowledgment frame as well as the times corresponding to the SIFS and channel access. Protocol parameters are shared among the nodes and it explains why they are considered as Topology Level parameters.

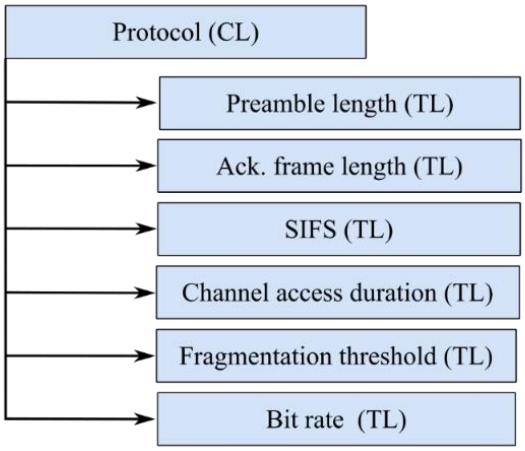



**Figure 4-10: XML tree of parameters for the Protocol object**

The last circuit to introduce is the Sensing Unit. Each node has an optional number of sensors (it can be configured 0). The XML tree for a Sensing Unit, presented in Figure 4-11, includes two parameters that describe the behaviour of the corresponding object: the payload length and the conversion duration. The first parameter shows the number of bits created by one measurement of the sensor, while the second is used for the time needed to do that.

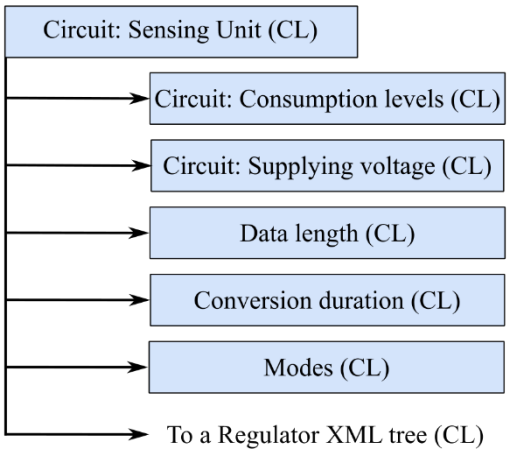



**Figure 4-11: XML tree of parameters for the Sensing Unit object**

Each node is equipped with one battery that characterises the storage of the energy. Figure 4-12 shows the XML tree for the Battery object where three parameters are included: the output voltage of the battery considered as a constant value in this implementation, the state of charge and the nominal capacity.

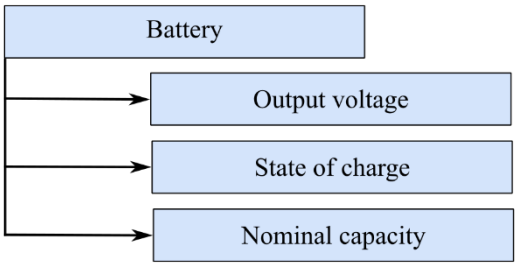



**Figure 4-12: XML tree of parameters for the Battery object**

### 4.2.3. Scenario creation

Once the model is implemented, scenarios can be created and then executed. In this section, an example showing how to create a scenario, following the bottom-to-top construction approach, will be presented. This scenario is used to illustrate the simulator capabilities. The scenario will be changed in the next sections for the comparison with NS2 and the experimental validation.

The first step of scenario creation is the configuration of the circuits, representing the different units of a node. Each one is created separately. To configure the parameters of each circuit, the details are obtained from the corresponding datasheet. After that, an object is created for each circuit and stored as an XML file in the repository.

In the scenario considered in this example, each node has the same structure. Thus, WSN nodes are homogeneous and are represented using only one Type object. Each one includes the following circuits with the corresponding electrical characteristics presented in Table 4-1:

- Processing Unit is the microcontroller PIC18F4620 based on the nanoWatt technology [PIC18];
- Telecommunication Unit is a CC2420 circuit. It supports 802.15.4 / ZigBee-ready protocol stack. It is designed for low power and low voltage wireless applications [TEX19]. The sending power, -25dB in the example, is automatically chosen considering the cross-level interactions, mainly the effect of the distance;
- Sensing Unit is the TMP102, a low-power digital temperature sensor [TEX07];
- Power Unit is a 2000 mAh/3.7 V Polymer Lithium-Ion battery [BAT37].

For all the circuits, a linear regulator is considered.

**Table 4-1: Electrical characteristics for the selected circuits (Appendix 2)**

| **Circuit name** | PIC18F4620 | | CC2420 | | TMP102 | |
|---|---|---|---|---|---|---|
| **Circuit Type** | Processing Unit | | Telecommunication Unit | | Sensing Unit | |
| **Supplying voltage [V]** | 5 | | 3.6 | | 3.3 | |
| **Current level [µA]** | **Sleep** | 0.2 | **Sleep** | 20 | **Sleep** | 0.5 |
| | **Wakeup** | 430 | **Sending (-25 dB)** | 8500 | **Active (400kHz)** | 15 |
| | **Active** | 1300 | **Receiving** | 18800 | | |

After this configuration step, XML files corresponding to the circuits are assembled to create a Node Type object, based on Figure 4-6.

In this example, we consider only two nodes, Node A and Node B which are identical in terms of structure. The duration of the two node patterns is $t_{pat} = 60s$ ($F_P \approx 0{,}017\ Hz$). The pattern of Node A is shown in Figure 4-13. In the beginning of the pattern, the node is in the initial sleep state for $t_i = 0.01s$ (covered with wavy grey stripes). After the Processing Unit wakes up, it activates the Sensing Unit. After sensing is done, the Sensing Unit turns to the sleep phase and the Telecommunication Unit will be activated to send 100 Bytes of data over the wireless medium to Node B (Exchange phase). Finally, Telecommunication and Processing Units, one

after the other, pass to the Sleep phase and stay in it until the end of the pattern. The pattern, thus created, is attached to the node, using the Node Pattern object.

The pattern of the Node B is identical to the previous pattern except for two changes:

1) The Sensing Unit is not used. Thus, it is always in the Sleep phase.
2) After the wake up is done, the Telecommunication Unit is put to the Exchange phase directly waiting to receive data from Node A. This is considered to cover the possible synchronisation errors between the two nodes. Thus, Node B has to wait for Node A to start sending.

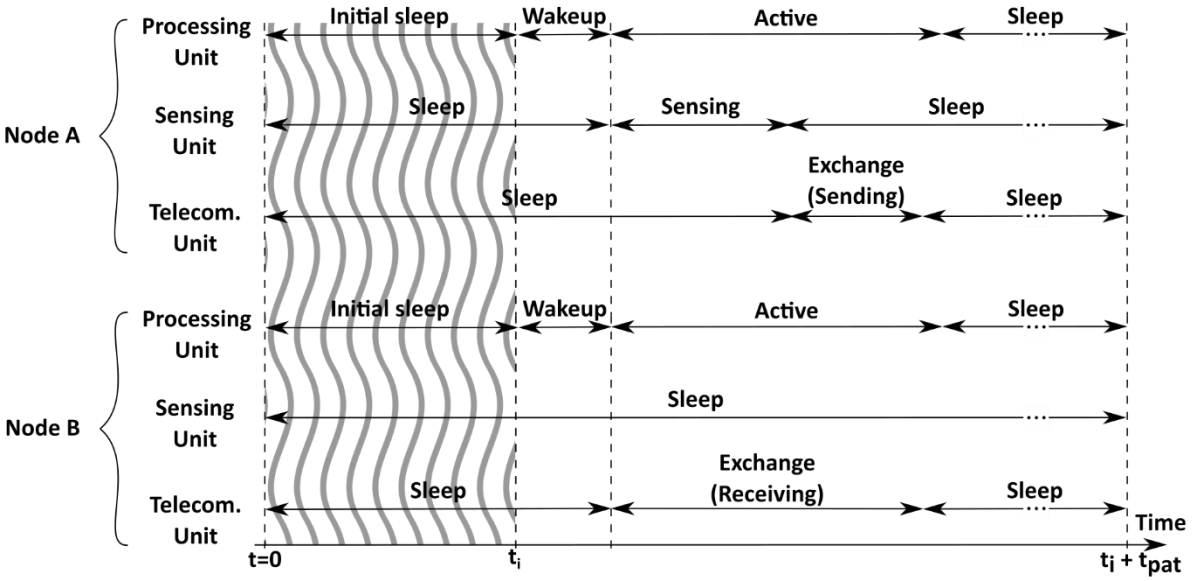



**Figure 4-13: Node A time sequence for the proposed scenario**

A Node Pattern and the Node Type objects are used to complete each Node object. In this example, as two nodes are used, two instances of the Node object are created.

After that, these nodes are assembled to create a network topology represented using a Topology object. The dimensions of the simulation area are set to 100 m x 100 m, and the constructed nodes will be attached to it. In this scenario, the two nodes are located at 20 m apart at ($x_1$=40 m, $y_1$=50 m) and ($x_2$=60 m, $y_2$=50 m) respectively.

Finally, a Scenario object is to be created, and the topology will be added to it as well as a wireless medium, configured as ideal, without physical obstacles. Therefore, in this propagation medium, signals can pass through with no additional distortions.

The last parameters of the Scenario object to be set are related to the simulation. The simulation can be configured to run on different time scales: seconds, hours, days and years. The simulation run time is an abstract value that can have meaning regarding the time scale only. In this

example, the time scale is “year” and the simulation run time is set to 1 resulting a simulation over 1 year.

### 4.2.4. Simulation and analysis stages

After the configuration is set, the simulation can be run. The results can be viewed for analysis using the built-in graphical user interface (GUI) that automatically appears after the simulation is done. The results can be displayed using a top-to-down approach. For the sake of comparison, several scenarios can be simulated before the result visualisation. Therefore, the user must first select the scenario to be presented in the GUI.

After that, the user can browse the results starting from the Topology Level, as illustrated in Figure 4-14. At the Topology Level, the user can see nodes' positions and has the possibility to select a node by name from a drop-down menu and this will provide details of the selected node.

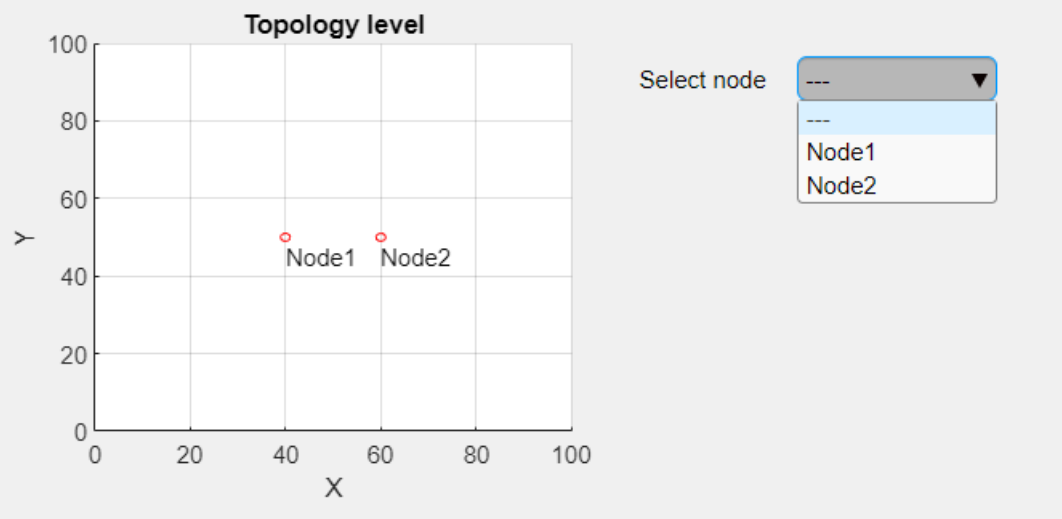
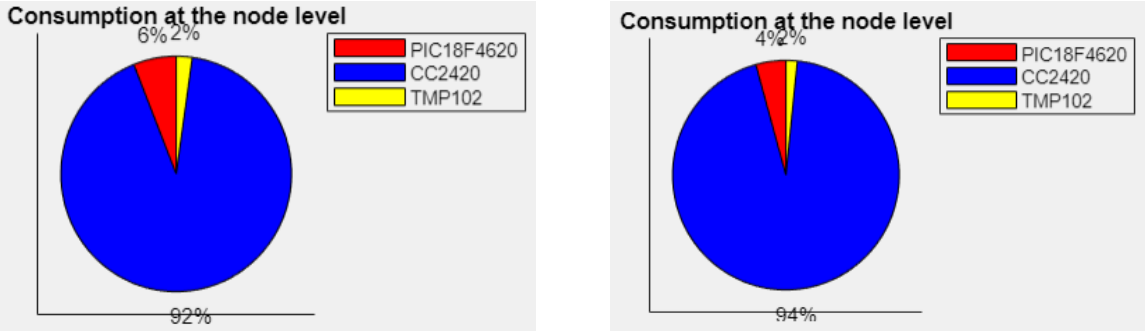



**Figure 4-14: Simulator output at the topology level**

Although Node A and Node B has identical structure, they consume energy in different ways, because they have two distinguishable patterns, as previously shown in Figure 4-13.

At the Node Level, user can have a general overview, presented in Figure 4-15, of the distribution of the energy consumption between the circuits of each node. In this scenario example, the Telecommunication unit of the Node B consume the larger amount of the node’s energy, because it stays for long time in receiving mode, as shown in the pattern obtained in Figure 4-17. The Sensing Unit is not activated in Node B, its energy consumption corresponds to the sleep phase.




Node A Node B

**Figure 4-15: Distribution of the energy consumption.**

At the Node Level, the user can view the node consumption pattern in terms of current and time. For example, Figure 4-16 shows the most important part of the Node A pattern. In this figure, only a small part of the sleep phase is shown. The different phases of this pattern are:

1) The node is in the sleep state.
2) The Processing Unit wakes up.
3) The Processing Unit and Sensing Unit are active. Sensing unit consumption is not visible in the figure due to its small relative value.
4) The Telecommunication Unit is activated. Access channel phase with two different current levels corresponding to waiting and sensing periods of the adopted algorithm are used to access the medium.
5) The Telecommunication Unit is sending data and receiving the acknowledgment.
6) All circuits are back to the sleep phase until the end of the pattern.

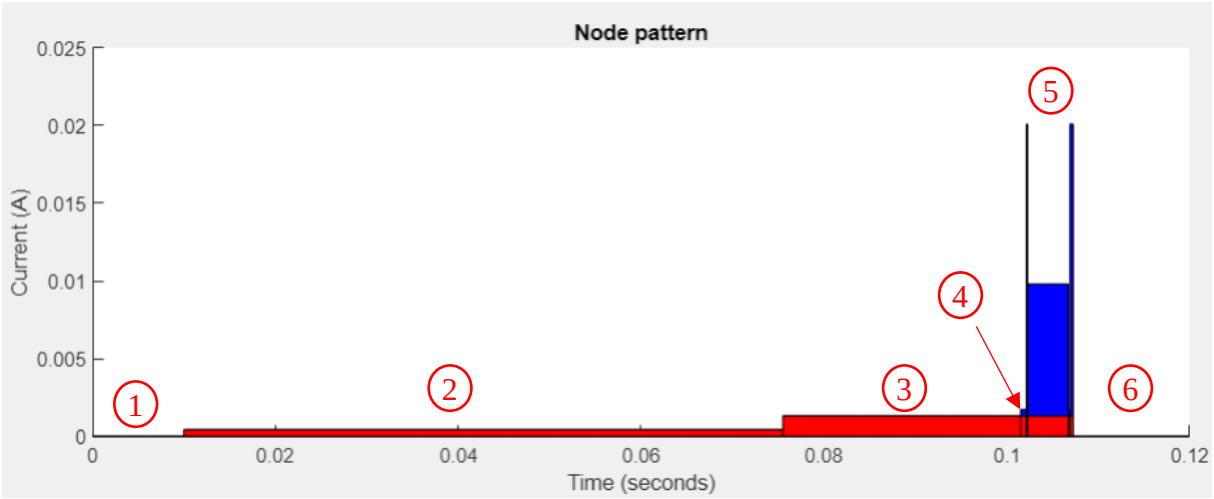



**Figure 4-16: Part of Node A consumption pattern**

For pattern of the Node B, presented in Figure 4-17, the phases are the following:

1) The node is in the sleep state.
2) The Processing Unit wakes up.
3) The Processing is active, and the Telecommunication Unit is waiting for a message. It is in the receiving mode.
4) The Telecommunication Unit is receiving data and sending an acknowledgment frame.
5) All circuits are back to the sleep phase until the end of the pattern.

All circuits are back to the sleep phase until the end of the pattern.

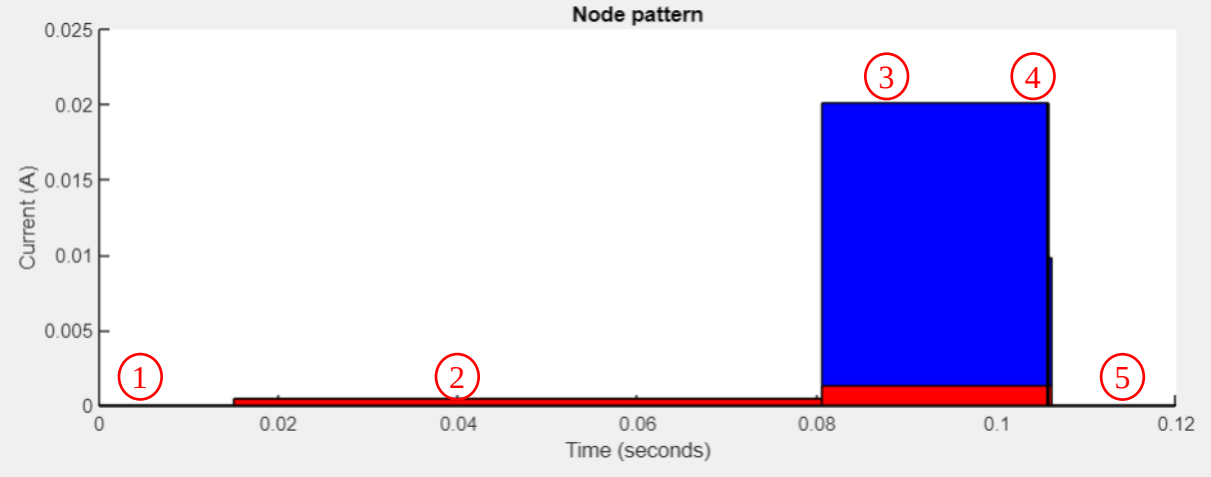



**Figure 4-17: Part of Node B consumption pattern**

At the Circuit Level, the simulator provides details of the consumption for each phase of each circuit of each node (Figure 4-18). The Processing Unit pattern is the same in Node A and Node B. It explains why the energy consumption of this unit is identical in the two nodes.

Regarding the Telecommunication Unit, Node A spent the major amount of energy in the Sleeping phase (SLP). This can be explained looking at the long time the nod stays in the sleep phase. There small amount of energy is spent sending the data (SND-DAT) and receiving the acknowledgment frame (RCV-ACK). The access phase is used only by the Node A because it is the only node that sends data in the proposed scenario. In Node B, the larger amount of energy is spent in the sleep state for the same mentioned in Node B. A remarkable amount of energy is also spent the energy receiving the data (RCV-DAT), it is due to the waiting time before receiving the message. Some energy is also spent sending the acknowledgment frame (SND-ACK).

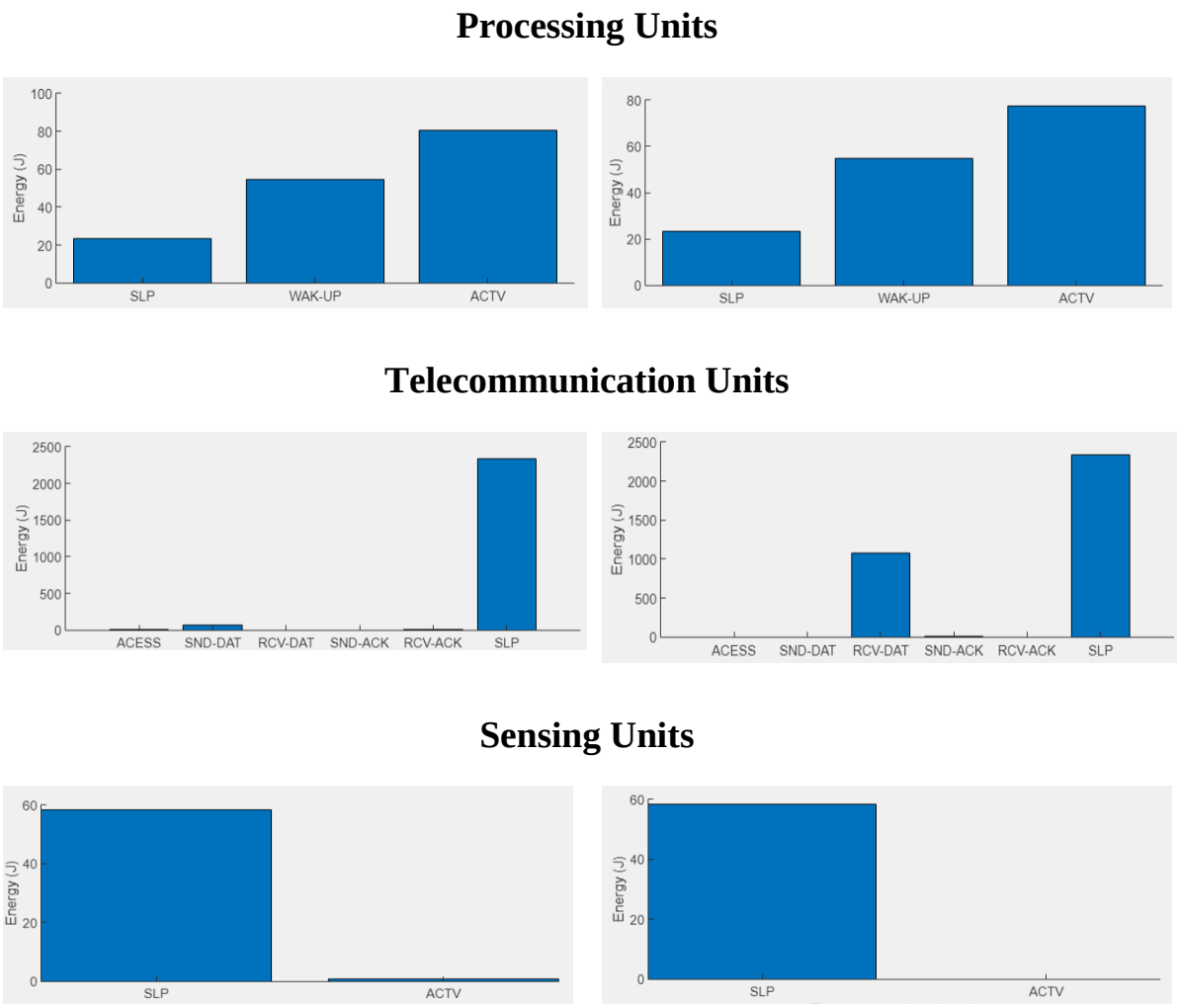



**Figure 4-18: Energy consumption in each circuit of each node**

In the Sensing Unit of Node A, most of the energy spent is in the Sleep phase (SLP) with a small portion in the active phase (ACTV) resulted from the measurement action. In Node B, the Sensing Unit was not used and was always in the sleep state, that explains why all the consumption is in the phase corresponding to that state.

The simulator also allows to estimate the node lifetime using Equation (3-22). For example, in the considered scenario the estimated lifetime for Node A is 9 years, for the Node B, it is 6.6 years. The results are in accord with the node activities considering that Node B Telecommunication Unit spend more energy of the peer unit in Node A.

## 4.2.5. Other functionalities of the simulator

The implemented simulator is capable to track cross-level interactions between parameters of the model. Based on the topology and the protocol parameters, some phenomena can be observed. For example, the transmission power must be adapted to the distance between the nodes and this imply the variation of the consumption. Another example is the effect of the data frame length (linked to the payload length) on the pattern of the Telecommunication Unit, because one or more fragment frames can be necessary (fragmentation).

### 4.2.5.1. Distance scenario

The distance between nodes could have an impact on energy consumption in the Telecommunication Unit. Indeed, the further the destination is, the higher the transmission power level selected must be. This explains why manufacturers generally provide a range of available transmission power levels in their telecommunication chipsets.

In our model, the relation between the selected transmission power level ($P_t$) and the distance between the nodes *d* is governed by the Friis equation (4-1):

$$P_T(dB) = -147{,}56 + 20\ \log_{10} f + \ 20 \log_{10} d \qquad (4\text{-}1)$$

where *f* is the frequency used by the link layer protocol, which is equal to 2.4 GHz in this case.

Table 4-2 recapitulates the maximum transmission distance d between two obtained from Equation (4-1) considering the different configuration transmission power level proposed in the CC2420 transceiver.

**Table 4-2: Relation between transmission power level, current consumption level and transmission distance for CC2420**

| Power level (dB) | Current level (mA) | Distance (m) |
|---|---|---|
| -25 | 8.5 | 31.5 |
| -15 | 9.9 | 99.5 |
| -10 | 11 | 177 |
| -5 | 14 | 315 |
| 0 | 17.4 | 560 |

The proposed model provides a way to trace these cross-level interactions, illustrated using red arrows in Figure 4-19. First, the distance between two nodes is a Topology-Level parameter. This parameter interacts with the power level selection in the Telecommunication Unit at the Circuit Level. Through intra-level interaction, the activity sequence in the Telecommunication Unit is also affected, as illustrated using a blue arrow on the same figure. After that, a cross-level interaction allows to reach the node pattern at the Node Level. Finally, energy

consumption in the battery, at the Circuit Level, and thus, the system's lifetime at the Use case Level are also affected.

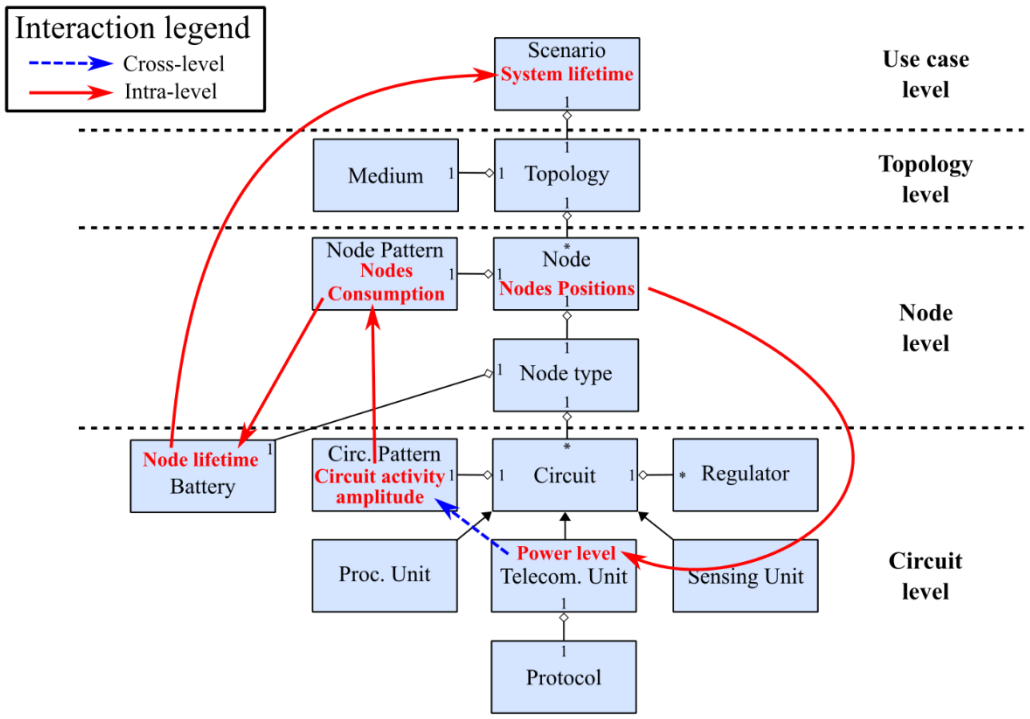



**Figure 4-19: Sequential interactions showing the impact of the node positions on the system lifetime.**

In order to show the previous cross-level interactions, the general scenario settings will be extended as follows. First, the payload length will be 20 Bytes in all scenarios. In addition, the distance between the two considered nodes will change to 25, 50, 100, 200 and 400 meters respectively. Those values were intentionally selected to cover the whole power level range proposed in the Telecommunication Unit's datasheet.

Figure 4-20 shows the obtained values for energy consumption of the Telecommunication Unit exchange phases from the distance and power level scenario. As the distance between nodes increases, the energy consumption in the exchange phase increases linearly. This can be explained from the chipset datasheet specification. As the distance rises above a particular threshold, recapitulated in Table 4-2, the model automatically changes the transmission power level to use the adapted value. This change has an impact on the energy consumed by the node. For example, when the distance between the two nodes changed from 25 to 400m, the corresponding consumed energy in the transmission phase changed from 60 to 120 μJ.

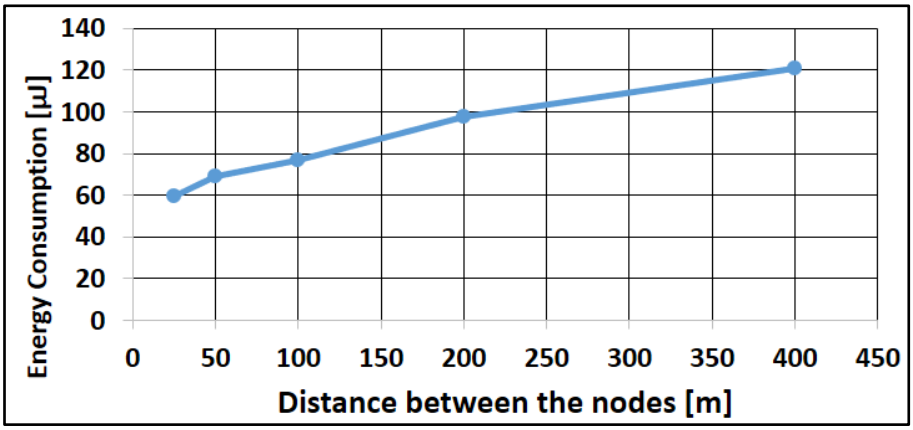



**Figure 4-20: One pattern energy consumption of the Telecommunication Unit exchange phase as a function to distance**

#### 4.2.5.2. Fragmentation scenario

Fragmentation is a mechanism to divide Protocol Data Units, i.e. frames or packets, into two or more units that are shorter in length. Fragmentation is used when WSN applications contain a constraint on the size of data units that could be transmit in one frame. For example, IEEE 802.15.4 has a maximum fragmentation threshold set to 127 Bytes, but any lower value down to one, can be set. Fragmentation threshold is a Topology level parameter which impacts the energy consumption of the telecommunication unit at the Circuit level. The proposed model can be used to show this cross-level interaction.

In our model, the fragmentation is implemented in the Telecommunication Unit. It takes place as illustrated by Figure 4-21:

1) The data frame arrived at the link layer coming from the network layer.
2) The unit calculate the length of the Link layer's PDU ($B_{L4PDU}$) using Equation (3-6).
3) The unit compare $B_{L4PDU}$ with the fragmentation threshold:
    a. **If** the fragmentation threshold is greater, a new link protocol header is created to encapsulate the packet, and the resulted frame is passed to the physical layer, **end**.
    b. **Otherwise**, the number of the fragment frames is calculated, and the following **is repeated** for each fragment frames
        1. New fragment is truncate from the original packet payload.
        2. A new link protocol header is created.
        3. A new fragment frame is created by encapsulating the new fragment and the new header.
        4. The fragment frame is passed to the physical layer.

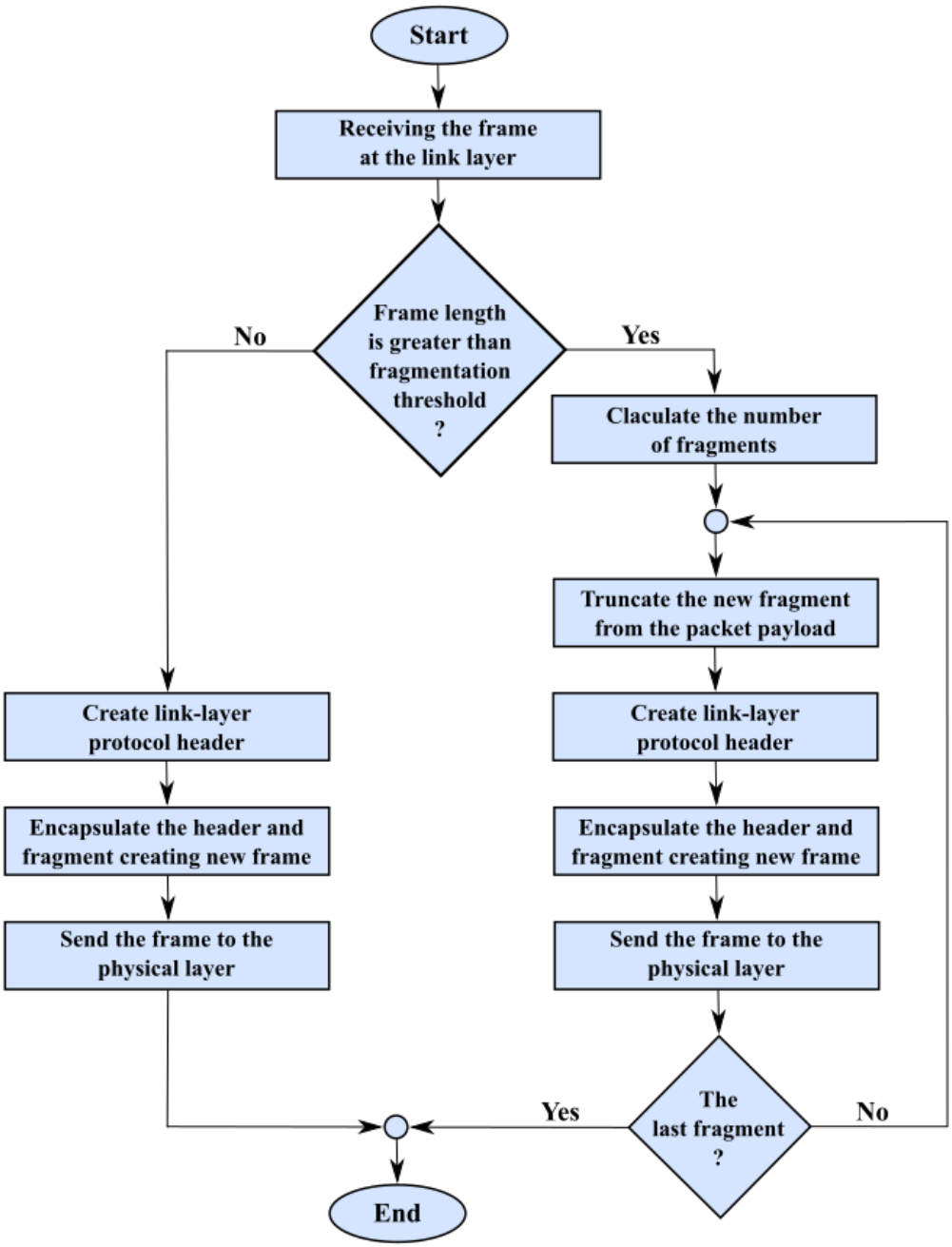



**Figure 4-21: The fragmentation algorithm used in our model**

In this scenario, the cross-level interactions take place as follows First, the fragmentation threshold will be changed. It is a Topology level parameter. This change will affect the protocol object, which is attached to the Telecommunication Unit object, and both are Circuit level parameters. These changes will affect the pattern structure at the node level, which, in turn, affects the energy consumption in the node, which is a Circuit level parameter. In the end, the system's lifetime will be affected by this cross-level chain interaction.

In practice, the settings of the complete node scenario will be implemented. However, the following changes will take place. The scenario will include sending a payload of 100 bytes,

and the fragmentation threshold is set to {40, 60, 80, 100} bytes respectively. These fragmentation thresholds are configured to test the proposed fragmentation algorithm. TMP102 sensor is not used in this scenario. It will always remain in the sleep state. Thus, its consumption is constant in all the scenarios and it will be ignored. On the other hand, Telecommunication and Processing units consumption changes as a response to the fragmentation setting, and these changes are to be traced. The node wakes up after 0.5 seconds from the pattern start and begins sending the data packets.

Figure 4-22 presents the results obtained from this fragmentation scenario. The energy consumption of the Telecommunication Unit phases is illustrated according to the fragmentation threshold. The consumption of access, exchange, and acknowledgment phases drop significantly after the first threshold. Energy consumption in the sleep phase changes slightly.

These results can be explained with the help of the number of frames sent in each scenario: 15, 4, 3 and 2 respectively. This sharp drop in the number of the sent frames is due to the effect of another Topology level parameter, namely protocol headers. The used protocols need to add headers to the payload. It is 33 bytes in total, and that leaves only 7 bytes for the payload when the fragmentation threshold is set to 40 bytes. However, the payload size is up to 27 bytes when the threshold is 60 bytes. Thus, 100 bytes of payload need only 4 frames to be completely sent. In general, these results show that energy consumption in the node decreases as the fragmentation threshold increases.

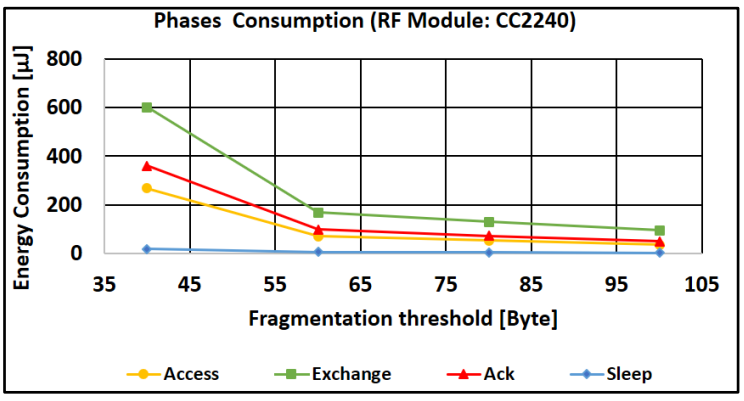



**Figure 4-22: Per-phase energy consumption from the fragmentation scenario for the Telecommunication Unit.**

The maximum value proposed by the protocol standard is 127 Bytes and we have chosen this value to illustrate the impact of the fragmentation mechanism on the node pattern. The scenario is identical to the one used in the Figure 4-13. The only difference is the payload to be sent which is configure to 1000 Bytes.

Figure 4-23 and Figure 4-24 show the patterns obtained for Node A and Node B respectively. It is possible to observe 11 frame that are sent by the Node A, received and acknowledged by the node B. This number can be explained by the fact that the fragmentation threshold is applied to the data frames and not the pure data. The headers added by each protocol are already included in the data frame.

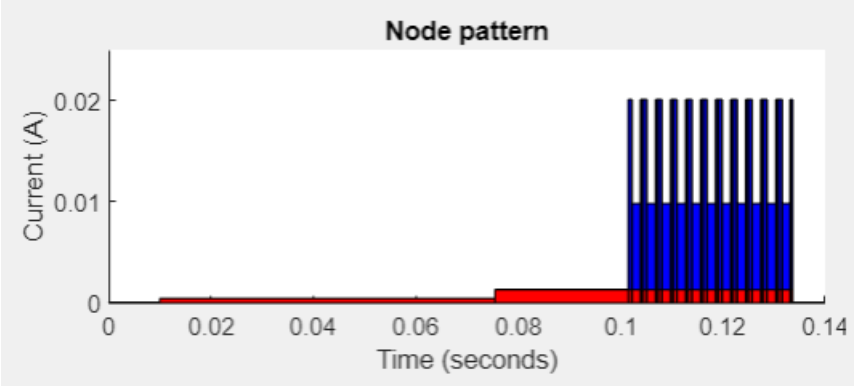



**Figure 4-23: Pattern of the fragmentation scenario for the Node A**

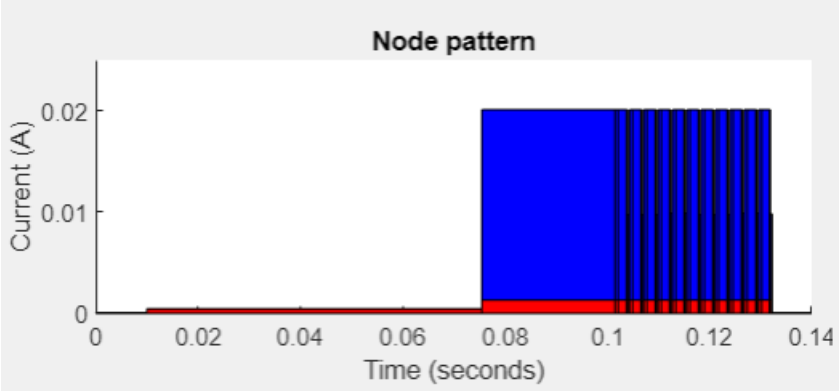



**Figure 4-24: Pattern of the fragmentation scenario for the Node B**

## 4.3. Comparison with NS2

In the previous sections, the implementation of our cross-level model in a simulator is described in detail, and an example is illustrated showing how the proposed simulator can be used. The next objective of our work is to demonstrate, based on the comparison between results obtained from NS2 and our simulator, the capabilities of the adopted cross-level design approach in an energy-aware context. NS2 is a network simulator that is widely used in the research domain. As shown in Chapter 2, NS2 energy model considers only the energy consumption of the Telecommunication Unit and does not cover other hardware component on the node.

Therefore, this section is structured as follows. Firstly, general and common settings for the scenarios to be used in this comparison are presented. After that, based on the proposed settings,

a scenario with two nodes communicating through the wireless medium is implemented both in NS2 and our simulator in order to compare the obtained results.

### 4.3.1. General settings

All the scenarios take place in an open area where there are two wireless nodes named Node A and Node B. Periodically, Node A sends a fixed-length payload to Node B through the wireless medium using its Telecommunication Unit.

Each node applies a TCP/IP network model. The implementation of the protocols starts at the network layer where Internet Protocol version 4 (IPv4) has been chosen. The Internet Control Message Protocol (ICMP), which is an integral part of IPv4, is used to create echo messages when necessary. Consequently, when the IPv4 module receives a data packet, it is going to send the same data back to the original source. In all scenarios, the length of IPv4 and ICMP headers are 20 and 4 bytes respectively.

Next, the link-layer parameters are to be set. Two different protocols are to be used, these are: IEEE 802.15.4 and IEEE 802.11a. The energy specification for the IEEE 802.15.4 module is derived from the CC2420 transceiver, providing by Texas Instruments. For IEEE802.11a, the specifications will be derived from an implemented chipset named HDG204, designing by H&D wireless.

NS2 does not consider the supplying voltage of the battery. It uses the Watt as a unit to present the energy consumed in the circuit. Therefore, in this set of scenarios, we have adapted the proposed simulator to be as close as possible to NS2 by ignoring the effect of the regulators.

Table 4-3 shows the settings for the two wireless link protocols. All scenarios begin after initialising the nodes.

**Table 4-3: Wireless Link Protocol Settings**

| Parameter | IEEE 802.11a (HDG204) | IEEE 802.15.4 (CC2420) |
|---|---|---|
| **Bitrate [bps]** | 12 M | 250 k |
| **Carrier Sense Mechanism** | Pure CSMA/CA | CCA-ED |
| **Transmitter power [mW]** | 725 | 52 |
| **Receiver Power [mW]** | 220 | 59 |
| **Sleep Power [mW]** | 0.2 | 0.06 |

The previous settings are general. They are to be implemented in all the scenarios. However, in the following sections, if additional settings are to be added, it will be explicitly mentioned.

## 4.3.2. Simulations using Telecommunication Unit only

In this section, the suggested scenario, to compare obtained results from NS2 and our simulator, will use only the Telecommunication Unit. The reason behind this choice is the need to have comparable results. Therefore, we have to consider that NS2 does not include models for other hardware components.

### 4.3.2.1. Scenarios implementation and obtained results

Pattern frequency ($F_p$) and payload size are UL parameters. In this set of scenarios, we trace the cross-level effects of the two on the consumed energy, which is a CL parameter. To effectively trace payload size and $F_p$, the following settings are to be added to the general settings presented previously. First, the two nodes will reside 10 m apart. Second, as stated previously, both IEEE 802.11a IEEE 802.15.4 are to be used. For each protocol, three different values of $F_p$ (0.1, 1 and 2 Hz), and ten values for the payload length, ranging between 10 to 100 bytes, are used. Each scenario requires a combination of the three parameters previously mentioned. As a result, there are 30 scenarios to be run for each wireless link protocol. Table 4-4 summarises the setting for the scenarios with only Telecommunication Unit included.

**Table 4-4: General settings of the scenarios limited to the Telecommunication Unit.**

| | **Value** |
|---|---|
| **Number of the nodes** | 2 |
| **Node positions** | (10,10), (10,20) [m] |
| **Scenario duration $T_{Sce}$** | 100 [s] |
| **Pattern Frequency $F_p$** | 0.1, 1, 2 [Hz] |
| **Payload length** | 10, 20, …, 100 [Byte] |
| **Link protocol** | IEEE 802.11a, IEEE 802.15.4 |

Figure 4-25 shows the pattern obtained for Node B using the proposed model, with the scenario where payload length and pattern frequency $F_p$ are fixed respectively at 100 Bytes and 1 Hz. The left side of the figure is dedicated to IEEE 802.11a and the right side to IEEE 802.15.4.

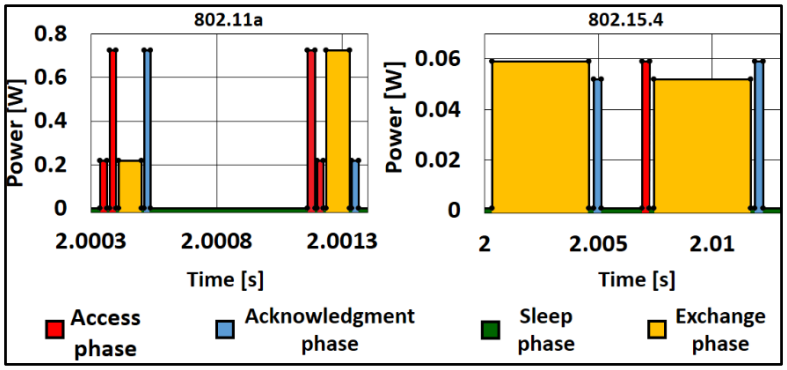



**Figure 4-25: Energy consumption Node B patterns for the two considered wireless link protocols with Payload length = 100 Bytes and $F_p$ = 1 Hz**

The main results, using the previously described scenarios with $F_p$ fixed at 1 Hz, obtained with NS2 and our model can be found in Table 4-5. The upper part shows the obtained results from IEEE 802.11a and the lower part for IEEE 802.15.4. Each part is further divided into two subparts, corresponding to scenarios with 10 and 100 bytes of the payload length respectively. These results present the energy consumed by different activities of the Telecommunication Unit. These activities are categorised into 4 phases which are defined and described in Chapter 3: access phase ($ph_{acc}$), exchange phase ($ph_{exch}$), acknowledge phase ($ph_{ack}$) and sleep phase ($ph_{slp}$). In these phases, as stated also in the previous chapter, the cross-level interaction between parameters is taking place. For example, in the exchange phase, the consumed energy depends on the headers' lengths, payload length, and bit rate, which belongs to UL, SL and NL levels.

For the two protocols, when comparing obtained results from NS2 and our proposed model, differences and similarities can be found as follows. The energy consumed in $ph_{acc}$ or in $ph_{ack}$ are identical and there is a slight difference in the energy consumed in $ph_{slp}$. In exchange phase $ph_{exch}$, the difference is notable, but stable and this is due to different interpretations of the link protocol specifications. For example, in our model, the ICMP header is considered to be part of the data packet, contrary to NS2 where this header is added to the data packet later.

**Table 4-5: Energy Consumption of the phases in different scenarios ($F_P$ = 1 Hz)**

| Simulation | Consumed energy per phase [µJ] | | | | |
|---|---|---|---|---|---|
| | **Access** | **Exchange** | **Acknowledgment** | **Sleep** | **Total** |
| **IEEE 802.11a** | | | | | |
| **10 Bytes** | | | | | |
| **Our Simulator** | 51.03 | 50.08 | 23.63 | 199.959 | 324.69 |
| **NS2** | 51.03 | 55.76 | 23.63 | 199.950 | 330.37 |
| **50 Bytes** | | | | | |
| **Our Simulator** | 51.03 | 75.60 | 23.63 | 199.940 | 343.18 |
| **NS2** | 51.03 | 80.33 | 23.63 | 199.945 | 354.39 |
| **100 Bytes** | | | | | |
| **Our Simulator** | 51.03 | 106.79 | 23.63 | 199.935 | 381.38 |
| **NS2** | 51.03 | 112.46 | 23.63 | 199.927 | 387.04 |
| **IEEE 802.15.4** | | | | | |
| **10 Bytes** | | | | | |
| **Our Simulator** | 18.89 | 145.73 | 39.07 | 59.78 | 263.47 |
| **NS2** | 18.88 | 152.84 | 39.07 | 59.78 | 270.47 |
| **50 Bytes** | | | | | |
| **Our Simulator** | 18.89 | 287.72 | 39.07 | 59.62 | 405.31 |
| **NS2** | 18.88 | 294.82 | 39.07 | 59.63 | 412.39 |
| **100 Bytes** | | | | | |
| **Our Simulator** | 18.89 | 465.37 | 39.07 | 59.44 | 582.77 |
| **NS2** | 18.88 | 472.42 | 39.07 | 59.45 | 589.82 |

#### 4.3.2.2. Relative error

Using the obtained results, as illustrated in Figure 4-26, a set of relative errors between NS2 and our model is provided considering the different values of $F_p$ and payload length used in the scenarios.

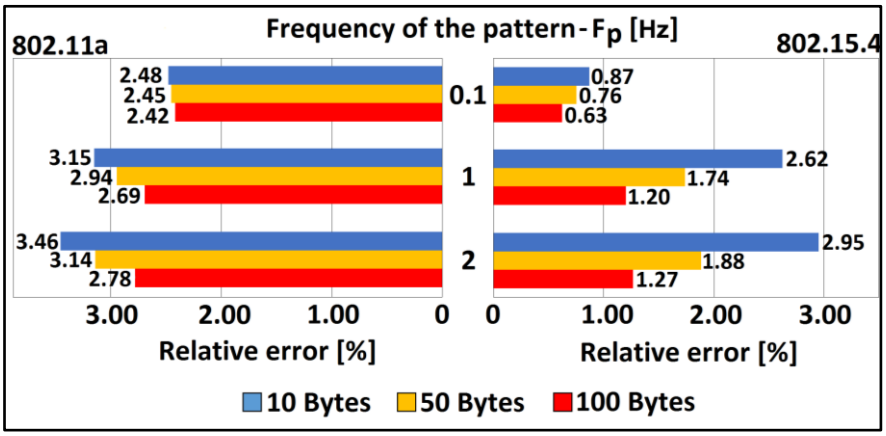



**Figure 4-26: Relative error between our simulator and NS2.**

In all the scenarios, the relative errors obtained from IEEE 802.11a are greater than those of the corresponding scenarios of IEEE 802.15.4. This difference can be explained by unplanned and non-periodic radio activities that appear periodically in NS2 IEEE 802.11a simulations. These activities have a fixed duration regardless of $F_p$ and the payload length. Each of these activities appears as a single pulse of transmission or reception causing an additional energy consumption of around 5 µJ and 2 µJ respectively.

Finally, the value of the relative error between NS2 and the proposed model did not exceed 3.5%. As stated previously, NS2 is widely used, in research domain, to study and evaluate communication protocols and algorithms in different kind of networks. Considering this simulator as a reference point from the network perspectives, we can consider the Telecommunication part of our model to be validated. Although not shown in this scenario, our proposed WSNs model also allows adding energy consumption phases for other hardware on the node, such as the processing unit or sensors. As a result, an accurate pattern can be constructed representing precisely the real consumption of the node contrary to NS2 that has a poor support for hardware, as demonstrated before in Chapter 2.

#### 4.3.2.3. Selection of the Wireless Link Protocol

To demonstrate the relevance of the proposed modelling approach, the results obtained from our simulator will be applied, in this section, to define the energy-efficiency boundary between

the two link protocols used previously. Based on these results, the objective is to detect which wireless link protocol is more energy-efficient in a specific scenario.

As defined in Equation (3-19), the energy consumed by one pattern of wireless link protocols depends on the Telecommunication Unit circuit and parameters shown in Table 4-4. Figure 4-27 presents the difference of energy consumed between the two protocols, respectively $E_{pat802.11a}$ and $E_{pat802.15.4}$, as a function of both $F_p$ and the payload length.

In this comparison, three situations for the energy efficiency can be distinguished:

- The red curve. It represents the equal-energy boundary. For scenarios that are positioned on the curve, the two protocols consume the same amount of energy.
- The blue area above the red curve. In this area, 802.11a is more energy-efficient than 802.15.4. For all scenarios that are positioned inside this area, it is recommended to use 802.11a as a wireless link protocol from an energy point of view.
- The yellow area below the red curve. In this area, 802.15.4 is more energy-efficient than 802.11a. For all scenarios that are positioned inside this area, it is recommended to use 802.15.4 as a wireless link protocol.

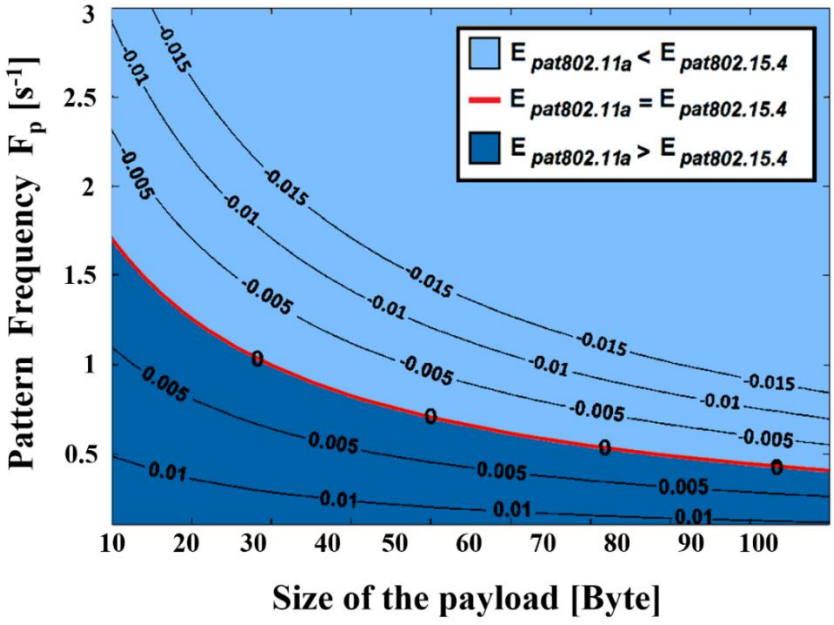



**Figure 4-27: The equal-energy curve as a function to $F_p$ and the payload size for the scenario limited only to the Telecommunication Unit**

Consequently, when $F_p$ is 2 Hz, the use of 802.11a is more efficient in terms of energy consumption. On the contrary, when $F_p$ is 0.1 Hz, the use of 802.15.4 is more energy efficient. In the case where $F_p$ is 1 Hz, the choice is a function of the payload size. If it is less than 30

Bytes, the use of 802.15.4 is more efficient. Conversely, when the payload size is greater than 30 Bytes, the use of 802.11a is more effective from an energy perspective.

The previous discussion is issued from an example of cross-level interaction where parameters from the use case level, such as the payload size, impact the energy consumed by the circuit, which is a circuit level parameter. This example demonstrates the need to consider and study the impact of these cross-level interactions in order to make the best compromise in the early design stages of a WSN.

## 4.4. Validation with real nodes

Physical WSN nodes were implemented in order to confront results obtained from the simulator with those obtained from real nodes. In this section, a description of the circuits used in the validation process will be carried out. After that, the results taken from the implementation will be presented and compared with those obtained with our simulator.

### 4.4.1. Description of the circuits

To test scenarios using real equipment, as illustrated in Figure 4-28, two WSN nodes were used. Node A creates data and sends it while Node B receives it. In these scenarios, no Sensing Units were used. The consumption of Node A will be measured, and results will be visualised using an oscilloscope.

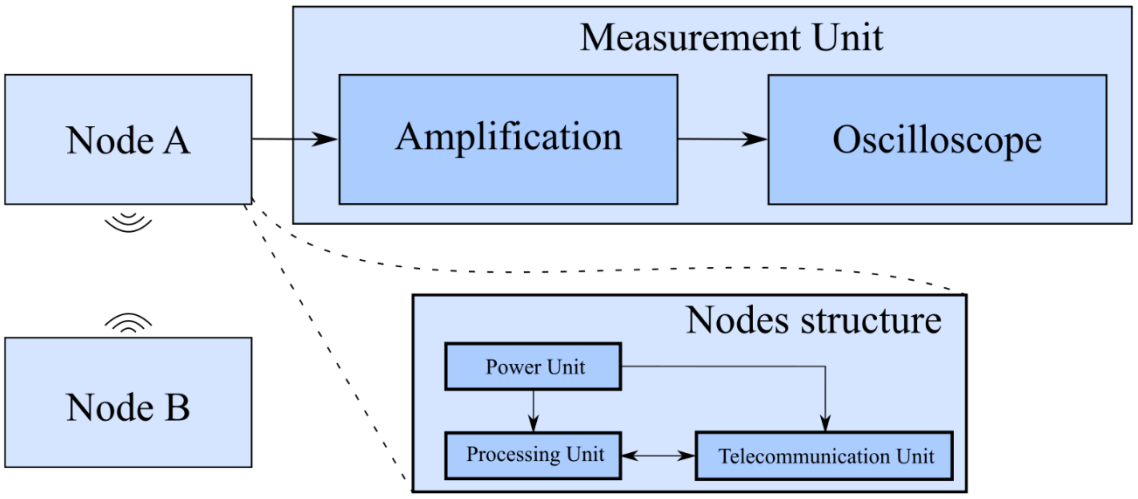



**Figure 4-28: Schematic structure for the equipment used in the validation stage**

In next sections, details are given about the WSN node implementation and the measurement unit used for this experimental part of our work.

#### 4.4.1.1. WSN node

As presented in Figure 4-29, the considered WSN nodes are composed of three parts: the battery, the Processing Unit, and the Telecommunication Unit. Two regulators are also used to supply the circuits with the required voltage level. They sit in-between the battery and the circuits.

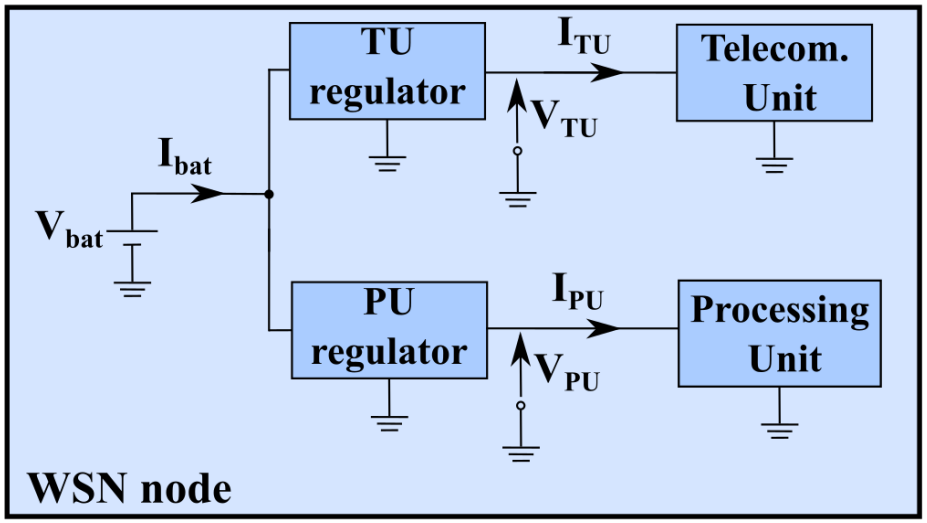



**Figure 4-29: Block diagram for the WSN node**

Node consumption is computed based on current measurements and nominal voltage values using Equation (4-2).

$$P_i = V_i \cdot I_i \tag{4-2}$$

A microcontroller is used as a Processing Unit, namely the PIC18F4620. It supports the nanowatt technologies, i.e., they can significantly reduce power consumption during operation. Thus, this microcontroller family is suitable for WSN implementation. For the Telecommunication Unit, the MRF24J40 is chosen. It is a radiofrequency transceiver module that supports 2.4 GHz IEEE 802.15.4.

The communication between the Processing Unit and the Telecommunication Unit is made, as shown in Figure 4-30, through a Serial Peripheral Interface (SPI). For this serial communication, four signals are necessary: Chip Select (CS), Slave Clock (SCK), Slave Data Input (SDI) and Slave Data Output (SDO). Furthermore, three other pins are required to control other necessary functions: Reset (RST) pin for restart the MRF24J40 [MRF10], Wake-up (WAKE) to wake up the considered transceiver from the sleep state, and INT to handle the

interruption generated by the RF circuit to inform the Processing Unit that it has received a data frame.

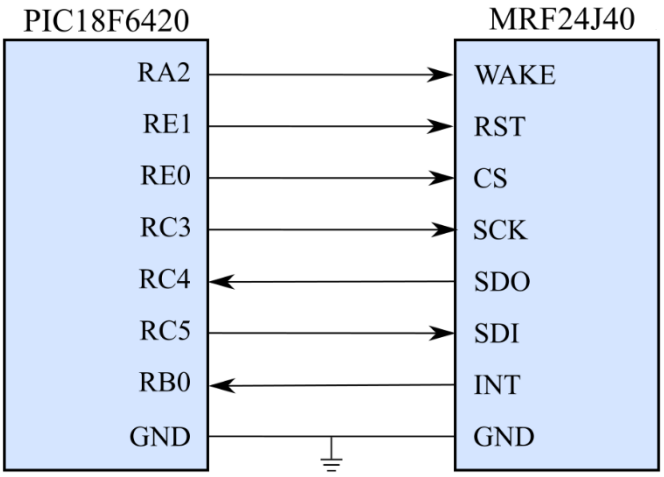



**Figure 4-30: Pin connections between PIC18F6420 and MRF24J40**

Finally, a rechargeable power bank was used to supply the circuit through a USB interface (5V).

#### 4.4.1.2. Measurement unit

To measure the needed currents, we have chosen to use sense resistors. To achieve it, as illustrated in Figure 4-31, small-value shunt resistors will be added in a serial connection between regulators and the targeted circuits but also between the circuit representing the battery and those regulators.

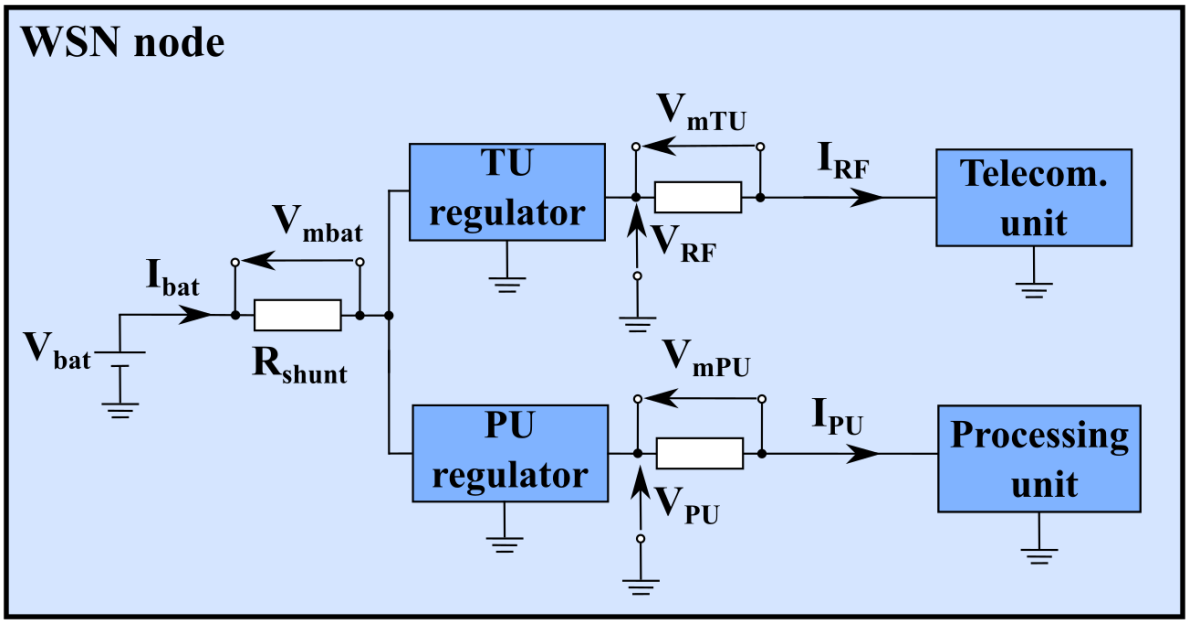



**Figure 4-31: Block diagram for the current measurement**

Using this method, the voltage drop across each resistor will be the image of the current. In order not to influence the function of the circuit, this drop must be as small as possible. The signal must be amplified to adapt it to the available voltage range supported by the analogue-to-digital converter (ADC) integrated in the microcontroller.

For this purpose, an amplification circuit that includes three stages is designed and presented in Figure 4-32. Each one provides a gain factor of 10, resulting in three possible values for the amplification factor: 10, 100 and 1000. The combination between the amplification factor and the sense resistors values provide the possibility to measure a wide range of currents. OPA2353, a rail-to-rail CMOS operational amplifier designed for low cost and miniature applications, is chosen to be integrated in this amplification circuit. It is optimised for low voltage, single-supply operation, perfectly adapted to work in microcontroller based-systems. Its high-speed operation (44MHz, 22V/µs) make it ideal for the amplification of signals with fast amplitude variation, such as the node drawn current, and for driving sampling analogue-to-digital converters [OPA98]. A set of high precision resistors (0.1%), were used to fix the gain of each stage of this amplification circuit.

In Figure 4-32, the elements marked in red represent the circuit that is being studied, respectively Telecommunication or Processing Units, and the sensing resistor $R_S$ that is a 1Ω resistor.

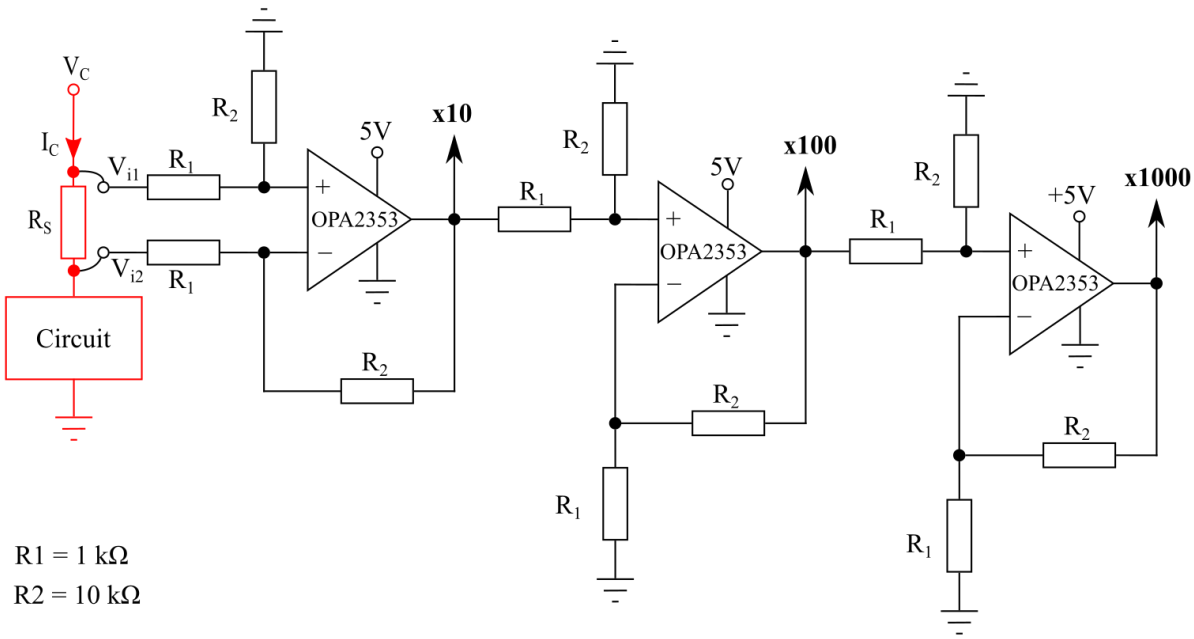



**Figure 4-32: Electronical diagram for the amplification circuit**

The current consumption to be measured is very different from one phase to another and also from one circuit to another. For example, the sleep current of PIC18F4620 is 0.1 µA, whereas the current consumed by MRF24J40 in transmission state is 23 mA, the ratio between the two values is more than 200000.

One of the possible solutions to handle this situation is by using an adaptive multiplexing. This can be achieved, as illustrated in Figure 4-33, by connecting the output of each amplification stage to an analogue input of an high-speed multiplexer which constantly reads the input signals, and select the one with most suitable amplitude to convert it into a digital form and deliver it to the oscilloscope.

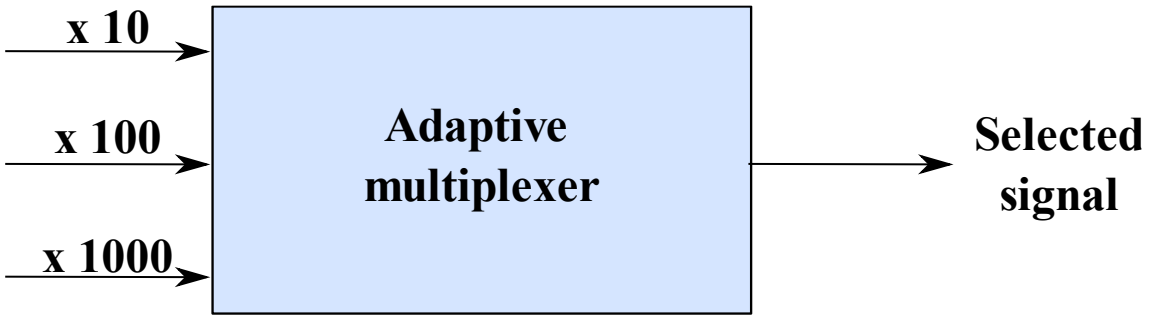



**Figure 4-33: Adaptive multiplexing for the amplification stages outputs**

During the first experimental part of this thesis, the implementation of the multiplexing concept was not possible. Therefore, the signal acquisitions were made using a digital oscilloscope Tektronix MSO2014B, 100 MHz, 1 GSPS.

### 4.4.2. Results

In this section, the circuits described above are used to create test scenarios, the obtained results are compared with results taken form the simulator where the same scenarios are implemented. As stated previously in Figure 4-28, the experimental scenario is based on two identical nodes, Node A and Node B.

The main objective of these experiments is to study only the Telecommunication Unit pattern. Therefore, in these tests, no sensors were used and the sleep mode was implemented only for the Telecommunication Unit of Node A. Moreover, measurements are made only on this node. Node B is always in the reception mode and it is used just to acknowledge the messages received from the Node A.

Thus, as presented in Figure 4-34, Processing Unit of Node A is always in the active modeFigure 4-33. At the beginning of the execution (not represented in the figure above), the Telecommunication Unit is configured and put in sleep state. It is waked up periodically by the

Processing Unit and sends one data frame. After that, it waits for the acknowledgment frame coming from Node B and going to sleep for 10 ms. This sleep duration, even if it is not representative in WSN, is chosen to facilitate the acquisition of the signal.

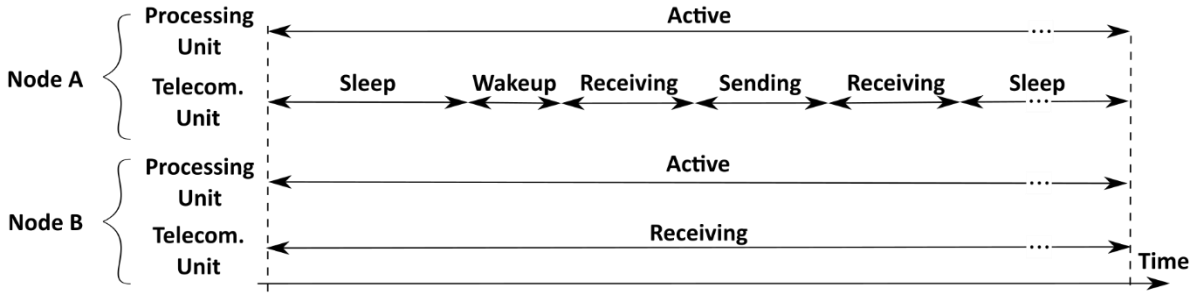



**Figure 4-34: Node patterns for the experimental scenario**

In practice, on real nodes, enforcing this specific pattern is a difficult task because some of the circuit parameters are preconfigured by the manufacturer and cannot be changed. For example, after an immediate wake up, waiting periods are needed in order to achieve the calibration of the Telecommunication Unit oscillator. During those periods the consumption of the circuit is the same as in the receiving state. The real duration of the calibration phase is not constant, and MRF24J40 datasheet suggests considering a duration of: 192 µs + 2 ms (Appendix 2 - Figure B-5 and Figure B-6).

Even after those periods, the Telecommunication Unit is not able to begin sending data. It is not possible to master the exact moment when the transmission starts because the MRF24J40 allows only to load data to the transmission FIFO queue, the circuit send it later when it is ready. Using our experimental testbed, as illustrated in Figure 4-35, we have observed that delay between the immediate wakeup and the start of the sending phase lasts between 4.08 and 6.40 ms.

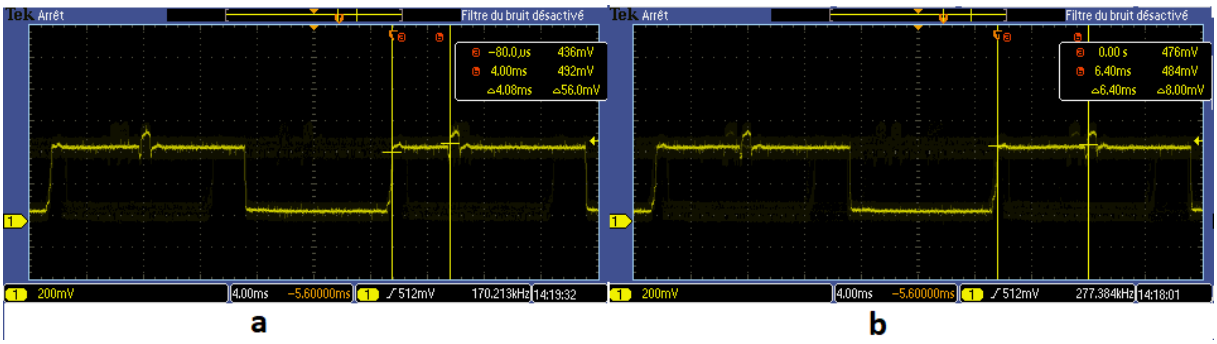



**Figure 4-35: Two measures for the delay before the transmission.**

Theoretically, after sending a data frame, Telecommunication Unit of Node A should receive an acknowledgment frame after a short and constant period equal to $t_{w\_ack}$, as suggested in

Figure 3-14. However, in practice this is not the case, we noticed that the acknowledgment frame sent by Node B could be delayed. Therefore, if Node A goes to sleep without receiving this acknowledgment frame, it will assume that the data frame is lost, and it will try to send the same data frame once again in the next pattern. To prevent this, we have imposed a 10 ms delay after loading the data to the FIFO queue, as shown in Figure 4-36.

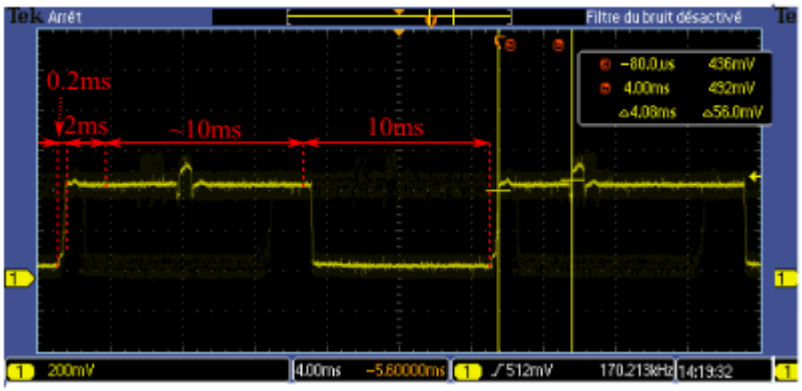


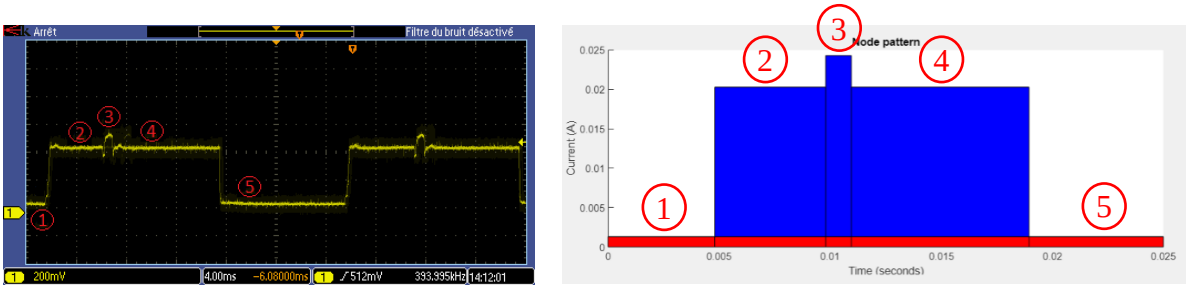

**Figure 4-36: Time delay imposed in the scenario**

After refining scenario using the previous results, the left part of Figure 4-37 shows the measured voltage corresponding to the current drawn from the battery by the Telecommunication Unit. The value of the sensing resistor is 1Ω and the amplification factor is 10 (100mV correspond to 10mA). On the right, the full node pattern where the blue area stands for the Telecommunication Unit pattern, and the red one is for the Processing Unit pattern.

Considering both part of Figure 4-37, the following phases of Node A Telecommunication Unit can be distinguished:

1) Sleep phase, where the current is at the lowest level.
2) Waiting phase corresponding to the preparation of the transmission. The wakeup is a very short phase that ensure the transition between the phases 1 and 2.
3) Exchange (Sending) phase.
4) Waiting phase necessary to ensure the reception of the acknowledgment frame.
5) Sleep phase until the end of the pattern.

Therefore, comparison of the two patterns shows that they are similar in terms of behavior and phases. However, in the simulation, it is not possible to consider a variable time for phase 2, but, from energetic point of view, the relative position of the exchange phase (phase 3) does not affect the consumption, only the duration is important and this duration is always the same.




a b

**Figure 4-37: Telecommunication unit consumption pattern (a) measured from the physical node (b) simulated in our simulator**

As expected, the Telecommunication Unit consumes the most amount of energy in the waiting phases (phases 2 and 4) corresponding to the receiving consumption level (RCV-DAT), as shown in Figure **4-38**. In this figure, the consumed energy is considered just for one pattern because the lifetime calculation is not considered in this experiment.

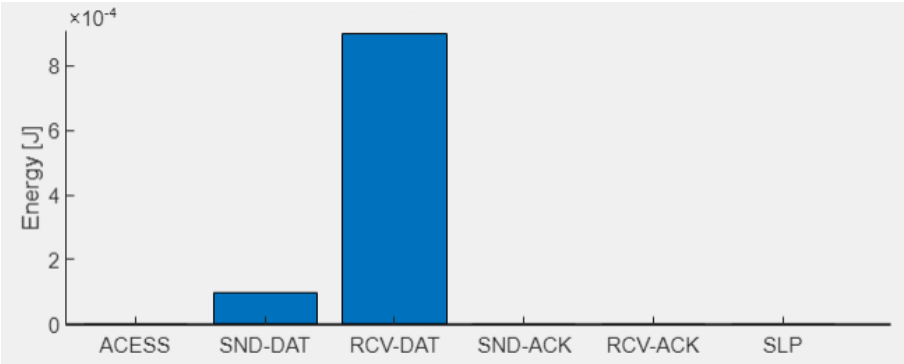



**Figure 4-38: Energy consumption by phase for the Telecommunication Unit**

Limitations observed in the experimental part of our work forced us to adopt a very specific scenario. Those limitations are linked to the configuration capabilities of the MRF24J40. Thus, other tests including different Telecommunication Units would be interesting to verify if the constated drawbacks are the same.

## 4.5. Conclusion

The first part of the Chapter 4 is dedicated to the implementation of the proposed model in order to create a cross-level energy-aware simulator for WSN. In second part, the simulator is compared with NS2, using a set of scenarios implemented in both simulators. The aim is to detect differences and highlight the cross-level interactions that can be traced in the developed simulator. In the last part, to validate the results, the real scenarios are created using physical WSN nodes, the obtained result from measurements and simulators were then studied.

The model proposed in Chapter 3 was implemented using MATLAB. The resulted simulator accepts commands and XML files as inputs to create WSN. The simulation results are shown

using a built-in graphical interface that is dedicated to reflecting the cross-level approach and the energy awareness of the model. The creation of a scenario is described in details and the obtained results were extended to show simulator capabilities as the effect of the distance and the fragmentation on the consumption of the energy. In general, the obtained results show the simulator's ability to provide energy consumption at different levels of abstraction. This includes the total energy consumed by each node, the consumption of one specific circuit, as well as that of a particular phase.

During the comparison with NS2, only the Telecommunication Unit was considered. The comparison showed a relative error percentage below 3.5% for the considered scenarios. The model's usefulness has also been illustrated by comparing two wireless link protocol (802.11a and 802.15.4) from an energetic point of view.

The main objective of the experimental part was to confront results obtained both from the simulation and real nodes. Although, it was not possible to correctly configure all the needed parameters for the selected telecommunication circuit, it has been demonstrated that our simulator is able to generate the corresponding consumption pattern from the considered scenario. Based on these observations from the experimental part, it seems that models implemented in other existing WSN simulators, considering the evaluation made in Chapter 2, include assumptions that simplify too much circuit and node behaviour resulting in possible important errors especially on the node lifetime estimation.

# GENERAL CONCLUSION

## Summary of work and contributions

This thesis has investigated the modelling and the simulation of energy in Wireless Sensor Networks. Based on a cross-level approach, an energy-aware model has been proposed and implemented creating a new WSN simulator. The results obtained using this simulator have been used to validate our model proposal and to demonstrate its relevance regarding the research questions and the set of issues, detailed in Chapter 2, related to modelling approach and energy awareness, we have to address in this work.

The proposed model is firstly based on a set of structural concepts (level, object, parameters) that are defined to model the interactions as follows (Question 1.1). Objects are used to model WSN components from topology to circuit points of view. Each object includes a set of parameters that could interact together. The nature of this interaction depends on the object locations. If they are in the same level, it is defined as an intra-level interaction, otherwise it is a cross-level one.

To implement these structural concepts (Question 1.2), a four-level model is adopted to cover the different points of view that we have to consider in a WSN: circuit, node and topology. A fourth level, called the use case level, is added to represent an instance used to describe an applicative scenario. Each level includes a set of objects used to represent structural and behavioural descriptions of WSN components allowing intra-level and cross-level interactions to take place in this multilevel structure.

Moreover, to introduce the temporal perspective, a set of concepts was also defined, including phase and pattern (Question 1.3). Depending on the adopted scenario defined in the use case level, the pattern concept, composed of one or more phases, is used to represent activities of each unit that composed a node from temporal and energetic points of view. As demonstrated in our model, the duration of each phase is a function of parameters that belongs to different levels. The circuit patterns are synchronised to create node pattern that is assumed, in this first version of our model, to be repeated periodically. From this perspective, our model is adapted to heterogeneous WSN because different types of node can be modelled and, even if nodes have the same structure, they can have different patterns.

Considering the capacity of our model to handle energy from cross-level perspective (Question 2.1), based on the proposed structural concepts, energy-related WSN properties, located in different levels, can also interact together or with other non-energy-oriented parameters to show how cross-level and intra-level interactions can impact the overall node consumption and lifetime. Moreover, to improve the energy awareness of our model (Question 2.2), the circuit's regulators impact is considered in the energy calculation. Thus, adopting and implementing the proposed concepts and multilevel structure, our model allows to study simultaneously how parameters that belong to different levels of abstraction, such as phase sequence and unit power levels, mainly influence temporal and energetic perspectives.

To validate our proposal and to demonstrate its relevance in a context of energy awareness WSN design, we create a cross-level energy-aware WSN simulator based on our model. The simulation results obtained are firstly compared with NS2, one of the most widely used network simulator in WSN research. Simulating only Telecommunication Unit activities, the relative error between results obtained from the two simulators was less than 3.5% which proves that our simulator can provide the same accuracy than NS2 in this kind of scenarios. Moreover, it has been demonstrated that, contrary to NS2, our simulator could provide more detailed information about the energy consumptions by considering not only Telecommunication Unit but also Processing and Sensing Unit activities of a node. In addition, several scenarios are implemented, using the proposed simulator, to point out the cross-level capability of our model. The obtained results demonstrate the relevance of this cross-level modelling approach to evaluate the impact of, for example, distance between nodes on their energy consumption but also to study, in WSN design stages, trade-offs that we have to consider between packet size, duty-cycle, and wireless sensor protocols. For this aim, it has been shown that the proposed simulator also success to trace energy consumption from different levels at the same time, such as the energy consumption of each node component but also the part of each circuit phase on the overall node consumption. Finally, we showed that our simulator can be used to trace the consumption of a real node, even if the behaviour of the telecommunication circuit is more complex than what is generally considered.

Finally, in the following Table I, we extend Table 2-5 by including our simulator to provide a comparative perspective with simulators studied in Chapter 2.

**Table 5-1: Wireless Comparation between our simulator and the selected simulators based on the modelling approach, energy-awareness and Heterogeneity**

| Item | | NS2 | OMNet++ | TOSSIM | IDEA1 | Our simulator |
|---|---|---|---|---|---|---|
| **Modelling approach** | | Single-level | | | Multi-level | Cross-level |
| **Energy-awareness** | **Consumption** | Yes | Yes | No Energy Model | Yes | Yes |
| | **Generation** | No | Yes | | No | In-progress |
| | **Battery** | Yes | Yes | | Yes | Yes |
| **Heterogeneity** | | No support for hardware | | Yes | | Yes |

# Future works

The results obtained in this thesis have demonstrated the relevance of the proposed cross-level energy-aware model to answer the modelling requirements and energy constraints of WSN. However, as stated previously, the investigated research area of WSN energy modelling and simulation is vast and complex. Thus, there are many additional future works that must be conducted to improve our propositions.

- For the Processing Unit, a simple model is used, with only three phases: sleep, wake-up and active. However, especially in active state, the behaviour of this unit can be much more complex. Indeed, activities such as analogue-to-digital conversion, data processing or serial bus communication can consume different amounts of energy. Thus, **extending the Processing Unit modelling** is a possible enhancement to our proposal.
- The wireless medium is also implemented using a basic model. No obstacles and no signal propagation models are included. Extending our model to include more sophisticated **features in the wireless medium** object is an interesting task, especially those which could impact the node energy efficiency such as distortion and noise models.
- The current node pattern can include only one wake-up phase. Thus, **more complex patterns** where the node wake-up several times cannot be modelled. Having more than one wake-up phase in the node pattern can be achieved. However, it requires changing how the node pattern is being constructed starting from the circuits' patterns as well as the circuit pattern synchronisation mechanism.
- Handling of the temporal aspect is implemented using phases. Nevertheless, the level of complexity increases when one phase includes sequential periods, making node

pattern synchronisation a hard task. For example, the acknowledgment phase in the Telecommunication Unit includes one waiting period as well as sending (or receiving) an acknowledgment frame, each one has its own specific duration and power level. To handle this kind of difficulties we think that **handling time** should be improved, and we are looking forward to introduce the concept of **event**. Unlike the phase, which is an energetic and temporal concept, the event will be a pure temporal representation to synchronise chained activities between nodes. Thus, events can create a complex activity and participate in defining the starting point of specific phase such as in the ACK phase as mentioned before. The event concept could be also used to implement non-periodic activities such as exchanging routing packets.

- **Simulator's GUI** must also be improved by adding more features, especially those related to the energetic perspective. For example, in the Topology level interface, an indication of battery status in a percentage form could be provide using a green bar positioned above each node. With this new indicator, we can visually reveal an existing link between energy storage and the topological node positions in order to identify, for example, strategic points in the adopted topology in terms of node lifetime.
- For the experimental part, a digital oscilloscope was used for the acquisition of the signal. This method is limited considering the wide range of the current. To improve the precision, it is necessary to implement a “smart” system such as the **adaptive multiplexer** presented in Chapter 4. Moreover, to validate our simulator, it will be interesting to investigate other telecommunication circuits to verify the existence of similar drawbacks than the selected one in the considered experimental scenario.

# APPENDIX 1: LIST OF ARTICLES RESULTED FROM THE FIRST PHASE OF METHODOLOGY IN CHAPTER 2

1 A. Abuarqoub, M. Hammoudeh, F. Alfayez and O. Aldabbas. "A survey on wireless sensor networks simulation tools and testbeds", Wireless Sensor Networks: Signal Processing and Communications Perspectives, pp. 283-302, 2007, ISBN: 9780470035573

2 I. Akyildiz, M. Vuran, O. Akan, and W. Su, "Wireless sensor networks: A survey revisited", Computer Networks Journal, pp. 1-39, 2006

3 S. Aswale, and V. Ghorpad, "Wireless Multimedia Sensor Network: A Survey on Multimedia Sensors", International Conference on Recent Trends in Information, Telecommunication and Computing, pp. 31-37, 2013

4 B. Ayyappan, and P. Kumar, "Security protocols in WSN: A survey", Third International Conference on Science Technology Engineering & Management (ICONSTEM), pp. 301-304, 2017, DOI: 10.1109/ICONSTEM.2017.8261297

5 L. Bergamini, C. Crociani, and A. Vitaletti, "Simulation vs real testbeds: a validation of WSN simulators", Department of Computer and System Sciences Antonio Ruberti Technical Reports, vol. 1, no. 3, 2009

6 R. Chéour, M. Jmal, O. Kanoun and M. Abid, "Evaluation of simulator tools and power-aware scheduling model for wireless sensor networks", IET Computers & Digital Techniques, vol. 11, no. 5, pp. 173-182, 2017, DOI: 10.1049/iet-cdt.2017.0003

7 P. Chhimwal, D. Rai and D. Rawat, "Comparison between different wireless sensor simulation tools", IOSR Journal of Electronics and Communication Engineering, vol. 5, no. 2, pp. 54-60, 2013

8 M. Chernyshev, Z. Baig, O. Bello and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds", IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1637-1647, 2018, DOI: 10.1109/JIOT.2017.2786639

9 W. Du, D. Navarro, F. Mieyeville and F. Gaffiot, "Towards a taxonomy of simulation tools for wireless sensor networks", 3rd International ICST Conference on Simulation Tools and Techniques, pp. 1-7, 2010, DOI: 10.4108/ICST.SIMUTOOLS2010.8659

10 E. Egea-López, J. Vales-Alonso, A. Martínez-Sala, P. Pavón-Mariño and J. García-Haro, "Simulation tools for wireless sensor networks", International symposium on performance evaluation of computer and telecommunication systems (SPECTS05), New Jersey, pp. 2-9, 2005

11 H. Fahmy, "Simulators and Emulators for WSNs", Wireless Sensor Networks – Concepts, Applications, Experimentation and Analysis in Signals and Communication Technology book series, pp. 381-491, 2016, DOI: 10.1007/978-981-10-0412-4_6

12 A. Förster, and A. Murphy, "A critical survey and guide to evaluating WSN routing protocols", First international workshop on networks of cooperating objects (CONET), Stockholm, 2010

13 R. Cheour, M. Jmal, O. Kanoun and M. Abid, "Evaluation of simulator tools and power-aware scheduling model for wireless sensor networks", IET Computers & Digital Techniques, vol. 11, no. 5, pp. 173-182, 2017, DOI: 10.1049/iet-cdt.2017.0003

14 J. Helkey, L. Holder, and B. Shirazi, "Comparison of simulators for assessing the ability to sustain wireless sensor networks using dynamic network reconfiguration", Sustainable Computing: Informatics and Systems, vol. 9, pp. 1-7, 2016, DOI: 10.1016/j.suscom.2016.01.003

15 M. Imran, A. Said and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks", International Symposium on Information Technology, Kuala Lumpur, pp. 897-902, 2010, DOI: 10.1109/ITSIM.2010.5561571

16 M. Jevtić, N. Zogović, and G. Dimić. "Evaluation of wireless sensor network simulators", 17th telecommunications forum (TELFOR 2009), Belgrade, pp. 1303-1306, 2009

17 P. Katkar and D. Ghorpade, "Comparative study of network simulator: NS2 and NS3", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 3, pp. 608-612, 2016

18 A. Kellner, K. Behrends and D. Hogrefe, "Simulation environments for wireless sensor networks", Technical Reports of the Institute of Computer Science at the Georg-August-Universitat Gottingen, no. IFI-TB-2010-04, 2010, ISSN: 1611-1044

19 M. Khan, B. Askwith, F. Bouhafs and M. Asim, "Limitations of Simulation Tools for Large-Scale Wireless Sensor Networks", IEEE Workshops of International Conference on

Advanced Information Networking and Applications, Singapore, pp. 820-825, 2011, DOI: 10.1109/WAINA.2011.59

20 A. Khan, S. Bilal and M. Othman, "A performance comparison of open source network simulators for wireless networks", IEEE International Conference on Control System, Computing and Engineering, Penang, pp. 34-38, 2012, DOI: 10.1109/ICCSCE.2012.6487111

21 M. Korkalainen, M. Sallinen, N. Kärkkäinen and P. Tukeva, "Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications", Fifth International Conference on Networking and Services, Valencia, pp. 102-106, 2009, DOI: 10.1109/ICNS.2009.75

22 M. Kuorilehto, M. Kohvakka, M. Hännikäinen and T. Hämäläinen, "High abstraction level design and implementation framework for wireless sensor networks", International Workshop on Embedded Computer Systems, Samos, vol. 3553, pp. 384-393, 2005, DOI: 10.1007/11512622_41

23 K. Lahmar, R. Cheour and M. Abid, "Wireless Sensor Networks: Trends, Power Consumption and Simulators", Sixth Asia Modelling Symposium, Bali, pp. 200-204, 2012, DOI: 10.1109/AMS.2012.50

24 S. Gupta, M. Ghonge, P. Thakare and P. Jawandhiya, "Open-source network simulation tools: An overview", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 2, no. 4, pp. 1629-1635, 2013

25 M. Mekni and B. Moulin, "A Survey on Sensor Webs Simulation Tools", Second International Conference on Sensor Technologies and Applications (SENSORCOMM), Cap Esterel, pp. 574-579, 2008, DOI: 10.1109/SENSORCOMM.2008.13

26 J. Yi, M. Kang and D. Noh, "SolarCastalia: Solar Energy Harvesting Wireless Sensor Network Simulator", International Journal of Distributed Sensor Networks, vol. 11, no. 6, pp. 1-10, 2015, DOI: 10.1155/2015/415174

27 I. Minakov, R. Passerone, A. Rizzardi, and S. Sicari. "A Comparative Study of Recent Wireless Sensor Network Simulators", ACM Transactions on Sensor Networks. vol. 12, no. 3, pp. 1-39, 2016, DOI: 10.1145/2903144

28 S. Mishra, and H. Thakkar, "Features of WSN and Data Aggregation techniques in WSN: A Survey", International Journal of Engineering and Innovative Technology (IJEIT), vol. 1, no. 4, pp. 264-273, 2012

29 D. Mishra, and R. Kumar, "Qualitative Analysis of wireless sensor network simulators", International Journal of Computer Applications, vol. 2, pp. 11-18, 2015

30 B. Musznicki, and P. Zwierzykowski, "Survey of simulators for wireless sensor networks", International Journal of Grid and Distributed Computing, vol. 5, no. 3, pp. 23-50, 2012

31 D. Navarro, F. Mieyeville, M. Galos, and L. Carrel. "Simulation of Hardware and Software in Heterogeneous Wireless Sensor Network", International Journal on Advances in Networks and Services, vol. 7, no. 1-2, pp. 97-107, 2014

32 E. Niewiadomska-Szynkiewicz and A. Sikora, "Performance Analysis of Energy Conservation Techniques for Wireless Sensor Networks", International Conference on Military Communications and Information Systems (ICMCIS), Budva, pp. 1-6, 2019, DOI: 10.1109/ICMCIS.2019.8842742

33 A. Pandey and R. Tripathi, "A survey on wireless sensor networks security", International Journal of Computer Applications, vol. 3, no. 2, pp. 43-49, 2010

34 G. Papadopoulos, K. Kritsis, A. Gallais, P. Chatzimisios and T. Noel, "Performance evaluation methods in ad hoc and wireless sensor networks: a literature study", IEEE Communications Magazine, vol. 54, no. 1, pp. 122-128, 2016, DOI: 10.1109/MCOM.2016.7378437

35 D. Pesic, Z. Radivojevic and M. Cvetanovic, "A survey and evaluation of free and open source simulators suitable for teaching courses in wireless sensor networks", 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, pp. 895-900, 2017, DOI: 10.23919/MIPRO.2017.7973549

36 K. Purbhoo and K. Khedo, "eMnSiM: An energy oriented model and simulator for wireless sensor networks", University of Mauritius Research Journal, vol. 15, no. 1, pp. 200-229, 2009

37 M. Rahman, A. Pakštas and F. Wang, "Network modelling and simulation tools", Simulation Modelling Practice and Theory, vol. 17, no. 6, pp. 1011-1031, 2009, DOI: 10.1016/j.simpat.2009.02.005

38 N. Sarkar and S. Halim, "A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations", Cyber Journals.

Special Issue, Journal of Selected Areas in Telecommunications (JSAT), vol. 2, no. 3, pp. 10-17, 2011

39 H. Sharma, A. Haque and Z. Jaffery, "Solar energy harvesting wireless sensor network nodes: A survey", Journal of Renewable and Sustainable Energy, vol. 10, no. 2, pp. 1-33, 2012, DOI: 10.1063/1.5006619

40 C. P. Singh, O. P. Vyas and M. K. Tiwari, "A Survey of Simulation in Sensor Networks", 2008 International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, 2008, pp. 867-872. DOI: 10.1109/CIMCA.2008.170.

41 A. Stetsko, M. Stehlik and V. Matyas, "Calibrating and Comparing Simulators for Wireless Sensor Networks", IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, pp. 733-738, 2011, DOI: 10.1109/MASS.2011.80

42 H. Sundani, H. Li, V. Devabhaktuni, M. Alam and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons", International Journal of Computer Networks, vol. 2, no. 5, pp. 249-265, 2011

43 I. Tomić, and J. McCann, "A survey of potential security issues in existing wireless sensor network protocols", IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1910-1923, 2017

44 O. Valle, C. Montez, G. Araujo, P. Portugal and F. Vasques, "Enhancing wireless sensor network simulators with a realistic battery discharge function", Information, vol. 16, no. 12, pp. 8767-8779, 2013

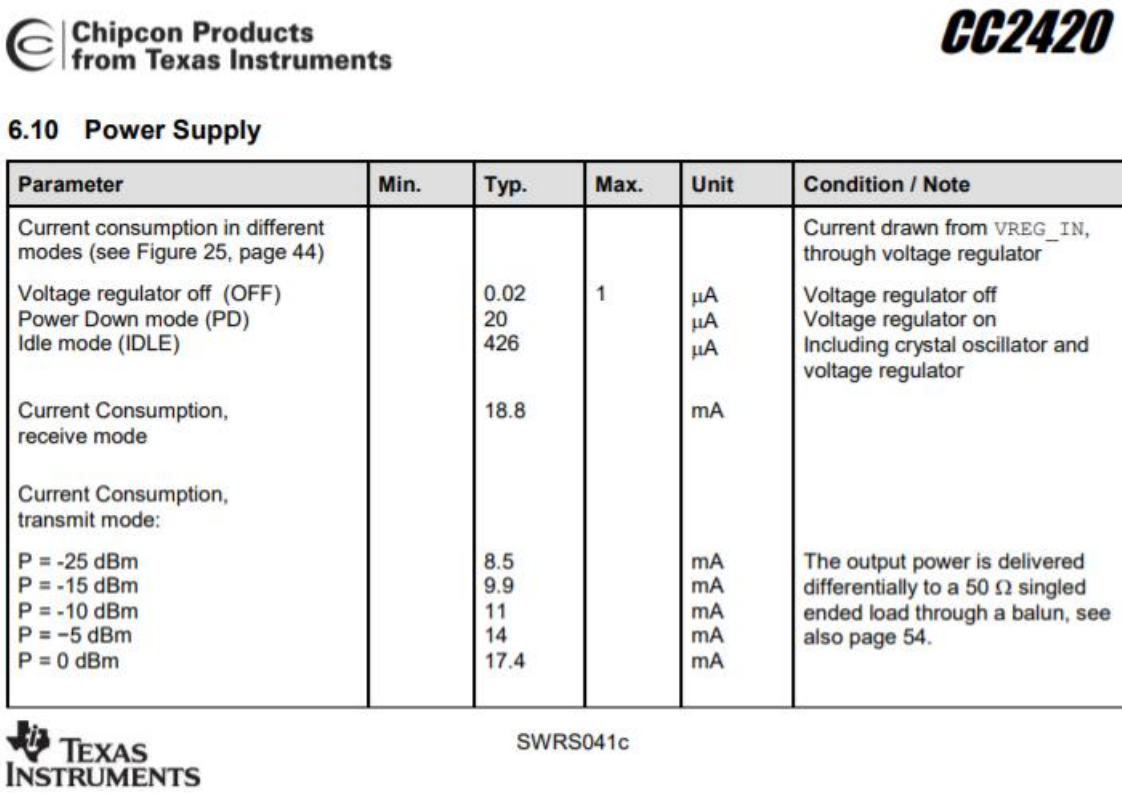
45 V. Vasanthi, "Simulators and emulators used for wireless sensor network", International Journal of Advanced Research in Computer and Communication Engineering, vol. 6, no. 1, pp. 171-175, 2017

46 S. Yadav and A. Chitra, "Wireless sensor networks-architectures, protocols, simulators and applications: a survey", International Journal of Electronics and Computer Science Engineering. vol. 1, no. 4, pp. 1941-1953, 2012

47 C. Zhu, C. Zheng, L. Shu and G. Han, "A survey on coverage and connectivity issues in wireless sensor networks", Journal of Network and Computer Applications, vol. 35, no. 2, pp. 619-632, 2012, DOI: 10.1016/j.jnca.2011.11.016

# APPENDIX 2: ELECTRICAL SPECIFICATIONS FOR THE CIRCUITS

## CC2420



Chipcon Products from Texas Instruments

CC2420

### 6.10 Power Supply

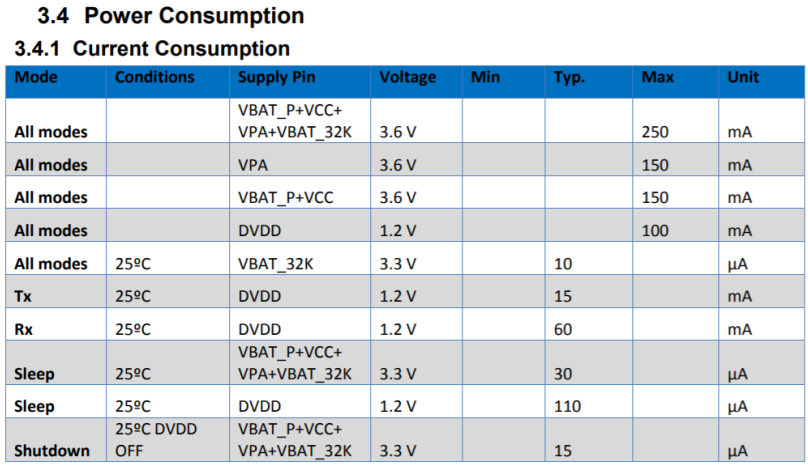
| Parameter | Min. | Typ. | Max. | Unit | Condition / Note |
|---|---|---|---|---|---|
| Current consumption in different modes (see Figure 25, page 44) | | | | | Current drawn from VREG_IN, through voltage regulator |
| Voltage regulator off (OFF)<br>Power Down mode (PD)<br>Idle mode (IDLE) | | 0.02<br>20<br>426 | 1 | µA<br>µA<br>µA | Voltage regulator off<br>Voltage regulator on<br>Including crystal oscillator and voltage regulator |
| Current Consumption, receive mode | | 18.8 | | mA | |
| Current Consumption, transmit mode: | | | | | |
| P = -25 dBm<br>P = -15 dBm<br>P = -10 dBm<br>P = −5 dBm<br>P = 0 dBm | | 8.5<br>9.9<br>11<br>14<br>17.4 | | mA<br>mA<br>mA<br>mA<br>mA | The output power is delivered differentially to a 50 Ω singled ended load through a balun, see also page 54. |

TEXAS INSTRUMENTS

SWRS041c

**Figure B-1: Electrical characteristics for CC2420**

## HDG204



### 3.4 Power Consumption

#### 3.4.1 Current Consumption

| Mode | Conditions | Supply Pin | Voltage | Min | Typ. | Max | Unit |
|---|---|---|---|---|---|---|---|
| All modes | | VBAT_P+VCC+ VPA+VBAT_32K | 3.6 V | | | 250 | mA |
| All modes | | VPA | 3.6 V | | | 150 | mA |
| All modes | | VBAT_P+VCC | 3.6 V | | | 150 | mA |
| All modes | | DVDD | 1.2 V | | | 100 | mA |
| All modes | 25ºC | VBAT_32K | 3.3 V | | 10 | | µA |
| Tx | 25ºC | DVDD | 1.2 V | | 15 | | mA |
| Rx | 25ºC | DVDD | 1.2 V | | 60 | | mA |
| Sleep | 25ºC | VBAT_P+VCC+ VPA+VBAT_32K | 3.3 V | | 30 | | µA |
| Sleep | 25ºC | DVDD | 1.2 V | | 110 | | µA |
| Shutdown | 25ºC DVDD OFF | VBAT_P+VCC+ VPA+VBAT_32K | 3.3 V | | 15 | | µA |

**Figure B-2: Electrical characteristics for HDG204**

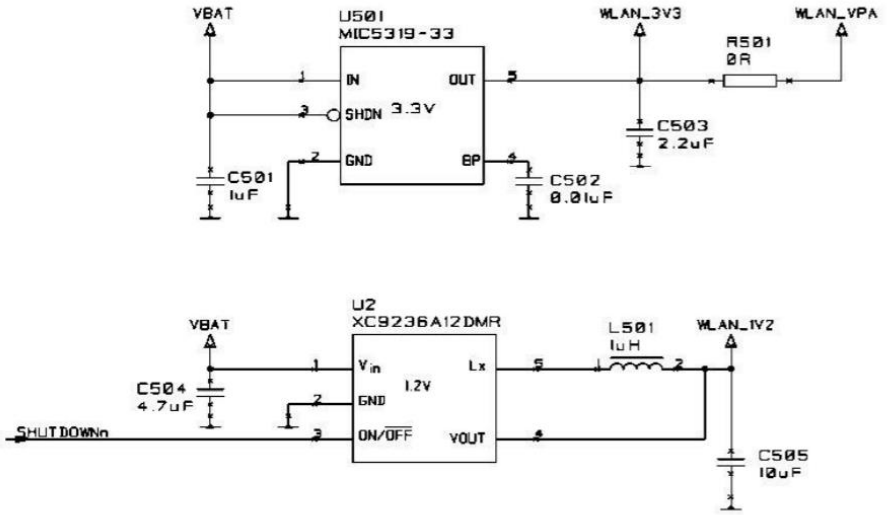



**Figure B-3: Regulator configuration for HDG204**

# MRF24J40

**EXAMPLE 3-1: INITIALIZING THE MRF24J40**

Example steps to initialize the MRF24J40:

1. SOFTRST (0x2A) = 0x07 – Perform a software Reset. The bits will be automatically cleared to '0' by hardware.
2. PACON2 (0x18) = 0x98 – Initialize FIFOEN = 1 and TXONTS = 0x6.
3. TXSTBL (0x2E) = 0x95 – Initialize RFSTBL = 0x9.
4. RFCON0 (0x200) = 0x03 – Initialize RFOPT = 0x03.
5. RFCON1 (0x201) = 0x01 – Initialize VCOOPT = 0x02.
6. RFCON2 (0x202) = 0x80 – Enable PLL (PLLEN = 1).
7. RFCON6 (0x206) = 0x90 – Initialize TXFIL = 1 and 20MRECVR = 1.
8. RFCON7 (0x207) = 0x80 – Initialize SLPCLKSEL = 0x2 (100 kHz Internal oscillator).
9. RFCON8 (0x208) = 0x10 – Initialize RFVCO = 1.
10. SLPCON1 (0x220) = 0x21 – Initialize $\overline{\text{CLKOUTEN}}$ = 1 and SLPCLKDIV = 0x01.

Configuration for nonbeacon-enabled devices (see **Section 3.8 "Beacon-Enabled and Nonbeacon-Enabled Networks"**):

11. BBREG2 (0x3A) = 0x80 – Set CCA mode to ED.
12. CCAEDTH = 0x60 – Set CCA ED threshold.
13. BBREG6 (0x3E) = 0x40 – Set appended RSSI value to RXFIFO.
14. Enable interrupts – See **Section 3.3 "Interrupts"**.
15. Set channel – See **Section 3.4 "Channel Selection"**.

**Note:** Maintain 0x200<3:0> = 0x03

16. Set transmitter power - See "REGISTER 2-62: RF CONTROL 3 REGISTER (ADDRESS: 0x203)".
17. RFCTL (0x36) = 0x04 – Reset RF state machine.
18. RFCTL (0x36) = 0x00.
19. Delay at least 192 µs.

**Figure B-4: Initialising of the MRF24j40**

**EXAMPLE 3-3: IMMEDIATE SLEEP AND WAKE**

The steps to prepare the MRF24J40 for immediate sleep and wake up on WAKE pin

Prepare WAKE pin:

1. WAKE pin = low
2. RXFLUSH (0x0D) = 0x60 – Enable WAKE pin and set polarity to active-high
3. WAKECON (0x22) = 0x80 – Enable Immediate Wake-up mode

Put to Sleep:

4. SOFTRST (0x2A) = 0x04 – Perform a Power Management Reset
5. SLPACK (0x35) = 0x80 – Put MRF24J40 to Sleep immediately

To Wake:

6. WAKE pin = high – Wake-up
7. RFCTL (0x36) = 0x04 - RF State Machine reset
8. RFCTL (0x36) = 0x00
9. Delay 2 ms to allow 20 MHz main oscillator time to stabilize before transmitting or receiving.

**Figure B-5: Immediate sleep and wakeup for MRF24j40**

**REGISTER 2-50: RFCTL: RF MODE CONTROL REGISTER (ADDRESS: 0x36)**

| W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| r | r | r | WAKECNT8 | WAKECNT7 | RFRST(2) | RFTXMODE | RFRXMODE |
| bit 7 | | | | | | | bit 0 |

| Legend: | r = reserved | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **Reserved:** Maintain as '0'

bit 4-3 **WAKECNT<8:7>:** Wake Count bits
Main oscillator (20 MHz) start-up timer counter bits. WAKECNT is a 9-bit value. WAKECNT<6:0> bits are located in SLPACK<6:0>. Units: Sleep clock (SLPCLK) period.(1) Default value: 0x00. Recommended value: 0x05F

bit 2 **RFRST:** RF State Machine Reset bit(2)
1 = Hold RF state machine in Reset
0 = Normal operation of RF state machine

bit 1 **RFTXMODE:** Forces RF Control State Machine to transmit State(3)

bit 0 **RFRXMODE:** Forces RF Control State Machine to receive State

**Note 1:** Sleep clock (SLPCLK) period depends on the Sleep clock selection (SLPCLKSEL) RFCON7<7:6> and Sleep clock divisor (SLPCLKDIV) SLPCON1<4:0>.

**2:** Perform RF Reset by setting RFRST = 1 and then RFRST = 0. Delay at least 192 µs after performing to allow RF circuitry to calibrate.

**3:** Recommended sequence RFCTL = 0x06 (reset mode) then RFCTL = 0x02 (transmit mode).

**Figure B-6: RFCTL register configuration**

**TABLE 5-2: CURRENT CONSUMPTION**

Typical Values: $T_A$ = 25°C, $V_{DD}$ = 3.3V

| Chip Mode | Condition | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Sleep | Sleep Clock Disabled | — | 2 | — | µA |
| TX | At maximum output power | — | 23 | — | mA |
| RX | — | — | 19 | — | mA |

**Figure B-7: Electrical characteristics for MRF24j40**

# PIC18F4620



**26.2 DC Characteristics: Power-Down and Supply Current**
**PIC18F2525/2620/4525/4620 (Industrial)**
**PIC18LF2525/2620/4525/4620 (Industrial)**

| | | | | | | |
|---|---|---|---|---|---|---|
| **PIC18LF2525/2620/4525/4620** (Industrial) | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2525/2620/4525/4620** (Industrial, Extended) | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | |
| | **Power-Down Current (IPD)(1)** | | | | | |
| | PIC18LFX525/X620 | 0.1 | 0.5 | µA | -40°C | VDD = 2.0V (**Sleep** mode) |
| | | 0.1 | 0.5 | µA | +25°C | |
| | | 0.2 | 2.5 | µA | +85°C | |
| | PIC18LFX525/X620 | 0.1 | 0.7 | µA | -40°C | VDD = 3.0V (**Sleep** mode) |
| | | 0.1 | 0.7 | µA | +25°C | |
| | | 0.3 | 3.5 | µA | +85°C | |
| | All devices | 0.1 | 1.0 | µA | -40°C | VDD = 5.0V (**Sleep** mode) |
| | | 0.2 | 1.0 | µA | +25°C | |
| | | 0.7 | 10 | µA | +85°C | |
| | Extended devices only | 10 | 100 | µA | +125°C | |

**Legend:** Shading of rows is to assist in readability of the table.

**Figure B-8: Electrical characteristics for PIC18F4620**

# TMP102



**Electrical Characteristics (continued)**

At $T_A$ = 25°C and $V_S$ = 1.4 to 3.6 V, unless otherwise noted.

| PARAMETER | | TEST CONDITIONS | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|
| **POWER SUPPLY** | | | | | | |
| | Operating supply range | | +1.4 | | +3.6 | V |
| $I_Q$ | Average quiescent current | Serial bus inactive, CR1 = 1, CR0 = 0 (default) | | 7 | 10 | µA |
| | | Serial bus active, SCL frequency = 400 kHz | | 15 | | |
| | | Serial bus active, SCL frequency = 3.4 MHz | | 85 | | |
| $I_{SD}$ | Shutdown current | Serial bus inactive | | 0.5 | 1 | µA |
| | | Serial bus active, SCL frequency = 400 kHz | | 10 | | |
| | | Serial bus active, SCL frequency = 3.4 MHz | | 80 | | |
| **TEMPERATURE** | | | | | | |
| | Specified range | | –40 | | 125 | °C |
| | Operating range | | –55 | | 150 | °C |

**Figure B-9: Electrical characteristics for TMP102**

At $T_A$ = 25°C and $V_S$ = 1.4 to 3.6 V, unless otherwise noted.

| PARAMETER | | | TEST CONDITIONS | MIN | TYP | MAX | UNIT |
|---|---|---|---|---|---|---|---|
| **TEMPERATURE INPUT** | | | | | | | |
| | Range | | | –40 | | 125 | °C |
| | Accuracy (temperature error) | | –25°C to 85°C | | ±0.5 | ±2 | °C |
| | | | –40°C to 125°C | | ±1 | ±3 | |
| | vs supply | | | | 0.2 | 0.5 | °C/V |
| | Resolution | | | | 0.0625 | | °C |
| **DIGITAL INPUT/OUTPUT** | | | | | | | |
| | Input capacitance | | | | 3 | | pF |
| $V_{IH}$ | Input logic high | | | 0.7 × (V+) | | 3.6 | V |
| $V_{IL}$ | Input logic low | | | –0.5 | | 0.3 × (V+) | V |
| $I_{IN}$ | Input current | | 0 < $V_{IN}$ < 3.6 V | | | 1 | µA |
| $V_{OL}$ | Output logic | SDA | V+ > 2 V, $I_{OL}$ = 3 mA | 0 | | 0.4 | V |
| | | | V+ < 2 V, $I_{OL}$ = 3 mA | 0 | | 0.2 × (V+) | |
| | | ALERT | V+ > 2 V, $I_{OL}$ = 3 mA | 0 | | 0.4 | |
| | | | V+ < 2 V, $I_{OL}$ = 3 mA | 0 | | 0.2 × (V+) | |
| | Resolution | | | | 12 | | Bit |
| | Conversion time | | | | 26 | 35 | ms |
| | Conversion modes | | CR1 = 0, CR0 = 0 | | 0.25 | | Conv/s |
| | | | CR1 = 0, CR0 = 1 | | 1 | | |
| | | | CR1 = 1, CR0 = 0 (default) | | 4 | | |
| | | | CR1 = 1, CR0 = 1 | | 8 | | |
| | Timeout time | | | | 30 | 40 | ms |

**Figure B-10: Conversion time for TMP102**

# REFERENCES

**A**

**[ABD13]** R. Abdulla and W. Ismail, "Survey of WSN technology based reliable and efficient active RFID", IEEE 11th Malaysia International Conference on Communications (MICC), Kuala Lumpur, pp. 116-121, 2013, DOI: 10.1109/MICC.2013.6805810

**[ABU16]** A. Abuarqoub, M. Hammoudeh, F. Alfayez and O. Aldabbas. "A survey on wireless sensor networks simulation tools and testbeds", Wireless Sensor Networks: Signal Processing and Communications Perspectives, pp. 283-302, 2007, ISBN: 9780470035573

**[ADI09]** R. Mallikarjuna, A. Kumar, D. Janakiram and G. Kumar, "Wireless sensor network operating systems: a survey", International Journal of Sensor Networks, vol. 5, no. 4, pp. 236-255, 2009, DOI: 10.1504/IJSNET.2009.027631

**[AGU16]** E. Aguirre, P. Lopez-Iturri, L. Azpilicueta, J. Astrain, J. Villadangos, D. Santesteban and F. Falconem, "Implementation and Analysis of a Wireless Sensor Network-Based Pet Location Monitoring System for Domestic Scenarios", Sensors, vol. 16, no. 9, p. 1384, 2016, DOI: 10.3390/s16091384

**[AHM10]** N. Ahmed, M. Rutten, T. Bessell, S. Kanhere, N. Gordon et al., "Detection and Tracking Using Particle-Filter-Based Wireless Sensor Networks". IEEE Transactions on Mobile Computing, vol. 9, no. 9, pp. 1332-1345, 2010, DOI: 10.1109/TMC.2010.83

**[AKK05]** K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks", Ad Hoc Networks, vol. 3, no. 3, pp. 325-349, 2005, DOI: 10.1016/j.adhoc.2003.09.010

**[AKY02]** I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey", Computer Networks, vol. 38, no. 4, pp. 393-422, 2002. DOI: 10.1016/S1389-1286(01)00302-4

**[AKY05]** I. Akyildiz, D. Pompili and T. Melodia, "Underwater acoustic sensor networks: research challenges", Ad Hoc Networks, vol. 3, no. 3, pp. 257-279, 2005, DOI: 10.1016/j.adhoc.2005.01.004

**[AKY10]** I. Akyildiz and M. Can Vuran, "Wireless Sensor Networks", WILEY, New York, 2010. ISBN: 047003601X

**[AZZ17]** T. Azzabi, H. Farhat and N. Sahli, "A survey on wireless sensor networks security issues and military specificities", International Conference on Advanced Systems and Electric Technologies (IC ASET), Hammamet, pp. 66-72, 2017, DOI: 10.1109/ASET.2017.7983668

**[AZM14]** N. Azmi, L. Kamarudin, M. Mahmuddin, A. Zakaria, A. Shakaff et al., "Interference issues and mitigation method in WSN 2.4GHz ISM band: A survey", 2nd International Conference on Electronic Design (ICED), Penang, pp. 403-408, 2014, DOI: 10.1109/ICED.2014.7015839

**B**

**[BAN10]** J. Banks, "Discrete-event system simulation", Pearson, 2010. ISBN: 0138150370

**[BAT37]** RS Pro 3.7V Li-Po Rechargeable Battery, 2000mAh, RS-PRO, 125-1266 [online] https://uk.rs-online.com/web/p/speciality-size-rechargeable-batteries/1251266/

**[BER12]** I. Beretta, F. Rincon, N. Khaled, P. Grassi, V. Rana and D. Atienza, "Design exploration of energy-performance trade-offs for wireless sensor networks", Design Automation Conference (DAC), San Francisco, pp. 1043-1048, 2012, DOI: 10.1145/2228360.2228549

**[BHA05]** S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose et. al., "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms", Mobile Networks and Applications, vol. 10, no. 4, pp. 563-579, 2005. DOI: 10.1007/s11036-005-1567-8.

**[BLU10]** Bluetooth, S. I. G., "Bluetooth core specification version 4.0", Specification of the Bluetooth System, 2010

**[BLU16]** Bluetooth, S. I. G., "Bluetooth core specification version 5.0", Specification of the Bluetooth System, 2016

**[BOU18]** T. Bouguera, J. Diouris, J. Chaillout, R. Jaouadi and G. Andrieux, "Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN", Sensors, vol. 18, no. 7, p. 2014, 2018, DOI: 10.3390/s18072104

**[BRA89]** R. Braden, "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, 1989, DOI: 10.17487/RFC1122

**[BRE00]** S. Bajaj, L. Breslau, D. Estrin, K. Fall, S. Floyd et al., "Advances in network simulation", IEEE Computer, vol. 33, no. 5, pp. 59-67, 2000, DOI: 10.1109/2.841785

**[BRI05]** D. Brinza, G. Calinescu, S. Tongngam and A. Zelikovsky, "Energy-efficient continuous and event-driven monitoring", IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, Washington, pp. 1-3, 2005, DOI: 10.1109/MAHSS.2005.1542792

**[BRI09]** D. Bri, M. Garcia, J. Lloret and P. Dini, "Real Deployments of Wireless Sensor Networks", Third International Conference on Sensor Technologies and Applications (SENSORCOMM), Athens, pp. 415-423, 2009, DOI: 10.1109/SENSORCOMM.2009.69

**[BRU16]** J. Brusey, J. Kemp, E. Gaura, R. Wilkins and M. Allen, "Energy Profiling in Practical Sensor Networks: Identifying Hidden Consumers", IEEE Sensors Journal, vol. 16, no. 15, pp. 6072-6080, 2016, DOI: 10.1109/JSEN.2016.2570420

**C**

**[CAT15]** L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, et al., "An IoT-Aware Architecture for Smart Healthcare Systems", IEEE Internet of Things Journal, vol. 2, no. 6, pp. 515-526, 2015, DOI: 10.1109/JIOT.2015.2417684

**[CHA08]** G. Chalivendra, R. Srinivasan and N. Murthy, "FPGA based re-configurable wireless sensor network protocol", International Conference on Electronic Design, Penang, pp. 1-4, 2008, DOI: 10.1109/ICED.2008.4786652.

**[CHA09]** W. Charfi, M. Masmoudi and F. Derbel, "A layered model for wireless sensor networks", 6th International Multi-Conference on Systems, Signals and Devices, Djerba, pp. 1-5, 2009, DOI: 10.1109/SSD.2009.4956693

**[CHE05]** S. Cheekiralla and D. Engels, "A functional taxonomy of wireless sensor network devices", 2nd International Conference on Broadband Networks, Boston, vol. 2, pp. 949-956, 2005, DOI: 10.1109/ICBN.2005.1589707

**[CHE06]** W. Cheng, W. Chung-Yi and F. Kai-Ten, "Power-Controlled Hybrid Multicast Routing Protocol for Mobile Ad Hoc Networks", IEEE 63rd Vehicular Technology Conference, Melbourne, pp. 1087-1091, 2006, DOI: 10.1109/VETECS.2006.1683002

**[CHE17]** R. Cheour, M. Jmal, O. Kanoun and M. Abid, "Evaluation of simulator tools and power-aware scheduling model for wireless sensor networks", IET Computers & Digital Techniques, vol. 11, no. 5, pp. 173-182, 2017, DOI: 10.1049/iet-cdt.2017.0003

**[CHH13]** P. Chhimwal, D. Rai and D. Rawat, "Comparison between different wireless sensor simulation tools", IOSR Journal of Electronics and Communication Engineering, vol. 5, no. 2, pp. 54-60, 2013

**[CHI05]** S. Chien-Chung, C. Ko-Ming, K. Yau-Hwang and H. Mong-Fong, "The new intrusion prevention and detection approaches for clustering-based sensor networks [wireless sensor networks]", IEEE Wireless Communications and Networking Conference, New Orleans, vol. 4, pp. 1927-1932, 2005, DOI: 10.1109/WCNC.2005.1424814

**[CHI10]** O. Chipara, C. Lu, T. Bailey and G. Roman, "Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit", 8th ACM Conference on Embedded Networked Sensor Systems, New York, pp. 155-168, 2010, DOI: 10.1145/1869983.1869999

**[CHI12]** T. Chien, H. Chan and T. Huu, "A Comparative Study on Hardware Platforms for Wireless Sensor Networks", International Journal on Advanced Science, Engineering and Information Technology, vol. 2, no. 1, pp. 70-74, 2012, DOI: 10.18517/ijaseit.2.1.157

**[CHI18]** M. Chernyshev, Z. Baig, O. Bello and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds", IEEE Internet of Things Journal, vol. 5, no. 3, pp. 1637-1647, 2018, DOI: 10.1109/JIOT.2017.2786639

**[COM00]** D. Comer, "Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architectures", Fourth Edition. Prentice Hall PTR, 2000. ISBN: 0130183806

**[COU11]** F. Cuomo, A. Abbagnale and E. Cipollone, "Cross-layer network formation for energy-efficient IEEE 802.15.4/ZigBee Wireless Sensor Networks", Ad Hoc Networks, vol. 11, no. 2, pp. 672-686, 2013, DOI: 10.1016/j.adhoc.2011.11.006

**D**

**[DAW12]** S. Dawson-Haggerty, S. Lanzisera, J. Taneja, R. Brown, and D. Culler, "@scale: insights from a large, long-lived appliance energy WSN", 11th international conference on Information Processing in Sensor Networks (IPSN), Beijing, pp. 37–48, 2012, DOI: 10.1145/2185677.2185683

**[DEE17]** S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, 2017, DOI: 10.17487/RFC8200

**[DER12]** A. Derhab, F. Ounini and B. Remli, "MOB-TOSSIM: An Extension Framework for TOSSIM Simulator to Support Mobility in Wireless Sensor and Actuator Networks", IEEE 8th International Conference on Distributed Computing in Sensor Systems, Hangzhou, pp. 300-305, 2012, DOI: 10.1109/DCOSS.2012.34

**[DIJ59]** E. Dijkstra, "A note on two problems in connexion with graphs", Numerische Mathematik, vol. 1, pp. 269–271, 1959, DOI: 10.1007/BF01386390.S2CID123284777

**[DOM16]** J. Dominguez-Morales, A. Rios-Navarro, M. Dominguez-Morales, R. Tapiador-Morales, D. Gutierrez-Galan et al., "Wireless Sensor Network for Wildlife Tracking and Behavior Classification of Animals in Doñana", IEEE Communications Letters, vol. 20, no. 12, pp. 2534-2537, 2016, DOI: 10.1109/LCOMM.2016.2612652

**[DUN04]** A. Dunkels, B. Gronvall and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors", 29th Annual IEEE International Conference on Local Computer Networks, Tampa, pp. 455-462, 2004, DOI: 10.1109/LCN.2004.38

**[DUW10]** W. Du, D. Navarro, F. Mieyeville and F. Gaffiot, "Towards a taxonomy of simulation tools for wireless sensor networks", 3rd International ICST Conference on Simulation Tools and Techniques, pp. 1-7, 2010, DOI: 10.4108/ICST.SIMUTOOLS2010.8659

**[DUN07]** A. Dunkels, F. Österlind, N. Tsiftes and Z. He, "Software-based on-line energy estimation for sensor nodes", 4th workshop on Embedded networked sensors, pp. 28–32, 2007, DOI: 10.1145/1278972.1278979

**[DUR12]** M. Đurišić, Z. Tafa, G. Dimić and V. Milutinović, "A survey of military applications of wireless sensor networks", Mediterranean Conference on Embedded Computing (MECO), Bar, pp. 196-199, 2012

**E**

**[EGA05]** E. Egea-López, J. Vales-Alonso, A. Martínez-Sala, P. Pavón-Mariño and J. García-Haro, "Simulation tools for wireless sensor networks", International symposium on performance evaluation of computer and telecommunication systems (SPECTS05), New Jersey, pp. 2-9, 2005

**[EHS11]** H. Ehsan and F. Khan, "Query Processing Systems for Wireless Sensor Networks", Ubiquitous Computing and Multimedia Applications (UCMA), Daejeon, vol. 150, pp. 273-282, 2011, DOI: 10.1007/978-3-642-20975-8_30

**F**

**[FAH16]** H. Fahmy, "Simulators and Emulators for WSNs", Wireless Sensor Networks – Concepts, Applications, Experimentation and Analysis in Signals and Communication Technology book series, pp. 381-491, 2016, DOI: 10.1007/978-981-10-0412-4_6

**[FAL11]** The ns Manual, The VINT Project, 2011 [online] https://www.isi.edu/nsnam/ns/doc/index.html

**[FAR11]** M. Farooq and T. Kunz, "Operating Systems for Wireless Sensor Networks: A Survey", Sensors, vol. 11, no. 6, pp. 5900-5930, 2011, DOI: 10.3390/s110605900

**[FAR19]** K. Farhan, F. Abdel-Fattah, F. Altarawneh and M. Lafi, "Survey Paper on Multicast Routing in Mobile Ad-hoc Networks", IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, pp. 449-452, 2019, DOI: 10.1109/JEEIT.2019.8717477

**[FIS15]** R. Fisher, L. Ledwaba, G. Hancke and G. Kruger, "Open Hardware: A Role to Play in Wireless Sensor Networks?", Sensors, vol. 15, no. 3. pp. 6818-6844, 2015, DOI: 10.3390/s150306818

**[FUL06]** V. Fuller and T. Li., "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, 2006, DOI: 10.17487/RFC4632

**G**

**[GAR12]** K. Garg, A. Förster, D. Puccinelli, S. Giordano, "Towards Realistic and Credible Wireless Sensor Network Evaluation". Ad Hoc Networks (ADHOCNETS), vol. 89, pp. 49-64, 2012, DOI: 10.1007/978-3-642-29096-1_4

**[GAO05]** T. Gao, D. Greenspan, M. Welsh, R. Juang and A. Alm, "Vital Signs Monitoring and Patient Tracking Over a Wireless Network", IEEE Engineering in Medicine and Biology 27th Annual Conference, Shanghai, pp. 102-105, 2005, DOI: 10.1109/IEMBS.2005.1616352

**[GAS11]** O. Gasser, "TCP/IP communication in a WSN". Sensor Nodes–Operation, Network and Application. vol. 75, pp. 75-82, 2011, DOI: 10.2313/NET-2011-07-1_11

**[GEN10]** Z. Gengzhong and L. Qiumei, "A Survey on Topology Control in Wireless Sensor Networks", Second International Conference on Future Networks, Sanya, pp. 376-380, 2010, DOI: 10.1109/ICFN.2010.31

**[GIO05]** A. Giovanardi and G. Mazzini, "Ad hoc routing protocols: emulation vs simulation", 2nd International Symposium on Wireless Communication Systems, Siena, pp. 140-144, 2005, DOI: 10.1109/ISWCS.2005.1547673

**[GOD17]** D. Godoy, E. Sosa, R. Díaz Redondo and H. Bareiro, "WebShawn, simulating wireless sensors networks from the web", IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, pp. 190-195, 2017, DOI: 10.1109/WiMOB.2017.8115829

**[GOL02]** A. Goldsmith and S. Wicker, "Design challenges for energy-constrained ad hoc wireless networks", IEEE Wireless Communications, vol. 9, no. 4, pp. 8-27, 2002, DOI: 10.1109/MWC.2002.1028874

**H**

**[HAA11]** J. Haase, J. Molina and D. Dietrich, "Power-Aware System Design of Wireless Sensor Networks: Power Estimation and Power Profiling Strategies", IEEE

Transactions on Industrial Informatics, vol. 7, no. 4, pp. 601-613, 2011, DOI: 10.1109/TII.2011.2166793

**[HAN05]** C. Han, R. Kumar, R. Shea, E. Kohler and M. Srivastava, "A dynamic operating system for sensor nodes", 3rd international conference on Mobile systems, applications, and services (MobiSys), New York, pp. 163-176, 2005, DOI: 10.1145/1067170.1067188

**[HAN08]** B. Han, D. Zhang and T. Yang, "Energy consumption analysis and energy management strategy for sensor node", International Conference on Information and Automation, Changsha, pp. 211-214, 2008, DOI: 10.1109/ICINFA.2008.4607998

**[HAO12]** J. Hao, B. Zhang and H. Mouftah, "Routing protocols for duty cycled wireless sensor networks: A survey", IEEE Communications Magazine, vol. 50, no. 12, pp. 116-123, 2012, DOI: 10.1109/MCOM.2012.6384460

**[HDG11]** "HDG204-WiFi 802.11b+g System in Package", H&D Wirless, Rev. PA5 02/2011, 2011.

**[HEA08]** M. Healy, T. Newe and E. Lewis, "Wireless Sensor Node hardware: A review", IEEE Sensors, Lecce, pp. 621-624, 2008, DOI: 10.1109/ICSENS.2008.4716517

**[HED88]** C. Hedrick, "Routing Information Protocol", RFC 1058, 1988, DOI: 10.17487/RFC1058

**[HEF12]** M. Hefeida, M. Shen, A. Kshemkalyani and A. Khokhar, "Cross-layer protocols for WSNs: A simple design and simulation paradigm", 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, pp. 844-849, 2012, DOI: 10.1109/IWCMC.2012.6314314

**[HEL16]** J. Helkey, L. Holder, and B. Shirazi, "Comparison of simulators for assessing the ability to sustain wireless sensor networks using dynamic network reconfiguration", Sustainable Computing: Informatics and Systems, vol. 9, pp. 1-7, 2016, DOI: 10.1016/j.suscom.2016.01.003

**I**

**[IDE14a]** IDEA1 Documentation [online] http://idea1inl.free.fr/IDEA1/doc.html, last visited: 9 September 2020.

**[IDE14b]** IDEA1 release [online] http://idea1inl.free.fr/IDEA1/distrib/IDEA1.tgz, last visited: 9 September 2020.

**[IEE00]** "IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band", IEEE Std 802.11b-1999, pp. 1-96, 2000, DOI: 10.1109/IEEESTD.2000.90914

**[IEE03]** "IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Further Higher Data Rate Extension in the 2.4 GHz Band", IEEE Std 802.11g-2003 (Amendment to IEEE Std 802.11, 1999 Edn. (Reaff 2003) as amended by IEEE Stds 802.11a-1999, 802.11b-1999, 802.11b-1999/Cor 1-2001, and 802.11d-2001), pp. 1-104, 2003, DOI: 10.1109/IEEESTD.2003.94282

**[IEE05]** "IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements -- Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)", IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002), pp. 1-700, 2005, DOI: 10.1109/IEEESTD.2005.96290

**[IEE06]** "IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements -- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)", IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003), pp. 1-320, 2006, DOI: 10.1109/IEEESTD.2006.232110

**[IEE07a]** "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats", IEEE Std 1451.5-2007, pp. 1-236, 2007, DOI: 10.1109/IEEESTD.2007.4346346

**[IEE07b]** "IEEE Standard for Information Technology -- Telecommunications and Information Exchange Between Systems -- Local and Metropolitan Area

Networks -- Specific Requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999), pp. 1-1076, 2007, DOI: 10.1109/IEEESTD.2007.373646

**[IEE09]** "IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput", IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009), pp. 1-565, 2009, DOI: 10.1109/IEEESTD.2009.5307322

**[IEE11]** "IEEE Standard for Local and metropolitan area networks -- Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)", IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006), pp. 1-314, 2011, DOI: 10.1109/IEEESTD.2011.6012487

**[IEE14]** "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE Std 802-2014 (Revision to IEEE Std 802-2001), pp. 1-74, 2014, DOI: 10.1109/IEEESTD.2014.6847097

**[IEE16a]** "IEEE Standard for Information technology -- Telecommunications and information exchange between systems Local and metropolitan area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012), pp. 1-3534, 2016, DOI: 10.1109/IEEESTD.2016.7786995

**[IEE16b]** "IEEE Standard for High Data Rate Wireless Multi-Media Networks", IEEE Std 802.15.3-2016 (Revision of IEEE Std 802.15.3-2003), pp. 1-510, 2016, DOI: 10.1109/IEEESTD.2016.7524656

**[IEE16c]** "IEEE Standard for Low-Rate Wireless Networks", IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011), pp. 1-709, 2016, DOI: 10.1109/IEEESTD.2016.7460875

**[IEE17]** "IEEE Standard for Information technology -- Telecommunications and information exchange between systems -- Local and metropolitan area networks

-- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation", IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016, as amended by IEEE Std 802.11ai-2016), pp. 1-594, 2017, DOI: 10.1109/IEEESTD.2017.7920364

**[IEE97]** "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Std 802.11-1997, pp. 1-445, 1997, DOI: 10.1109/IEEESTD.1997.85951

**[IEE19]** "IEEE Draft Standard for Information Technology -- Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks -- Specific Requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment: Wake-up radio operation", IEEE P802.11ba/D2.0, (amendment to IEEE P802.11REVmd/D2.0, as amended by IEEE P802.11ax/D3.3, IEEE P802.11ay/D2.2, and IEEE P802.11az/D0.6), pp. 1-130, 2019.

**[IEE98]** "IEEE Standard for Information Technology -- Telecommunications and information exchange between systems -- Local and Metropolitan Area networks -- Specific requirements -- Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", ANSI/IEEE Std 802.11, 1999 Edition (R2003), pp. 1-512, 1998, DOI: 10.1109/IEEESTD.1998.8684613

**[IMR10]** M. Imran, A. Said and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks", International Symposium on Information Technology, Kuala Lumpur, pp. 897-902, 2010, DOI: 10.1109/ITSIM.2010.5561571

**[ISS09]** T. Issariyakul and E. Hossain, "Introduction to Network Simulator 2 (NS2)", Springer, Boston, DOI: 10.1007/978-0-387-71760-9_2, ISBN: 9780387717609

**J**

**[JAC01]** P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyume et al., "Optimized link state routing protocol for ad hoc networks", IEEE International Multi Topic Conference, Lahore, pp. 62-68, 2001, DOI: 10.1109/INMIC.2001.995315

**[JAY07]** L. Jayashree and S. Arumugam, "Design Challenges for Optimizing the Performance of Energy Constrained Wireless Sensor Networks", International Conference on Signal Processing, Communications and Networking, Chennai, pp. 537-540, 2007, DOI: 10.1109/ICSCN.2007.350659

**[JEV09]** M. Jevtić, N. Zogović, and G. Dimić. "Evaluation of wireless sensor network simulators", 17th telecommunications forum (TELFOR 2009), Belgrade, pp. 1303-1306, 2009

**[JIA14]** J. Li, H. Zhou, D. Zuo, K. Hou, H. Xie et al., "Energy consumption evaluation for wireless sensor network nodes based on queuing Petri net", International Journal of Distributed Sensor Networks, vol. 10, no. 4 ,pp. 1-11, 2014, DOI: 10.1155/2014/262848

**[JOH09]** M. Johnson, M. Healy, P. de Ven, M. Hayes, J. Nelson et al., "A comparative review of wireless sensor network mote technologies", IEEE Sensors, Christchurch, pp. 1439-1442, 2009, DOI: 10.1109/ICSENS.2009.5398442

**[JUN12]** J. Deokwoo, T .Teixeira, and Andreas Savvides, "Sensor node lifetime analysis: Models and tools", ACM Transactions on Sensor Networks, vol. 5, no. 1, pp. 1-33, 2009, DOI: 10.1145/1464420.1464423

**K**

**[KAT16]** P. Katkar and D. Ghorpade, "Comparative study of network simulator: NS2 and NS3", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 3, pp. 608-612, 2016

**[KAW10]** S. Kawamura and M. Kawashima, "A Recommendation for IPv6 Address Text Representation", RFC 5952, 2010, DOI: 10.17487/RFC5952

**[KEL10]** A. Kellner, K. Behrends and D. Hogrefe, "Simulation environments for wireless sensor networks", Technical Reports of the Institute of Computer Science at the Georg-August-Universitat Gottingen, no. IFI-TB-2010-04, 2010, ISSN: 1611-1044

**[KHA11]** M. Khan, B. Askwith, F. Bouhafs and M. Asim, "Limitations of Simulation Tools for Large-Scale Wireless Sensor Networks", IEEE Workshops of International Conference on Advanced Information Networking and Applications, Singapore, pp. 820-825, 2011, DOI: 10.1109/WAINA.2011.59

**[KHA12]** A. Khan, S. Bilal and M. Othman, "A performance comparison of open source network simulators for wireless networks", IEEE International Conference on Control System, Computing and Engineering, Penang, pp. 34-38, 2012, DOI: 10.1109/ICCSCE.2012.6487111

**[KHA13]** M. Khan, E. Felemban, S. Qaisar and S. Ali, "Performance Analysis on Packet Delivery Ratio and End-to-End Delay of Different Network Topologies in Wireless Sensor Networks (WSNs)", IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks, Dalian, pp. 324-329, 2013, DOI: 10.1109/MSN.2013.74

**[KHA15]** J. Khan, H. Qureshi, A. Iqbal, "Energy management in Wireless Sensor Networks: A survey", Computers & Electrical Engineering, vol. 41, pp. 159-176, 2015, DOI: 10.1016/j.compeleceng.2014.06.009

**[KIM09]** J. Kim, J. Lee and S. Kim, "An Enhanced Cross-Layer Protocol for Energy Efficiency in Wireless Sensor Networks", Third International Conference on Sensor Technologies and Applications (SENSORCOMM), Athens, pp. 657-664, 2009, DOI: 10.1109/SENSORCOMM.2009.106

**[KOR09]** M. Korkalainen, M. Sallinen, N. Kärkkäinen and P. Tukeva, "Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications", Fifth International Conference on Networking and Services, Valencia, pp. 102-106, 2009, DOI: 10.1109/ICNS.2009.75

**[KOU12]** A. Kouche, "Towards a wireless sensor network platform for the Internet of Things: Sprouts WSN platform", IEEE International Conference on Communications (ICC), Ottawa, pp. 632-636, 2012, DOI: 10.1109/ICC.2012.6364196

**[KUO05]** M. Kuorilehto, M. Kohvakka, M. Hännikäinen and T. Hämäläinen, "High abstraction level design and implementation framework for wireless sensor networks", International Workshop on Embedded Computer Systems, Samos, vol. 3553, pp. 384-393,2005, DOI: 10.1007/11512622_41

**[KUO06]** M. Kuorilehto, J. Suhonen, M. Kohvakka, M. Hannikainen and T. Hamalainen, "Experimenting TCP/IP for Low-Power Wireless Sensor Networks", IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications, Helsinki, pp. 1-6, 2006, DOI: 10.1109/PIMRC.2006.254157

**[KUS07a]** B. Kusý, G. Balogh, J. Sallai, A. Lédeczi and M. Maróti, "inTrack: High Precision Tracking of Mobile Sensor Nodes. European Conference on Wireless Sensor Networks (EWSN), Delft, vol. 4373, pp. 51-66, 2007, DOI: 10.1007/978-3-540-69830-2_4

**[KUS07b]** N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, 2007, DOI: 10.17487/RFC4919

**L**

**[LAH12]** K. Lahmar, R. Cheour and M. Abid, "Wireless Sensor Networks: Trends, Power Consumption and Simulators", Sixth Asia Modelling Symposium, Bali, pp. 200-204, 2012, DOI: 10.1109/AMS.2012.50

**[LEE09]** S. Lee, S. Lee, H. Song and H. Lee, "Wireless sensor network design for tactical military applications: Remote large-scale environments", IEEE Military Communications Conference (MILCOM), Boston, pp. 1-7, 2009, DOI: 10.1109/MILCOM.2009.5379900

**[LEE12]** C. Lee and C. Yang, "Distributed Energy-Efficient Topology Control Algorithm in Home M2M Networks", International Journal of Distributed Sensor Networks, vol. 8, no. 9, pp. 1-8, 2012, DOI: 10.1155/2012/387192

**[LEI09]** L. Dong, X. Wang and S. Li, "An energy-efficient scheme by cross-layer for wireless sensor network", Chinese Control and Decision Conference, Guilin, pp. 4738-4742, 2009, DOI: 10.1109/CCDC.2009.5194845

**[LEV03]** P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications", 1st international conference on Embedded networked sensor systems (SenSys), Los Angeles, pp. 126-137, 2003, DOI: 10.1145/958491.958506.

**[LEV05]** P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse et al., "TinyOS: An Operating System for Sensor Networks", Ambient Intelligence, pp. 115-148, 2005, DOI: 10.1007/3-540-27139-2_7

**[LIN17]** L. Xu, R. Collier and G. O'Hare, "A Survey of Clustering Techniques in WSNs and Consideration of the Challenges of Applying Such to 5G IoT Scenarios",

IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1229-1249, 2017, DOI: 10.1109/JIOT.2017.2726014

**[LIU02]** W. Liu and H. Song, "Research and implementation of mobile ad hoc network emulation system", 22nd International Conference on Distributed Computing Systems Workshops, Vienna, pp. 749-754, 2002, DOI: 10.1109/ICDCSW.2002.1030858

**[LLA15]** A. Llaria, G. Terrasson, H. Arregui and A. Hacala, "Geolocation and monitoring platform for extensive farming in mountain pastures", IEEE International Conference on Industrial Technology (ICIT), Seville, pp. 2420-2425, 2015, DOI: 10.1109/ICIT.2015.7125454

**M**

**[MAD02]** S. Madden, M. Franklin, J. Hellerstein and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks", ACM SIGOPS Operating Systems Review, vol. 3, pp. 131-146, 2002, DOI: 10.1145/844128.844142

**[MAG16]** M. Magno, V. Jelicic, B. Srbinovski, V. Bilas, E. Popovici et al., "Design, Implementation, and Performance Evaluation of a Flexible Low-Latency Nanowatt Wake-Up Radio Receiver", IEEE Transactions on Industrial Informatics, vol. 12, no. 2, pp. 633-644, 2016, DOI: 10.1109/TII.2016.2524982

**[MAN13]** S. Gupta, M. Ghonge, P. Thakare and P. Jawandhiya, "Open-source network simulation tools: An overview", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), vol. 2, no. 4, pp. 1629-1635, 2013

**[MEK08]** M. Mekni and B. Moulin, "A Survey on Sensor Webs Simulation Tools", Second International Conference on Sensor Technologies and Applications (SENSORCOMM), Cap Esterel, pp. 574-579, 2008, DOI: 10.1109/SENSORCOMM.2008.13

**[MER09]** G. Merrett, N. White, N. Harris and B. Al-Hashimi, "Energy-Aware Simulation for Wireless Sensor Networks", 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, Rome, pp. 1-8, 2009, DOI: 10.1109/SAHCN.2009.5168932

**[MIN15]** J. Yi, M. Kang and D. Noh, "SolarCastalia: Solar Energy Harvesting Wireless Sensor Network Simulator", International Journal of Distributed Sensor Networks, vol. 11, no. 6, pp. 1-10, 2015, DOI: 10.1155/2015/415174

**[MIN16]** I. Minakov, R. Passerone, A. Rizzardi, and S. Sicari, "A Comparative Study of Recent Wireless Sensor Network Simulators", ACM Transactions on Sensor Networks. vol. 12, no. 3, pp. 1-39, 2016, DOI: 10.1145/2903144

**[MIS15]** D. Mishra, and R. Kumar,"Qualitative Analysis of wireless sensor network simulators", IJCA Proceedings on National Conference on Knowledge, Innovation in Technology and Engineering (NCKITE 2015), vol. 2, pp. 11-18, 2015

**[MON07]** G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, 2007, DOI: 10.17487/RFC4944

**[MOT11]** L. Mottola and G. P. Picco, "MUSTER: Adaptive Energy-Aware Multisink Routing in Wireless Sensor Networks", IEEE Transactions on Mobile Computing, vol. 10, no. 12, pp. 1694-1709, 2011, DOI: 10.1109/TMC.2010.250

**[MRF10]** "MRF24J40 Data Sheet IEEE 802.15.4™ 2.4 GHz RF Transceiver", Microchip Technology Inc., DS39776C, 2010

**[MUS12]** B. Musznicki, and P. Zwierzykowski, "Survey of simulators for wireless sensor networks", International Journal of Grid and Distributed Computing, vol. 5, no. 3, pp. 23-50, 2012

**N**

**[NAC05]** L. Nachman, R. Kling, R. Adler, J. Huang and V. Hummel, "The Intel© mote platform: a Bluetooth-based sensor network for industrial monitoring", Fourth International Symposium on Information Processing in Sensor Networks (IPSN), Boise, pp. 437-442, 2005, DOI: 10.1109/IPSN.2005.1440968

**[NAV14a]** D. Navarro, F. Mieyeville, M. Galos, and L. Carrel. "Simulation of Hardware and Software in Heterogeneous Wireless Sensor Network", International Journal on Advances in Networks and Services, vol. 7, no. 1 and 2, pp. 97-107, 2014

**[NAV14b]** M. Navarro, Y. Li and Y. Liang, "Energy profile for environmental monitoring wireless sensor networks", IEEE Colombian Conference on Communications

and Computing (COLCOM), Bogota, pp. 1-6, 2014, DOI: 10.1109/ColComCon.2014.6860416

**[NIE15]** J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby et al., "IPv6 over BLUETOOTH(R) Low Energy", RFC 7668, 2015, DOI: 10.17487/RFC7668

**O**

**[OPA98]** "High-Speed, Single-Supply, Rail-to-Rail OPERATIONAL AMPLIFIERS", BURR-BROWN, PDS-1479B, 1999

**[ORA90]** D. Oran, "OSI IS-IS Intra-domain Routing Protocol", RFC 1142, 1990, DOI: 10.17487/RFC1142

**[OSI94]** "ISO/IEC 7498-1:1994, Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model", International Organization for Standardization (ISO), 1994

**[OST06]** F. Osterlind, A. Dunkels, J. Eriksson, N. Finne and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA", 31st IEEE Conference on Local Computer Networks, Tampa, pp. 641-648, 2006, DOI: 10.1109/LCN.2006.322172

**[OST10]** F. Österlind, J. Eriksson and A. Dunkels, "Cooja TimeLine: a power visualizer for sensor network simulation", 8th ACM Conference on Embedded Networked Sensor Systems, pp. 385-386, 2010, DOI: 10.1145/1869983.1870035

**P**

**[PAL16]** L. Palma, L. Pernini, A. Belli, S. Valenti and L. Maurizi et al., "IPv6 WSN solution for integration and interoperation between smart home and AAL systems", IEEE Sensors Applications Symposium (SAS), Catania, pp. 1-5, 2016, DOI: 10.1109/SAS.2016.7479840

**[PAP16]** G. Papadopoulos, K. Kritsis, A. Gallais, P. Chatzimisios and T. Noel, "Performance evaluation methods in ad hoc and wireless sensor networks: a literature study", IEEE Communications Magazine, vol. 54, no. 1, pp. 122-128, 2016, DOI: 10.1109/MCOM.2016.7378437

**[PAR00]** S. Park, A. Savvides and M. Srivastava, "SensorSim: A simulation framework for sensor networks", 3rd ACM international workshop on Modeling, analysis

and simulation of wireless and mobile systems, pp. 104-111, 2000, DOI: 10.1145/346855.346870

**[PAT13]** N. Patel, H. Kathiriya and A. Bavarva, "Wireless sensor network using Zigbee", International Journal of Research in Engineering and Technology, vol. 2, no. 6, pp. 1038-1042, 2013

**[PER03]** C. Perkins, E. Belding-Royer and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, 2003, DOI: 10.17487/RFC3561

**[PER08]** E. Perla, A. Catháin, R. Carbajo, M. Huggard, and M. Goldrick, "PowerTOSSIM z: realistic energy modelling for wireless sensor network environments", 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks (PM2HW2N), Vancouver, pp. 35–42, 2008, DOI: 10.1145/1454630.1454636

**[PES17]** D. Pesic, Z. Radivojevic and M. Cvetanovic, "A survey and evaluation of free and open source simulators suitable for teaching courses in wireless sensor networks", 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, pp. 895-900, 2017, DOI: 10.23919/MIPRO.2017.7973549

**[PHA07]** H. Pham, D. Pediaditakis and A. Boulis, "From Simulation to Real Deployments in WSN and Back", IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, Espoo, pp. 1-6, 2007, DOI: 10.1109/WOWMOM.2007.4351800

**[PIC18]** "PIC18F2525/2620/4525/4620 Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology", Microchip Technology, DS39626E, 2008

**[POR07]** J. Portilla, T. Riesgo and A. de Castro, "A Reconfigurable FPGA-Based Architecture for Modular Nodes in Wireless Sensor Networks", 3rd Southern Conference on Programmable Logic, Mar del Plata, pp. 203-206, 2007, DOI: 10.1109/SPL.2007.371750

**[POS80a]** J. Postel, "Internetwork Protocol Approaches", IEEE Transactions on Communications, vol. 28, no. 4, pp. 604-611, 1980, DOI: 10.1109/TCOM.1980.1094705

**[POS80b]** J. Postel, "Transmission Control Protocol", STD 7, RFC 793, 1981, DOI: 10.17487/RFC0793

**[POS81a]** J. Postel, "Internet Protocol", STD 5, RFC 791, 1981, DOI: 10.17487/RFC0791

**[POS81b]** J. Postel, "Transmission Control Protocol", STD 7, RFC 793, 1981, DOI: 10.17487/RFC0793

**[PRA15]** Prachi and S. Sharma, "Target tracking technique in wireless sensor network", International Conference on Computing, Communication & Automation, Noida, pp. 486-491. 2015, DOI: 10.1109/CCAA.2015.7148426

**[PRI15]** M. Prist, S. Longhi, A. Monteriù, F. Giuggioloni and A. Freddi, "An integrated simulation environment for Wireless Sensor Networks", IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), Boston, pp. 1-3, 2015, DOI: 10.1109/WoWMoM.2015.7158177

**[PUR09]** K. Purbhoo and K. Khedo, "eMnSiM: An energy oriented model and simulator for wireless sensor networks", University of Mauritius Research Journal, vol. 15, no. 1, pp. 200-229, 2009

**R**

**[RAH09]** M. Rahman, A. Pakštas and F. Wang, "Network modelling and simulation tools", Simulation Modelling Practice and Theory, vol. 17, no. 6, pp. 1011-1031, 2009, DOI: 10.1016/j.simpat.2009.02.005

**[RAM18]** J. Ramírez, P. Roose, M. Dalmau and M. Bakni, "Proposal and Validation of a Domaine Specific Language for the Representation of the AGGIR Constants", 11th International Joint Conference on Biomedical Engineering Systems and Technologies (HEALTHINF), vol. 5, pp. 438-445, 2018, DOI: 10.5220/0006590604380445

**[RAN14]** A. Ranjithkumar and J. Paranthaman, "Identifying human and tracking using WSN with webcam and GSM", International Conference on Information Communication and Embedded Systems (ICICES2014), Chennai, pp. 1-5, 2014, DOI: 10.1109/ICICES.2014.7034166

**[RAN15]** 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)", Ph.D. Thesis. UNIVERSITE DE NANTES, 2015, HALID: tel-01120429

**[RIZ12]** S. Rizvi, S., H.Qureshi, S. Khayam, V. Rakocevic, and M. Rajarajan, "A1: An energy efficient topology control algorithm for connected area coverage in wireless sensor networks", Journal of Network and Computer Applications, vol. 35,no. 2, pp. 597-605, 2012, DOI: 10.1016/j.jnca.2011.11.003

**[ROM04]** K. Romer and F. Mattern, "The design space of wireless sensor networks", IEEE Wireless Communications, vol. 11, no. 6, pp. 54-61, 2004, DOI: 10.1109/MWC.2004.1368897

**[ROM15]** G. Romaniello, "Energy Efficient Protocols for Harvested Wireless Sensor Networks. Other", Ph.D. Thesis, Université Grenoble Alpes, 2015, HALID: tel-01149450

**[RUI03]** L. Ruiz, J. Nogueira and A. Loureiro, "MANNA: a management architecture for wireless sensor networks", IEEE Communications Magazine, vol. 41, no. 2, pp. 116-125, 2003, DOI: 10.1109/MCOM.2003.1179560

**S**

**[SAD06]** C. Sadler and M. Martonosi, "Data compression algorithms for energy-constrained devices in delay tolerant networks", 4th international conference on Embedded networked sensor systems (SenSys), Boulder, pp. 265-278, 2006, DOI: 10.1145/1182807.1182834

**[SAG16]** S. Saginbekov and C. Shakenov, "Hybrid simulators for wireless sensor networks", IEEE Conference on Wireless Sensors (ICWiSE), Langkawi, pp. 59-65, 2016, DOI: 10.1109/ICWISE.2016.8188543

**[SAH05]** P. Sahoo, J. Sheu and C. Huang, "Power control based topology construction for the distributed wireless sensor networks", 24th IEEE International Performance, Computing, and Communications Conference (PCCC), Phoenix, vol. 30, no. 14-15, pp. 541-546, 2005., DOI: 10.1109/PCCC.2005.1460631

**[SAL01]** A. Salhieh, J. Weinmann, M. Kochhal and L. Schwiebert, "Power efficient topologies for wireless sensor networks", International Conference on Parallel Processing, Valencia, pp. 156-163, 2001, DOI: 10.1109/ICPP.2001.952059

**[SAK12]** G. Sakthidharan and S. Chitra, "A survey on wireless sensor network: An application perspective", International Conference on Computer Communication and Informatics, Coimbatore, pp. 1-5, 2012, DOI: 10.1109/ICCCI.2012.6158870

**[SAN07]** J. Sanchez, P. Ruiz, J. Liu and I. Stojmenovic, "Bandwidth-Efficient Geographic Multicast Routing Protocol for Wireless Sensor Networks", IEEE Sensors Journal, vol. 7, no. 5, pp. 627-636, 2007, DOI: 10.1109/JSEN.2007.894149

**[SAR11]** N. Sarkar and S. Halim, "A review of simulation of telecommunication networks: simulators, classification, comparison, methodologies, and recommendations", Cyber Journals. Special Issue, Journal of Selected Areas in Telecommunications (JSAT), vol. 2, no. 3, pp. 10-17, 2011

**[SAV13]** T. Savolainen, J. Soininen and B. Silverajan, "IPv6 Addressing Strategies for IoT", IEEE Sensors Journal, vol. 13, no. 10, pp. 3511-3519, 2013, DOI: 10.1109/JSEN.2013.2259691

**[SAV16]** D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch et al., "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)", RFC 7868,2016, DOI: 10.17487/RFC7868

**[SHA09]** A. Sharif, V. Potdar, and A. Rathnayaka, "Performance evaluation of different transport layer protocols on the IEEE 802.11 and IEEE 802.15.4 MAC/PHY layers for WSN", 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM), Kuala Lumpur, pp. 300–310, 2009, DOI: 10.1145/1821748.1821805

**[SHA13]** S. Sharma, D. Kumar, and K. Kishore. "Wireless sensor networks-A review on topologies and node architecture", International Journal of Computer Sciences and Engineering, vol. 1, no. 2, pp. 19-25, 2013

**[SHE14]** Z. Shelby, K. Hartke and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, 2014, DOI: 10.17487/RFC7252

**[SHR07]** A. Shrestha and L. Xing, "A Performance Comparison of Different Topologies for Wireless Sensor Networks", IEEE Conference on Technologies for Homeland Security, Woburn, pp. 280-285, 2007, DOI: 10.1109/THS.2007.370059

**[SIM04]** G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy et al., "Sensor network-based countersniper system", 2nd international conference on Embedded networked sensor systems (SenSys), Baltimore, pp. 1-12, DOI: 10.1145/1031495.1031497

**[SIN08]** C. Singh, O. Vyas and M. Tiwari, "A Survey of Simulation in Sensor Networks", International Conference on Computational Intelligence for Modelling Control & Automation, Vienna, pp. 867-872, 2008, DOI: 10.1109/CIMCA.2008.170

**[SOB06]** A. Sobeih, W. Chen, J. Hou, L. Kung, N. Li et al., "J-Sim: a simulation and emulation environment for wireless sensor networks", IEEE Wireless Communications, vol. 13, no. 4, pp. 104-119, 2006, DOI: 10.1109/MWC.2006.1678171

**[SOC91]** T. Socolofsky and C. Kale, "TCP/IP tutorial", RFC 1180, 1991, DOI: 10.17487/RFC1180

**[SOH07]** K. Sohraby, D. Minoli and T. Znati, "Wireless Sensor Networks: Technology, Protocols, and Applications", Wiley-Interscience, 2007, ISBN: 0471743003

**[SON08]** H. Song, S. Zhu and G. Cao, "SVATS: A Sensor-Network-Based Vehicle Anti-Theft System", IEEE 27th Conference on Computer Communications (INFOCOM), Phoenix, pp. 2128-2136, 2008, DOI: 10.1109/INFOCOM.2008.279

**[STE11]** A. Stetsko, M. Stehlik and V. Matyas, "Calibrating and Comparing Simulators for Wireless Sensor Networks", IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, pp. 733-738, 2011, DOI: 10.1109/MASS.2011.80

**[SUN11a]** Y. Sun, L. Li and H. Luo, "Design of FPGA-Based Multimedia Node for WSN", 7th International Conference on Wireless Communications, Networking and Mobile Computing, Wuhan, pp. 1-5, 2011, DOI: 10.1109/wicom.2011.6040365

**[SUN11b]** H. Sundani, H. Li, V. Devabhaktuni, M. Alam and P. Bhattacharya, "Wireless sensor network simulators a survey and comparisons", International Journal of Computer Networks, vol. 2 no. 5, pp. 249-265, 2011

**T**

**[TAH12]** D. Täht, "ns2 repository", github, [online] https://github.com/dtaht/ns2, last visited: 6 September 2020

**[TER09]** G. Terrasson, R. Briand, S. Basrour, V. Dupé and O. Arrijuria, "Energy Model for the Design of Ultra-Low Power Nodes for Wireless Sensor Networks", Procedia Chemistry, vol. 1, no. 1, pp. 1195-1198, 2009, DOI: 10.1016/j.proche.2009.07.298

**[THO07]** S. Thomson, T. Narten and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, 2007, DOI: 10.17487/RFC4862

**[TIA06]** T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu et al., "VigilNet: An integrated sensor network system for energy-efficient surveillance", ACM Transactions on Sensor Networks, vol. 2, no. 1, pp. 1-38, 2006, DOI: 10.1145/1138127.1138128

**[TIN13]** "tinyos-2.x-contrib", github, [online] https://github.com/tyll/tinyos-2.x-contrib, last visited: 6 September 2020

**[TOL05]** G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner et al., "A macroscope in the redwoods", 3rd international conference on Embedded networked sensor systems (SenSys), San Diego, pp. 51-63, 2005, DOI: 10.1145/1098918.1098925

**[TOS20]** "TOSSIM", Stanford University, [online] http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM, last visited: 6 September 2020, archive URL: https://web.archive.org/web/20200429050936/http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM, archive date: 29 April 2020.

**[TRE11]** P. Trenkamp, M. Becker and C. Goerg, "Wireless Sensor Network Platforms — Datasheets versus measurements", IEEE 36th Conference on Local Computer Networks, Bonn, pp. 966-973, 2011, DOI: 10.1109/LCN.2011.6115579

**V**

**[VAL11]** A. Vallimayil, K. Raghunath, V. Dhulipala and R. Chandrasekaran, "Role of relay node in Wireless Sensor Network: A survey", 3rd International Conference on Electronics Computer Technology, Kanyakumari, pp. 160-167, 2011, DOI: 10.1109/ICECTECH.2011.5941977

**[VAR16]** A. Varga, "OMNeT++ Simulation Manual Version 5.6.1", OpenSim Ltd, 2016.

**[VAR20]** A. Varga and B. Zoltan, "inet-framework", github, [online] https://github.com/inet-framework/inet, last visited: 6 September 2020

**[VAS17]** V. Vasanthi, "Simulators and emulators used for wireless sensor network", International Journal of Advanced Research in Computer and Communication Engineering, vol. 6, no. 1, pp. 171-175, 2017

**[VIE03]** M. Vieira, C. Coelho, D. da Silva and J. da Mata, "Survey on wireless sensor network devices", IEEE Conference on Emerging Technologies and Factory Automation (EFTA), Lisbon, vol. 1, pp. 537-544, 2003, DOI: 10.1109/ETFA.2003.1247753

**[VIE06]** M. Vieira, A. da Cunha and D. da Silva, "Designing Wireless Sensor Nodes", Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS, vol. 4017, pp. 99-108, 2006, DOI: 10.1007/11796435_12

**[VLA06]** N. Vlajic and D. Xia, "Wireless sensor networks: to cluster or not to cluster?", International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Buffalo-Niagara Falls, pp. 1-9, 2006, DOI: 10.1109/WOWMOM.2006.116

**W**

**[WAN10]** W. Du, D. Navarro, F. Mieyeville and F. Gaffiot, "Towards a taxonomy of simulation tools for wireless sensor networks", 3rd International ICST Conference on Simulation Tools and Techniques, Malaga, pp. 1-7, 2010, DOI: 10.4108/ICST.SIMUTOOLS2010.8659

**[WAN10b]** W. Du, F. Mieyeville and D. Navarro, "Modeling Energy Consumption of Wireless Sensor Networks by SystemC", Fifth International Conference on Systems and Networks Communications, Nice, pp. 94-98, 2010, DOI: 10.1109/ICSNC.2010.20

**[WAN11a]** W. Du, D. Navarro, F. Mieyeville and I. O'Connor, "IDEA1: A Validated System C-Based Simulator for Wireless Sensor Networks", IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, pp. 825-830, 2011, DOI: 10.1109/MASS.2011.95

**[WAN11b]** W. Du, “Modeling and simulation of wireless sensor networks”, Ph.D. thesis, Ecole Centrale de Lyon, 2011. NNT: 2011ECDL0026. HAL ID: tel-00690466

**[WER06]** G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson et al., "Deploying a wireless sensor network on an active volcano", IEEE Internet Computing, vol. 10, no. 2, pp. 18-25, 2006, DOI: 10.1109/MIC.2006.26

**[WIG08]** P. Wightman and M. Labrador, "A3: A Topology Construction Algorithm for Wireless Sensor Networks", IEEE Global Telecommunications Conference (GLOBECOM), New Orleans, pp. 1-6, 2008, DOI: 10.1109/GLOCOM.2008.ECP.74

**[WIT06]** G. Wittenburg, and J. Schiller, "Running real-world software on simulated wireless sensor nodes", ACM Workshop on Real-World Wireless Sensor Networks (REALWSN), Uppsala, 2006

**Y**

**[YIC08]** J. Yick, B. Mukherjee and D. Ghosal, "Wireless sensor network survey", Computer Networks, vol. 52, no. 12, pp. 2292-2330, 2008, DOI: 10.1016/j.comnet.2008.04.002

**[YAN13]** R. Yan, H. Sun and Y. Qian, "Energy-Aware Sensor Node Design with Its Application in Wireless Sensor Networks", IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 5, pp. 1183-1191, 2013, DOI: 10.1109/TIM.2013.2245181

**[YOU00]** Y. Ko and N. Vaidya, "Location-Aided Routing (LAR) in mobile ad hoc networks", Wireless Networks, vol. 6, pp. 307-321, 2000, DOI: 10.1023/A:1019106118419

**[YUA06]** L. Yuan and Y. Zhu, "Modeling and Simulating Wireless Sensor Transportation Monitoring Network", 6th World Congress on Intelligent Control and Automation, Dalian, pp. 8640-8644, 2006, DOI: 10.1109/WCICA.2006.1713667

**Z**

**[ZHA14]** Y. Zhang, L. Sun, H. Song and X. Cao, "Ubiquitous WSN for Healthcare: Recent Advances and Future Prospects", IEEE Internet of Things Journal, vol. 1, no. 4, pp. 311-318, 2014, DOI: 10.1109/JIOT.2014.2329462

**[ZHO11]** H. Zhou, D. Luo, Y. Gao and D. Zuo, "Modeling of node energy consumption for wireless sensor networks", Wireless Sensor Network, vol. 3, pp. 18-23, 2011, DOI: 10.4236/wsn.2011.31003

**[ZIE13]** S. Ziegler, C. Crettaz, L. Ladid, S. Krco, B. Pokric et al., "IoT6 – Moving to an IPv6-Based Future IoT", Future Internet Assembly (FIA), vol. 7858. pp. 161-172, 2013, DOI: 10.1007/978-3-642-38082-2_14

**[ZIG12]** "ZigBee Specification", ZigBee Standards Organization: San Ramon, Document 053474r20, 2012

**[ZIM80]** H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection", IEEE Transactions on Communications, vol. 28, no. 4, pp. 425-432, 1980, DOI: 10.1109/TCOM.1980.1094