بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

Dédicaces

A mes chers parents

Que nulle dédicace ne puisse exprimer ce que je leurs dois, pour leur bienveillance, leur affection et leur soutien…Trésors de bonté, de générosité et de tendresse, en témoignage de mon profond amour et ma grande reconnaissance « Que Dieu vous garde ».

A mes chersscœurs et frères

En témoignage de mes sincères reconnaissances pour les efforts qu'ils ont consenti pour l'accomplissement de mes études. Je leur dédie ce modeste travail en témoignage de mon grand amour et ma gratitude infinie.

A tous mes amis

Pour leur aide et leur soutien moral durant l'élaboration du travail de fin d'études.

Remerciements

Après avoir rendu grâce à DIEU,

Nous adressonsnos remerciements les plus chaleureux à :

Nos parents que nous ne remercierons jamais assez,

Nous souhaitons remercier toutes les personnes qui nous ont aidé d'une façon directe ou indirecte à la réalisation de ce rapport. Entre autres:

- ✓ Le département d'informatique de l'université de Biskra à travers son chef de département Mr BABA HNINI Mohamed Chawki.
- ✓ Notre encadreur académique Mr Djaber Khaled.
- ✓ Tous nos enseignants.
- ✓ Tous nos camarades de classe et amis.
- ✓ Nos amis : Djamal Adine Zerari, Sedik Menzer,Zouaoui Amer,Berreghis Abdelhakim,Hicham benhabrou.

ملخص :

إنترنت الأشياء (IoT) هو نموذج واعد يقوم على تمديد الربط بشبكة الإنترنت ليشمل أنواع مختلفة من الأشياء الذكية غير الحواسيب والهواتف النقالة، مما يسمح بتطوير أسلوب الحياة و تحسين جودة الخدمات في عدة مجالات. شبكات الاستشعار اللاسلكية (RCSFs) باعتبارها عنصرا حيويا في إنترنت الأشياء، تسمح بتمثيل الخصائص الديناميكية للعالم الحقيقي في العالم الافتراضي للإنترنت .على هاذ الأساس تم إستحداث نسخة مضغوطة من البروتوكول IPv6 لإنترنت الأشياء، مما يمكن للأجهزة المتصلة بشبكة الانترنت بالتواصل يبعضها و إيصال المعلومات كما تم تطوير بروتوكولات الاتصال المخصصة لإنترنت الأشياء، ومن أكثرالبروتوكولات استخدامًا في مجال هذه الأخيرة ، بروتوكول RPL. لذا نرى من المهم القيام بمحاكاة لبروتوكول RPL في محاكي COOJA و تمثيل نتائج استهلاك الطاقة الإجمالي و العدد الإجمالي للرسائل المستقبلة و المرسلة عبر الشبكة على شكل منحنيات وكذلك عرض كيفية نمذجة والتحقق من بروتوكول RPL بواسطة برنامج UPPAAL ، كما نفسر النتائج التي تم الحصول عليها في UPPAAL ، وفي الأخير نقوم بمقارنتها مع النتائج التي تم الحصول عليها في COOJA للتحقق من صحة نموذجنا.

Résumé:

Internet Objects (IoT) est un modèle prometteur qui étend la connexion Internet pour inclure différents types d'objets intelligents autres que les ordinateurs et les téléphones mobiles, permettant le développement de style de vie et l'amélioration de la qualité des services dans plusieurs domaines. Les réseaux de capteurs sans fil (RCSF) comme un élément essentiel dans l'Internet des objets, permettant de représenter les caractéristiques dynamiques du monde réel dans le monde virtuel de l'Internet. Sur cette base, une version compressée du protocole Internet IPv6 a été créée, permettant aux appareils connectés à Internet de communiquer et de communiquer les paramètres. Nous allons introduire l'internet des objets, ainsi que d'expliquer certains des concepts de base utilisés dans l'internet des objets. Le protocole RPL est l'un des plus couramment utilisé dans l'internet des objets. Nous simulons protocole RPL sous le simulateur COOJA et nous représentons les résultats (consommation totale d'énergie et le nombre total des messages envoyés/réceptionnés) sous forme de courbes ainsi que nous montrons comment nous avons modélisé et vérifié ce protocole sous UPPAAL, et nous interprétons les résultats obtenus par UPPAAL, et nous les comparons avec ceux obtenus par COOJA pour valider notre modèle.

Table des matières

LISTE DES FIGURES:

LISTE DES TABLEAUX

Glossaire des acronymes

- ✓ **RCSF** Réseau de Capteurs Sans Fil
- ✓ **IoT** Internet of Things
- ✓ **IoE** Internet of Everything
- ✓ **Cooja** Contiki os java simulator
- ✓ **TinySec** Tiny Security
- ✓ **6LoWPAN** IPv6 over Low power Wireless Personal Area Networks
- ✓ **RPL** Routing Protocol for Low power and lossy networks
- ✓ **DODAG** Destination Oriented Directed Acyclic Graph
- ✓ **DIO** DODAG Information Object
- ✓ **DIS** DODAG Information Solicitation
- ✓ **DAO** DODAG Destination Advertisement Object
- ✓ **DAO-ACK** DAO Acknowledgement
- ✓ **CoAP** Constrained Application Protocol
- ✓ **UDP** User Datagram Protocol
- ✓ **LBR** Low power and Lossy network Border Router
- ✓ **OF** Objective Function

Introduction Générale :

L'internet des objets a été introduit pour la première fois par kevin ashton. Il désigne l'omniprésence autour de nous d'une variété d'objets qui, à travers des schémas d'adressage uniques, sont capables d'interagir les uns avec les autres et de coopérer avec leurs voisin pour atteindre des objectifs communs. Les objets intelligents, qui sont considéré comme la plateforme de base de l'IOT, sont les objets de la vie quotidienne (réfrigérateur, téléviseur….etc.). Ces objets sont équipés de composants électronique tels que des supports de communication radio, des processeurs pour le traitement, des capteurs et/ou actionneurs etc. Cet ajout vise à les rendre capables d'être conscient du monde dans lequel ils se trouvent et de prendre son contrôle à un instant donné. Grace à ses potentialités et son aspect ubiquitaire, la montée en puissance de l'IOT peut s'observer dans plusieurs domaines des plus personnelles aux plus industrielles. Ceci a conduit à des bénéfices énormes, meilleur gestion d'énergie, traçabilité des produits, amélioration du suivi de la santé, simplification des tâches quotidiennes, etc.

Néanmoins, l'IOT n'est qu'en ses début, plusieurs progrès restent à faire en matière de standardisations, de sécurité et d'identification, d'optimisation de la consommation d'énergie et surtout de communications. En effet, l'omniprésence de l'internet des objets dans la vie quotidienne des individus impose l'établissement des solutions de communications meilleur. Ceci est une de communications faible engendre non seulement les perte dans les données et les réseaux, mais aussi, l'apparition de nouvelles menaces qui touchent à l'intégrité des objets eux-mêmes, les infrastructures et processus ainsi que privacy des personnes. Cela fait appel à de nouvelles tendances et innovations que ce soit sur le plan architectures de protocole de routage.

Ce mémoire est organisé en quatre chapitres :

Le premier chapitre sera consacré à la présentation de l'internet des objets, ainsi que l'introduction de quelques notions fondamentales utilisées dans le domaine de l'IOT(Les applications, Les technologies de communication, les protocoles…).

Dans le deuxième chapitre, nous présentons un des protocoles de communication les plus utilisés, dans le domaine de l'IoT, qui est le protocole RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks).

Le troisième chapitre est consacré à la compréhension du fonctionnement du protocole RPL, et l'explication de la simulation COOJA par une présentation de quelques interfaces de simulation. Après la simulation du protocole RPL sous COOJA, les résultats seront représentés sous forme de courbes pour une bonne lecture.

Dans le quatrième chapitre, nous montrons comment modéliser et vérifier le protocole RPL par le logiciel UPPAAL, et nous expliquons les résultats obtenus par UPPAAL, ces derniers vont être le sujet d'une comparaison avec les résultats obtenus par COOJA pour valider notre modélisation.

Chapitre I : Généralités sur l'Internet des objets

I.1. Introduction :

L'Internet des objets ou IoT (Internet of Things) [13], est un paradigme émergeant dans le monde des réseaux informatiques. Il peut être défini comme une évolution et extension de l'Internet de nos jours pour l'inclusion de tous les objets et les endroits dans notre entourage (réfrigérateurs, thermostat, maisons, véhicules, routes, etc.). Le concept prometteur de l'IoT va nous simplifier la vie, nous faire gagner du temps, décharger notre cerveau de la mémorisation de données logistiques (itinéraires, temps de prise des médicaments, etc.). Ainsi, l'accès ubiquitaire à différents types d'informations permettrait la sophistication du mode de vie et une amélioration significative de la qualité des services dans différents domaines.

L'IoT qui est une nouvelle vague de l'Internet, est en réalité une partie naissante de l'Internet du futur, appelé l'Internet de tous les objets ou IoE (Internet of Everything), qui vise à interconnecter les gens, les données et tous les objets, de telle sorte qu'il y ait une fusion entre le monde réel (physique) et le monde numérique (virtuel) ; les objets du monde physique vont être incorporés dans le monde virtuel de l'Internet. Cela fait appel à de nouvelles tendances et innovations que ce soit sur le plan architectures de communications ou sur le plan présentation et exploitation des services.

Ce chapitre est consacré à la présentation du domaine de l'Internet des objets et les aspects qui s'y rapportent.

I.2. Historique de l'Internet des objets :

L'émergence de l'Internet des objets ce n'est qu'un résultat de convergence entre multiples technologies, à savoir l'Internet, la communication sans fil, les systèmes embarqués, systèmes micro-électroniques et la nanotechnologie. Dans cette section, nous citons les évènements les plus marquants sur le chemin de la concrétisation de l'IoT.

Le concept d'un réseau de dispositifs intelligents a été évoqué pour la première fois en 1982, avec le premier appareil connecté à Internet à l'Université Carnegie Melon capable de signaler à son inventaire si les boissons nouvellement chargées sont bien froides. Ainsi, en 1991, Mark Weiser a introduit l'informatique omniprésente à travers son papier intitulé : «L'ordinateur du 21ème siècle » et a présenté d'avance la vision contemporaine de l'Internet des objets. Un peu plus tard, en 1994, Steve Mann avait créé le Wear Cam qui était parmi les premières caméras à apparaitre sur le web. Wear Cam comporte les parties suivantes : (1) un

groupe de caméras (ou uniquement une) qui sont fixées au corps, d'une manière quelconque, à deux mains libres (2) des moyens d'enregistrement, de traitement et de transmission des images capturées par les caméras (3) un moyen d'affichage qui a la capacité de présenter une image ou un flux d'images de l'appareil photo. Les images capturées seront communiquées vers une entité (une station de base) à la disposition de l'utilisateur. Ensuite, en 1998, l'informatique ubiquitaire a commencé d'attiré l'attention par le fait qu'elle permettrait l'incorporation flexible et efficace de l'informatique dans la vie quotidienne. Mark Weiser disait : « là où la réalité virtuelle met l'humain en dedans du monde des ordinateurs, l'informatique ubiquitaire force plutôt l'ordinateur à s'instaurer dans le monde réel».

En 1999, la désignation Internet des objets a été prononcée pour la toute première fois par Kevin Ashton. Après, en 2000 la société LG annonce son premier réfrigérateur intelligent connecté à Internet. De plus, la technologie RFID (Radio Fréquence Identification) qui est l'une des technologies constitutionnelles de l'IoT, a commencé à être massivement déployée vers les années 2003 et 2004. D'autre part, une initiative très intéressante a été prise en 2008 ; un groupe de recherche appelé IPSo Alliance s'est consacré à promouvoir l'utilisation du protocole IP (Internet Protocol) pour les réseaux d'objets miniatures intelligents.

De nombreux travaux de recherches ont été succédés et se sont tous concentrés autour de la réalisation, dans les meilleures conditions, de la vision de l'Internet des objets et la mener à sa maturité en dépit de tous les défis soulevés. Cela avec la considération des progrès technologiques continus dans le marché des dispositifs intelligents et dans le domaine de technologies de télécommunication (comme : le cloud computing, le concept du SDN (Software-Defined Networking) etc.).[6]

I.3. Définition :

L'Internet des objets, ou IoT (Internet of Things), est un scénario dans lequel les objets, les animaux et les personnes se voient attribuer des identifiants uniques, ainsi que la capacité de transférer des données sur un réseau sans nécessiter aucune interaction humain-à-humain ou humain-à-machine.

Alors qu'Internet ne se prolonge habituellement pas au-delà du monde électronique, l'Internet des objets connectés représente les échanges d'informations et de données provenant de dispositifs du monde réel avec le réseau Internet.

Considéré comme la troisième évolution de l'Internet, baptisé Web 3.0 (parfois perçu comme la généralisation du Web des objets mais aussi comme celle du Web sémantique) qui fait suite à l'ère du Web social, l'Internet des objets revêt un caractère universel pour désigner des objets connectés aux usages variés, dans le domaine de l'e-santé, de la domotique ou du quantified self.

L'Internet des objets est en partie responsable d'un accroissement exponentiel du volume de données généré sur le réseau, à l'origine du big data (ou méga données en français).

I.4. Typologie des objets :

Avec l'avènement de l'Internet des objets, la connexion Internet acquiert une troisième dimension; en plus de la possibilité de se connecter n'importe quand et n'importe où, il est désormais possible d'être connecté avec n'importe quel objet. De plus, les objets connectés sont identifiés de façon unique et sont capable de récolter des informations environnementales (liées aux changements des paramètres de l'environnement, comme la température) ou comportementales (issues des variations d'état de l'objet lui-même ou des objets contextuels), de les traiter et de les communiquer sur Internet. D'où vient leur appellation par objets intelligents.

CisCo prévoit que d'ici quelques années, spécifiquement en 2020, l'Internet des objets sera une réalité et le nombre d'objets connectés dépassera les 50 milliards [23]. A ce stade, il est nécessaire de noter que les données massives générées par un nombre immense d'objets intelligents connectés présente, partiellement, une source de la charge globale de données qualifiées de BigData sur Internet [24].

On distingue différents types de dispositifs connectés à l'IoT, ou qui font connecter d'autres objets à Internet, dont on cite principalement :

I.4.1. Les objets d'identification :

Codes barre, marqueurs RFID et autres dispositifs miniaturisés qui servent à l'identification et la traçabilité des objets sur lesquels ils sont collé pouvant être collés sur les objets d'usage courant (ex. vêtements, marchandises, livres, véhicules, etc.).

Ce type de dispositifs nécessite qu'il y ait un lecteur pour récupérer leurs données qui seront par la suite téléchargées sur un serveur et deviennent alors accessibles via le système d'information d'une organisation ou directement sur Internet.

I.4.2. Les capteurs :

Les capteurs dans l'IoT permettent de récolter des informations contextuelles concernant les objets dans lesquels ils sont intégrés, ou les environnements sur lesquels ils sont déployés. Les capteurs communiquent les informations collectées sur Internet d'une manière directe ou indirecte, tout dépend du modèle adopté pour l'intégration des réseaux de capteurs à l'internet.

I.4.3. Les drones :

Un drone désigne un aéronef miniature sans pilote, pouvant porter des charges utiles, communiquer et exécuter des commandes en toute flexibilité. Les drones sont utilisés dans des applications civiles aussi bien que dans des applications militaires pour accomplir des missions bien déterminées. On entend parler de l'efficacité de l'utilisation des drones dans le domaine commerciale pour par exemple, les livraisons à domicile des commandes faites sur Internet. Aussi, des opérations de sauvetage, d'exploration et de surveillance sont réalisables par les drones dans le contexte des applications militaires. Bien que la technologie (ou bien son prototype) des drones en elle-même existait depuis bien longtemps, son exploitation idéale dans différentes applications demeure modeste. Récemment, les drones sont élus pour faire une importante part de l'Internet du futur, soit en tant que objets intelligents terminaux rapportant des données de contrôle, soit en tant que routeurs particuliers (mobiles et volants) de données entre les parties connectées à Internet. Comparés aux capteurs qui sont le plus souvent stationnaires ou dans certains cas mobiles mais dans tous les cas, manquent de l'aspect aérien, un drone parvient très efficacement à donner une vision aérienne sur l'état de

la zone à contrôler même dans les zones isolée et/ou inaccessibles (là où il est difficile d'installer une infrastructure terrestre avec des points d'accès et des stations de base).

I.4.4. Smartphones et tablettes électroniques :

Les smartphones et les tablettes qui sont déjà connectés à Internet par le biais de diverses technologies (Wi-Fi, 3G, 4G) permettent aux utilisateurs de communiquer à distances avec les autres types d'objets connectés dans l'IoT. Les objets intelligents peuvent rapporter en temps réel l'état actuel aux utilisateurs via Internet. Dans ce cas, les utilisateurs reçoivent des e-mails ou simplement des messages d'alertes sur leurs Smartphones ou tablettes, tout dépend de l'application. Il est même possible que les utilisateurs supervisent ou ordonnent leurs objets connectés, à distance, via leurs smartphones ou tablettes.

La figure ci-dessous présente les principaux types d'objets dans l'IoT.

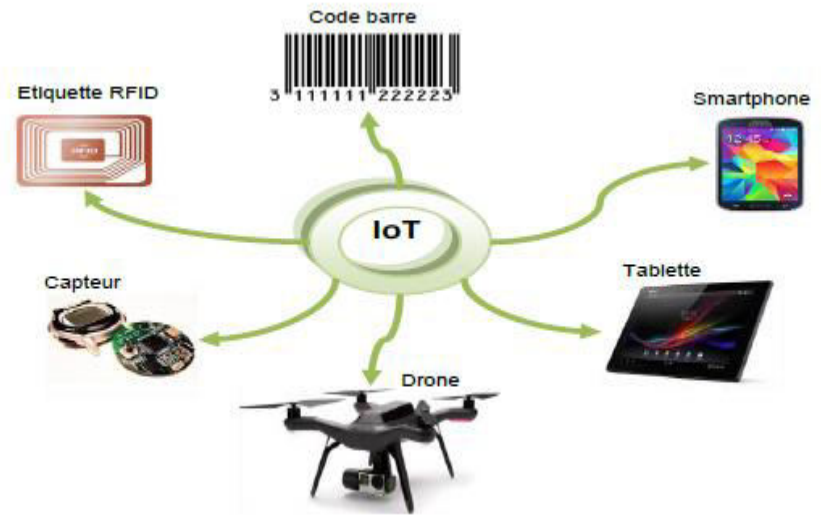

Figure 1:Typologie des objets dans l'IoT

I.5. Les applications de l'Internet des objets : [23]

L'Internet des objets ce n'est pas qu'un immense ensemble d'objets intelligents interconnectés et connectés à Internet mais c'est également et plus considérablement, les applications qui sont en fait la raison d'être de cette nouvelle vague de connectivité sur Internet. L'existence des objets intelligents avec de nouvelles possibilités de communications automatiques et intelligentes vont sensiblement améliorer le mode de vie

des gens ainsi que la qualité de services dans divers domaines à travers des degrés élevés d'autonomie et d'intelligence.

Les potentialités de l'Internet des objets ont mené à ce que des modèles de nouvelles applications soient développées sur Internet. Dans cette section, nous citons les applications en vedette de l'IoT.

I.5.1. Les applications médicales :

L'IoT aura de nombreuses applications dans le secteur de la santé où l'objectif est d'arriver à prévenir des situations graves et de suivre à distance des patients atteints des maladies chroniques et agir rapidement si cela s'est avéré nécessaire. Des capteurs corporels implantés dans le corps du patient récoltent des informations relatives aux paramètres médicaux, telles que la température, la glycémie, le rythme des battements du coeur ou encore même la tension artérielle. Ces informations seront stockées et traitées sur Internet (plus précisément sur un cloud) et mises à la disposition du médecin qui pourra les consulter n'importe quand et depuis n'importe quel dispositif connecté à Internet (ex : son Smartphone ou sa tablette). Le médecin est alerté en temps réel (en lui envoyant un mail ou un SMS) de tout changement brusque concernant l'état de son patient. Suivant le degré de gravité de la situation, le médecin réagit soit en se déplaçant chez le patient ou juste en le contactant et lui indiquant ce qu'il faut faire pour revenir à l'état normal. Imaginons par exemple un patient avec un rythme cardiaque irrégulier. Le capteur détectant tel évènement déclenche une alerte au cardiologue s'occupant du patient. Le médecin peut également consulter à tout moment les rapports médicaux de ses patients ou bien interroger les capteurs pour avoir les valeurs actuelles.

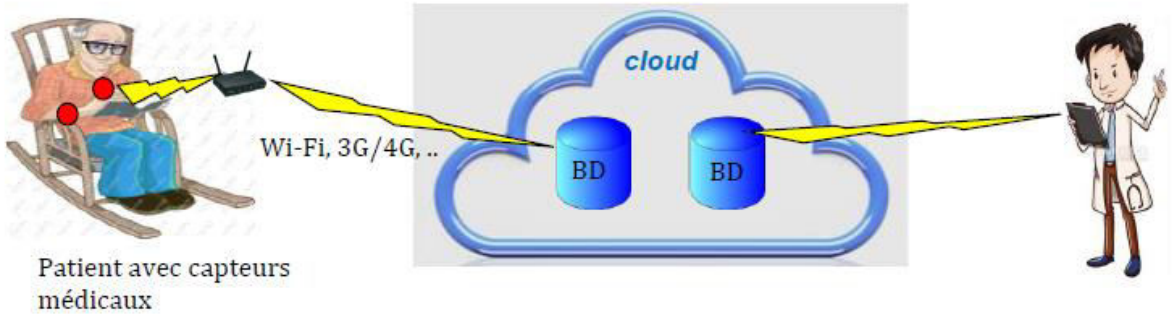

Figure 2:L'internet des objets dans le domaine médical

I.5.2. Les applications militaires :

L'Internet des objets est un domaine fertile tant pour les applications civiles que pour les applications militaires. Dans le domaine de défense les capteurs et les nano-drones connectés à Internet permettent d'envisager des applications sophistiquées pour l'exploration, la surveillance des champs de batailles et des frontières, ainsi que la poursuite et la localisation géographique des objets connectés. Les forces militaires ont la tendance d'utiliser des infrastructures propriétaires pour la connectivité et les communications. En transitant vers l'Internet, il sera plutôt possible d'utiliser des infrastructures cloud, qui offrent une flexibilité opérationnelle très intéressante. Le soldat en mission peut lui-même être connecté à Internet à travers les capteurs connectés, intégrés dans sa tenue. Ces capteurs peuvent être par exemple des capteurs médicaux qui rapportent l'état de santé du soldat, ou des capteurs multimédia qui captent des images, une vidéo ou du son depuis la zone où il se trouve (le soldat).

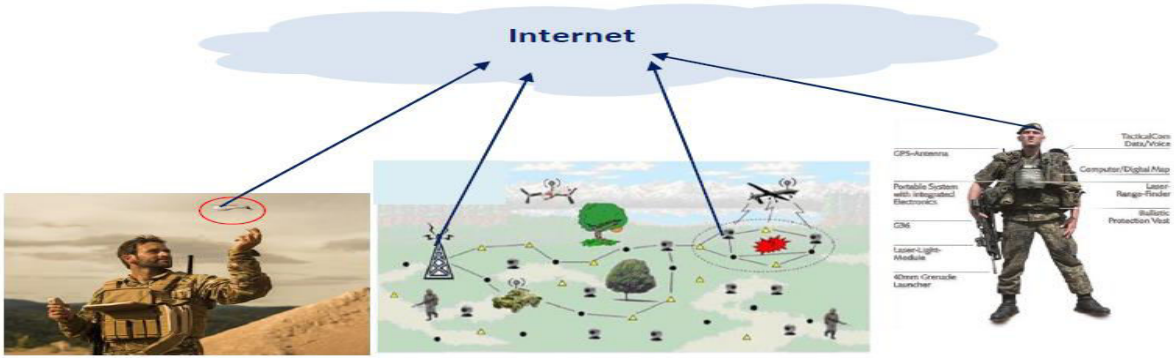

Figure 3:Le domaine militaire et l'Internet des objets

I.5.3. Les maisons intelligentes :

La maison du futur sera un objet connecté à Internet accessible à distance par ses propriétaires via des Smartphones, tablettes ou ordinateurs connectés. La porte, la télévision, le thermostat, le réfrigérateur, les parapluies, les montres, etc. de telle sorte qu'une porte connectée informe les parents par Internet de la rentrée de leurs enfants. La télévision qui était seulement un terminal récepteur. Connectée à Internet, elle (la télévision) devient plutôt un dispositif émetteur/récepteur qui fournit à ses téléspectateurs la possibilité d'envoyer et recevoir des e-mails, faire des appels téléphoniques sur Internet, ou autre. Un thermostat intelligent connecté au réseau Wi-Fi de la maison permet de contrôler facilement la température de celle-ci à partir de n'importe où, pour une amélioration du confort et une optimisation des économies énergétiques. Le réfrigérateur intelligent connecté à Internet et

muni d'un système RFID traque les produits élémentaires qui y sont stockés et enregistre des informations pertinentes leur concernant (comme la durée du stockage et la date d'expiration). L'utilisateur peut l'interrogé à distance pour savoir ce qui reste et ramener les produits manquant avant de rentrer à la maison. Ou alternativement, le réfrigérateur peut être programmé pour commander automatiquement les produits qui manquent.

Figure 4:L'Internet des objets et la domotique

I.6. cycle de vie d'un objet connecté dans l'IoT :

Dans l'IoT, les objets intelligents passent par trois étapes : la phase préparatoire la phase opérationnelle et la phase de maintenance [9].

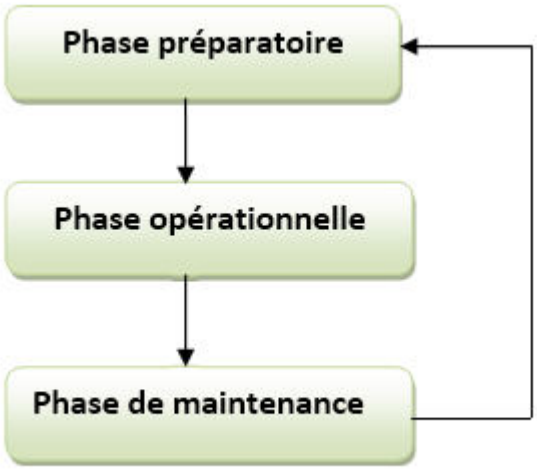

Figure 5:Cycle de vie de l'objet.

- **La phase préparatoire :** déploiement des objets (capteurs, tags), leur configuration avec les informations nécessaires, par exemple les identificateurs, les clés de sécurité, etc.
- **La phase opérationnelle :** dans la phase opérationnelle, l'objet connecté se met à réaliser samission qui diffère d'une application à une autre.
- **La phase de maintenance :** effectuer des mises à jours, régler les problèmes en faisant d'éventuelles réparations des objets en cas de défaillances par exemple. Il est même possible de remplacer carrément des objets et redémarrer à nouveau à partir de la phase préparatoire.

I.7. Les avantages de l'Internet des objets :

Nous avons cité dispersement dans différentes parties du présent chapitre quelques avantages de l'Internet des objets. Dans cette section, nous résumons les principaux avantages de l'IoT.

1) Accès ubiquitaire à l'information pour un monde plus intelligent et un mode vie sophistiqué et confortable.
2) Amélioration de la qualité de service et de la télésurveillance dans différents domaines d'applications, à savoir le domaine industriel, médical, etc.
3) Améliorer la productivité et l'expérience-client : les objets connectés envoient des rapports à leurs constructeurs indiquant les préférences et les habitudes des clients aidant davantage les entreprises à agir de manière proactive et adaptée qui satisfait la demande et les exigences de la clientèle.
4) Le gain du temps est un autre avantage de l'IoT. Les déplacements inutiles sont dès lors remplacés par une simple navigation sur le web pour commander des produits, contrôler l'état des objets et/ou endroits connectés.
5) Dans certaines applications, l'IoT nous permet même de rationaliser nos dépenses et faire des économies car on ne consomme qu'en cas de besoin, que ça soit pour les achats ou la consommation énergétique (nécessaire pour l'éclairage ou la climatisation) ou autre.
6) Possibilité d'exploitation des ressources géantes de l'Internet pour le stockage et le traitement des données écoulées de l'IoT.

I.8. Architecture de l'Internet des objets : [23]

De point de vue architectural, on peut dire que l'Internet des objets est organisée en trois couches principales: la couche de perception de donnée, la couche réseau et troisièmement la couche application. La figure ci-dessous illustre telle organisation.

I.8.1. La couche perception :

La couche perception, au niveau bas dans la hiérarchie, est responsable de la capture de données, ainsi que leur identification dans leur environnement. Cette couche comprend ainsi le matériel nécessaire pour parvenir à la collection de données contextuelles des objets connectés, à savoir les capteurs, les étiquettes RFID, caméras, GPS (Global Positioning System), etc.

I.8.2. La couche réseau :

Cette couche se charge de la transmission fiable des données générées dans la couche perception ainsi que l'assurance de la connectivité inter-objets connectés et entre objets intelligents et les autres hôtes de l'Internet. D'autre part, il est prévu que les données issues de la couche perception soient énormes car le nombre d'objets connectés à Internet ne cesse d'augmenter à grands pas. De ce fait, il s'est avéré nécessaire de mettre en place des mécanismes et des équipements de stockage et de traitement massif de ces données sur Internet, à faible coût. Cela est bel et bien garanti par les services cloud [7] qui assurent une gestion élastique des ressources de mémorisation et de traitement sur les géants centres de données résidant sur Internet et qui sont en mesure d'absorber efficacement la charge de données générée du côté de l'Internet des objets. à ce stade, il est important de noter que le cloud utilise un concept récent dénommé SDN (Software Defined Networking) qui vise une méthode de gestion abstraite basée sur le découplage des fonctionnalités décisionnelles et opérationnelles des équipements réseau, en vue de pouvoir déployer les tâches de contrôle sur des plateformes beaucoup plus performantes que les commutateurs classiques. Cela va réduire davantage la latence réseau et rendre possible l'automatisation de la gestion du large ensemble de serveurs sur le cloud et leur auto-configuration.

I.8.3. La couche application :

Quant à elle, la couche application définit les profiles des services intelligents et les mécanismes de gestion de données de différents types, provenant de différentes sources

(différents types d'objets). Dans la section suivante nous abordons cet aspect applicatif et ce que représentent réellement les services intelligents dans chaque champ d'application.

L'architecture peut être étendue à une quatrième couche dite la couche middleware [22] entre la couche application et les deux autres couches. Cette couche sert pour une interface entre la couche matérielle et les applications. Elle comprend des fonctionnalités assez compliquées permettant la gestion des dispositifs, et traite aussi l'agrégation, l'analyse et le filtrage de données et le contrôle d'accès aux services. La couche middleware permet également la dissimulation de la complexité des mécanismes de fonctionnement du réseau et rend plus facile le développement des applications par les concepteurs.

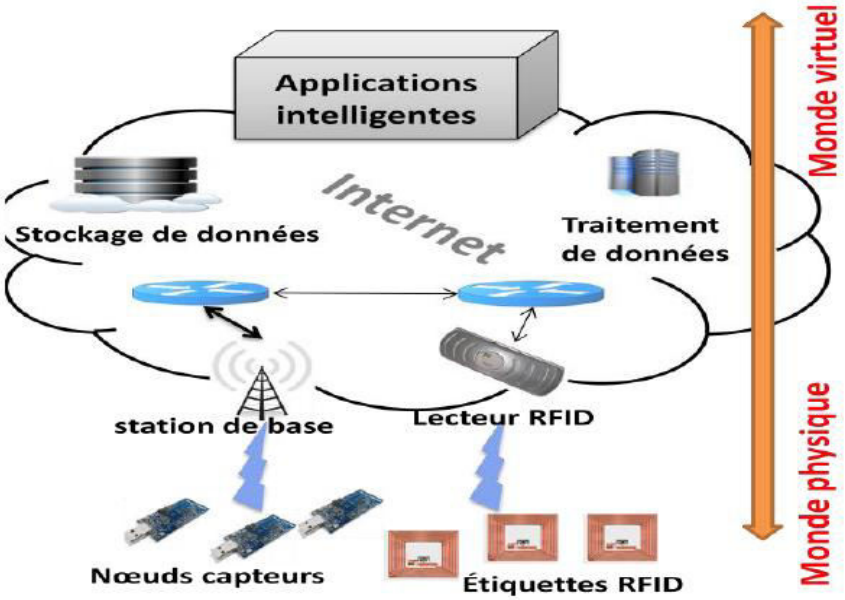

Figure 6:Architecture de l'internet des objets.

I.9. Les technologies de communication des objets connectés : [31]

I.9.1. Les technologies de courte portée:

I.9.1.1 NFC :

Les protocoles Near Field Communication (NFC) sont fondés sur la technologie d'identification par radio fréquence RFID (Radio frequency identification). Les objets équipés d'une puce électronique RFID possèdent une « étiquette » et sont automatiquement identifiés par radio fréquence lorsqu'ils se trouvent à proximité d'un équipement appelé interrogateur. Le protocole NFC est un standard de communication radiofréquence sans contact à très courte distance, de l'ordre de quelques centimètres, permettant une communication simple entre deux équipements électroniques. Il est par exemple utilisé dans

de nombreuses entreprises pour les badges d'accès aux locaux, ou comme support d'un abonnement à un réseau de transport en commun.

Figure 7:La technologie NFC.

I.9.1.2. Bluetooth :

Inventé en 1994 par la société suédoise Ericsson, le protocole Bluetooth est un standard de transfert de données sans fil. Il utilise une faible bande passante, ce qui ne lui permet de transférer que peu de données à de courtes distances, mais est également très peu énergivore. Inclus à l'immense majorité des téléphones mobiles, afin de réaliser une communication entre deux téléphones, ou entre un téléphone et un objet connecté de nature différente, il possède désormais de nombreuses applications : oreillette de discussion téléphonique sans fil, montre intelligente, moniteur de fréquence cardiaque, enceinte portative de diffusion de musique, station météo, thermostat, etc. Ce protocole est également utilisé sur des capteurs statiques appelés beamers pour mesurer des flux, par exemple des clients dans un magasin [26].

Figure 8:La technologie Bluetooth.

I.9.1.3. Zigbee :

Zigbee est un protocole de communication radio développé spécifiquement pour les applications de domotique. D'une portée moyenne de 10 mètres, il utilise une faible bande passante et est idéal pour le transfert de données en faible volume. Peu énergivore et conçu pour des échanges de données à bas débit, le dispositif Zigbee convient aux appareils alimentés par une pile ou une batterie, et en particulier aux capteurs. Il est conçu pour fonctionner en réseau maillé : chaque nœud reçoit, envoie et relaie des données. Il est par exemple utilisé par certains détecteurs de fumée.

Figure 9:La technologie Zigbee.

I.9.2. Les technologies de moyenne portée :

I.9.2.1. Z-Wave :

Le Z-Wave est un protocole de communication sans fil principalement dédié à la domotique. Il permet de transmettre des données sur des distances allant de 30 mètres en intérieur à 100 mètres en plein air. Il fonctionne en réseau maillé, chaque appareil connecté pouvant relayer les informations émises par ses voisins, ce qui lui permet d'élargir sa portée. Le protocole Z-Wave a été développé pour des usages peu énergivores nécessitant un faible débit de données. Tout comme le protocole Zigbee, l'utilisation de Z-Wave ne nécessite que très peu de puissance et les appareils peuvent donc communiquer pendant plusieurs années avec une simple pile[32].

Figure 10:La technologie Z-Wave.

I.9.2.2. Wi-Fi :

Le Wi-Fi désigne un ensemble de protocoles de communications sans fil, permettant des connexions à haut débit sur des distances de 20 à 100 mètres. Il s'agit d'un réseau local sans fil très énergivore, qui ne convient que pour les appareils branchés sur secteur ou dont l'alimentation électrique peut être aisée et fréquente. Il permet de transférer rapidement beaucoup de données. Il existe différentes normes Wi-Fi correspondant à une portée et un débit variables.

Figure 11:La technologie Wi-Fi.

I.9.3. Les technologies de longue portée :

I.9.3.1. Réseaux cellulaires mobiles :

Fournis par les opérateurs de télécommunication, les réseaux cellulaires mobiles, basés sur la technologie GSM, permettent de transférer une quantité importante de données à une longue portée. Ils nécessitent l'installation d'une carte SIM dans l'appareil à connecter, afin d'identifier celui-ci sur le réseau de communication. Succédant aux premières générations des standards pour la téléphonie mobile, qui ont progressivement permis d'accroître le débit de communication, la quatrième génération (4G) permet une communication mobile à très haut débit.

I.9.3.2. Réseaux radio bas-débit :

I.9.3.2.1. Sigfox :

Est un réseau de communication radio sans fil à bas débit et à basse fréquence, d'une portée moyenne de 10 kilomètres en milieu urbain et de 30 à 50 kilomètres en milieu rural. Il est également une technologie créée par l'entreprise du même nom. Ce réseau convient à des appareils à basse consommation, dotés ainsi d'une grande autonomie, qui transfèrent une faible quantité de données.

Figure 12:La technologie Sigfox.

I.9.3.2.2. LoRa :

Est un protocole de communication radio à très basse consommation, qui permet de transmettre des données en petite quantité, à des distances de 2 à 5 kilomètres en ville et jusqu'à 45 kilomètres en milieu urbain. À l'instar de Sigfox, il s'agit d'un dispositif qui convient particulièrement aux équipements peu énergivores n'émettant que périodiquement, notamment les capteurs.

Figure 13:La technologie LoRa.

Les entreprises choisissent la technologie de télécommunication qui connectera leur parc d'objets communicants en fonction d'un certain nombre de critères, notamment techniques, tels que la portée, le débit et l'autonomie, c'est-à-dire la consommation électrique des objets connectés.

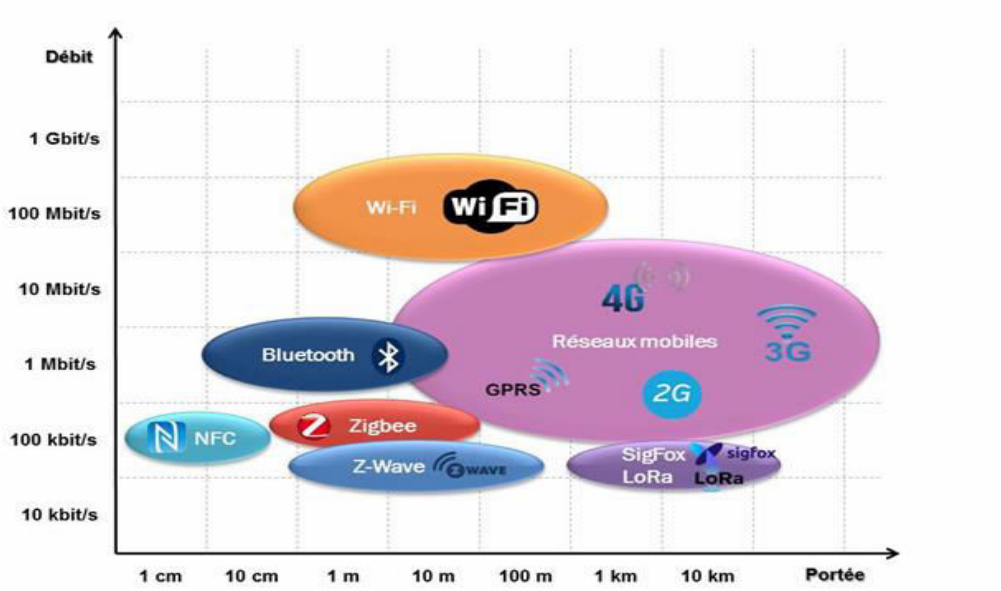

Figure 14:Les technologies de communication des objets connectés par Débit.

I.10. les protocoles de l'IoT :

Les protocoles spéciaux de l'IoT sont :

I.10.1. Le protocole IEEE 802.15.4 :

Le protocole IEEE 802.15.4 est fortement lié au protocole ZigBee. Il définit la couche MAC et la couche physique pour les réseaux personnels sans fil à bas débit, aussi appelés LOW Rate Wireless Personal Area Networks en anglais(LRWPAN). Cette norme convient très bien aux besoins des réseaux de capteurs sans fil en terme de bas débit, faible consommation énergétique et faible cout des entités du réseau.

Deux types d'entités sont présents dans ce réseau : les FFD (Full FunctionDevice) et les RFD (ReducedFunctionDevice). Un FFD peut fonctionner selon trois profils différents dans le réseau : celui d'un coordinateur du PAN (Personnal Area Network), d'un coordinateur ou d'une feuille. Un RFD en revanche ne peut être qu'une feuille, et dance ne contient pas toutes les fonctionnalités prévues pour la couche MAC 802.15.4 Un coordinateur peut communiquer avec toute autre entité du réseau alors qu'un RFD ne peut communiquer qu'avec un FFD.

Le protocole IEEE 802.15.4 supporte aussi deux types de topologies (figure 15) :[17]

- Les topologies en étoile : les stations communiquent uniquement avec un nœud central, qui est le coordinateur du PAN qui est la seule entité à pouvoir accepter de nouvelles associations dans le réseau. La taille du réseau est limitée à la portée du coordinateur.
- Les topologies pair-à-pair : les coordinateurs fils du coordinateur du PAN permettent à d'autres entités de s'associer au réseau à travers eux. Cela permet la formation d'un réseau plus complexe et plus large comme les réseaux mailles.

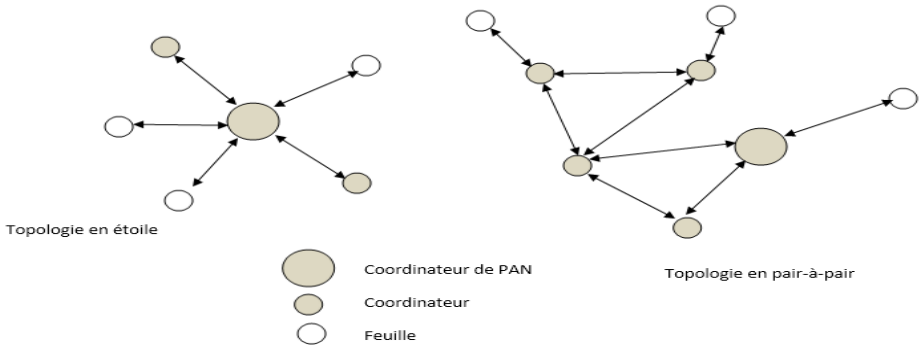

Figure 15:Exemple de la topologie en étoile et pair-à-pair.

I.10.2. Le protocole CoAP :

CoAP (Constrained Application Protocol en anglais) est un protocole de couche d'application qui est destiné à être utilisé principalement aux petits engins , qui sont des petits capteurs de 8 bits pour tout processeur, très peu de mémoire et qui sont connectées par des liens radio lents et peu fiables , spécialement conçu pour des applications M2M (machine-to-machine). On pourrait décrire CoAP comme composé de deux parties, un transport utilisant UDP avec extensions optionnelles si on veut de la fiabilité (accusés de réception), et conçu pour traduire facilement HTTP pour une intégration simplifiée avec le web[8].

I.10.3. Le protocole RPL :

Le protocole RPL (IPv6 Routing Protocole for Low-Power and Lossy Networks(LLNs)) définie par IETF (Internet Engineering Task Force en Anglia) est un protocole de routage IPv6 à vecteur de distance. Ce protocole et ses règles seront détailles dans le deuxième chapitre.

I.11. Conclusion :

Ce chapitre décrit une vision technique de l'Internet des objets en fonction des phases qu'elle a franchies, Aussi que la Typologie des objets et la description de l'architecture Internet des objets et Les technologies de communication des objets connectés, ainsi que les différents protocoles . Le protocole RPL fera l'objet du chapitre suivant.

Chapitre II : Le protocole RPL

II.1. Introduction :

Dans les réseaux de l'Internet des objets, les dispositifs qui les composent sont très limités en terme de mémoire et d'énergie, Dans ce type d'environnement on a besoin d'avoir un bon protocole de routage qui permet d'économiser l'énergie. C'est pour cette raison qu'un nouveau protocole de routage appelé RPL beaucoup plus optimisé en terme de coût énergétique a été proposé. Dans ce chapitre, une description détaillée de ce protocole sera présentée.

II.2. Définition :

Le protocole de routage RPL a été défini de manière à pouvoir fonctionner sur n'importe quelle couche de liaison de données, c'est-à-dire sur une grande variété de technologie ayant des caractéristiques hétérogènes telles que des solutions sans fil (par exemple : IEEE 802.154, Bluetooth, LOW Power Wi-Fi) ou filaires (par exemple : CPL).

RPL définit donc un ensemble de mécanismes génériques pour la construction de la topologie, mais laisse un grand nombre de paramétrages possibles afin de l'adapter aux environnements spécifiques. L'administrateur en charge du réseau RPL doit donc configurer avec soin le protocole RPL.

RPL fait partie du travail réalisé par l'IETF (Internet Engineering Task Force) pour définir une architecture IPv6 pour les LLN (Low-Power and Lossy Network). Le choix d'IPV6 au niveau IP est un choix cohérent du fait de l'espace d'adressage IPv6 quasiment illimité (le célèbre chiffre de 667 millions de milliards d'appareils connectés par millimètre carré de notre Terre pour saturer le système est souvent cité), ce qui permet d'envisager l'internet des objets [30]

II.3. La structure DAG : [9]

RPL est un protocole de routage comprenant un algorithme de structuration du réseau. La topologie crée par RPL est un graphe orienté acyclique DAG (Directed Acyclic Graph). Le DAG définit une structure arborescente spécifiant les routes par défaut entre les nœuds du LLN. Cependant, une structure de DAG est plus qu'un arbre typique dans le sens où un nœud peut être associé à plusieurs parents dans le DAG.

L'objectif de cette topologie est d'offrir un routage efficace et fiable de n'importe quel point du réseau vers la racine du DAG.

Le protocole RPL organise les nœuds sous forme de DODAG (DAG Destination-Ofiented) c'est à dire un graphe acyclique orienté dirigé vers une destination qui est la racine du réseau (seul un parent est autorisé) comme illustré dans la figure 16 Chaque DODAG a un numéro de version.

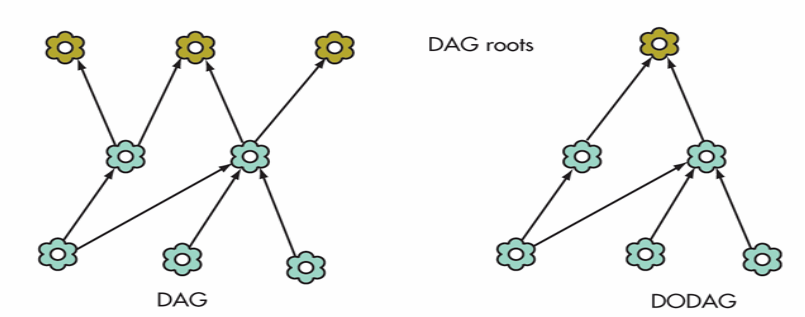

Figure 16:DAG et DODAG.

II.3.1. Identifiants RPL :

RPL utilise quatre valeurs pour identifier et maintenir une topologie :

II.3.1.1. RPLInstanceID :

Il identifie un ensemble d'un ou plusieurs DAGs orientés vers la destination (DODAG). Un réseau peut avoir plusieurs RPLInstanceID, chacun définissant un ensemble indépendant de DODAGs, qui peut être optimisé pour différentes Fonctions Objectives (OF) et / ou applications. L'ensemble des DODAG identifiés par un RPLInstanceID est appelé une instance RPL. Tous les DODAG dans la même Instance RPL utilisent la même fonction objective [10].

II.3.1.2. DODAGID :

Est une instance RPL. La combinaison de "RPL Instance ID" et DODAGID identifie de manière unique un seul DODAG dans le réseau. Une instance RPL peut avoir plusieurs DODAGS, dont chacun a un DODAGID unique Il est nécessaire en raison des exigences de l'application qu'un nœud peut éviter de choisir un nouveau parent préféré qui va à une autre racine DODAG. Les DODAGID ajoutent beaucoup de complexité et augmentent considérablement la taille des messages, Dans ce cas, la présence d'instances RPL et le préfixe de destination indique que la connectivité est déjà traitée [10.19].

II.3.1.3. DODAG Version Number :

Est une itération spécifique d'un DODAG avec un DODAGID donné ou bien est un compteur Séquentiel qui est incrémenté racine pour Former une nouvelle version du DODAG. La combinaison de "RPL instance ID", DODAGID et "DODAG Version Number" identifie de manière unique une version DODAG.

Lorsque le "DODAG Version Number" est incrémenté, une nouvelle version DODAG se propage vers l'extérieur à partir de la racine DODAG. Un parent qui annonce le nouveau " DODAG Version Number" ne peut pas appartenir au sous-DOPAG d'un par conséquence, un nœud peut ajoutez en toute sécurité un parent de n'importe quel rang avec un "DODAG Version Number" plus récent sans former de boucle [10.19]

II.3.1.4. Rank :

C'est le rang et il est défini comme étant « la position individuelle d'un nœud par rapport à d'autres nœuds et la racine DODAG ». C'est un entier qui représente l'emplacement d'un nœud dans le DODAG. Le rang augmente strictement dans le sens aval de DODAG, et diminue strictement dans le sens amont. En d'autres termes, les nœuds sur le haut de la hiérarchie reçoivent plus petits rangs que ceux dans le fond et le plus petit rang est affecté à la racine DODAG (figure 17). [9]

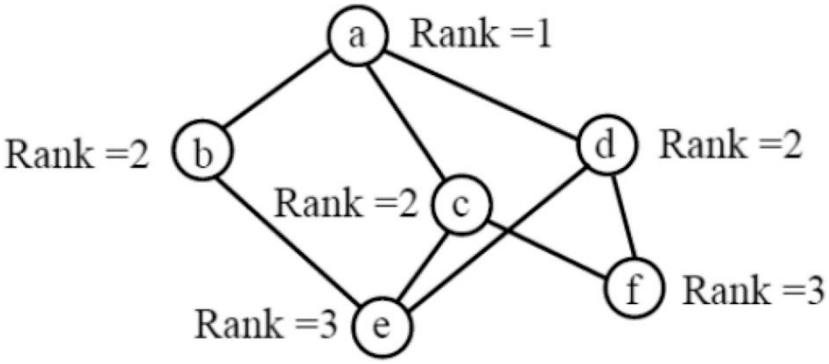

Figure 17:représentation d'un rang.

Le rôle du DODAG est de minimiser le coût pour atteindre le nœud source à partir de n'importe quel nœud dans le réseau. Le routage vers la source du DODAG est basé sur les valeurs du rang.

Un nœud choisit comme parent le voisin avec le rang le plus petit lorsqu'il envoie un paquet vers la source du DODAG. [1]

II.4. Instance et fonction objective :

Un réseau RPL contient au moins une instance RPL qui elle-même se compose d'un ou plusieurs DODAGs qui partagent un "RPL Instance LD". Au plus, un nœud RPL peut appartenir à un DODAG dans une instance RPL.

Une instance RPL fonctionne indépendamment des autres instances Chaque instance RPL est associée à une fonction objectif (OF) qui permet d'optimiser la topologie en fonction d'un ensemble de contraintes et/ou de métriques comme la préservation de l'énergie, le chemin le plus court ou la qualité des liens. Un nœud peut faire partie d'un seul DODAG par instance, mais peut participer à plusieurs instances simultanément. [15]

II.5. Types de messages du protocole RPL :

La construction et la maintenance des DODAGS sont réalisées grâce à des messages de contrôle (DID, DAO et DIS)

II.5.1. DIO (DODAG Information Object) :

Les nœuds dans un réseau RPL échangent périodiquement des messages DIO La transmission de ces messages est régulée par un temporisateur appelé «TrickleTimer» pour éliminer les messages de contrôle redondants. Dans l'algorithme de ce temporisateur, le temps est divisé en une infinité d'intervalles de taille « I ». La transmission de message DIO est aussi conditionnée par un seuil. Le nombre de messages de contrôle émis depuis le début del'intervalle ne doit pas dépasser ce seuil. Cet échange périodique de message DIO permet de faire le roulage vers la source du DODAG

II.5.2. DAO (Destination Advertisement Object) :

Ce message est envoyé en unicast vers la source du DODAG par chaque nœud dans le réseau. Le contenu des messages DAO permet d'effectuer du routage vers le bas, c'est-à-dire que la source du DCDAG connait les chemins pour atteindre n'importe quel nœud du réseau.

N.B. : Un autre type de message RPL est le DAO ACK (Destination Advertisement Object ACknowlodgement) qui est optionnel. Lorsqu'il est émis par un nœud, ce dernier n'est pas prêt à devenir un nœud parent pour le nœud émetteur du message DAO [1].

II.5.3. DIS (DODAG Information Sollicitation) :

Un nouveau nœud peut rejoindre un réseau déjà formé en diffusant un message DIS pour solliciter en réponse un message DlO qui contient des informations sur le DODAG comme le numéro de version et l'identifiant DODAG, l'identifiant de l'instance et l'OF utilisée. Un nœud peut également attendre de recevoir un message DIO diffusé périodiquement par ses voisins. La fréquence d'envoi des messages DlO est déterminée par un temporisateur fondé sur l'algorithme Tricklc (appelé également temporisateur Trickle).

À la moindre anomalie dans le réseau, le temporisateur Trickle est réinitialisé pour permettre à la topologie de ré-converger. [21]

II.6. Le fonctionnement de l'algorithme Trickle : [5]

L'algorithme Trickle est utilisé pour contrôler l'émission des messages DlO par chaque nœud dans le réseau. Pour son fonctionnement, Trickle utilise trois paramètres

- La valeur de l'intervalle de départ appelé «Imin ». Dans ce cas l'algorithme Trickle commence son fonctionnement avec une taille d'intervalle égale à « min »
- La valeur de l'intervalle maximale de fonctionnement appelé « Imax ». Lorsque l'intervalle de fonctionnement atteint la valeur « Imax », alors elle se stabilise et n'augmente plus.
- La constante de redondance « k » qui fixe le nombre de messages appelés "message consistant" reçus par un nœud pendant un intervalle « I »et au-delà du quel ce nœud peut considérer que le réseau est stable et qu'il est donc inutile de générer un message DIO.

Trickle initialise l'intervalle « l » qui sépare l'envoie de deux DIO à Imin, Si aucune inconsistance n'a été détectée pendant cet intervalle, I est augmenté de façon exponentielle (doublé) jusqu'à Imax et se stabilisera alors à cette valeur de Imax. Au début d'un intervalle I chaque nœud choisi aléatoirement l'instant d'attente avant l'envoi possible d'un message DIO.

Ce temps d'attente est choisi aléatoirement en suivant une distribution uniforme dans l'intervalle [1/2, I], A l'échéance de ce délai, si le nombre de messages consistants reçus depuis le début de l'intervalle I est inférieur à la constante de redondance k alors le nœud envoie un message DlO sinon il est annulé.

Dans le protocole RPL, un nœud du réseau détecte une inconsistance s'il reçoit un message DlO d'un de ses voisins qui possède des informations non cohérentes avec celles qu'il dispose. Par exemple si le message DIO reçu comporte un identifiant du DODAG qui est différent de celui dont le nœud disposait.

En cas d'inconsistance, quel que soit la taille actuelle de l'intervalle de fonctionnement, l'algorithme Trickle revient à l'intervalle Imin afin d'envoyer rapidement des messages DIO.

De façon générale, Trickle permet au protocole RPL de réduire le trafic de contrôle lorsque le réseau est consistant et stable, mais en cas d'inconsistance, il envoi plus de messages.

II.7. Procédure de construction du DODAG : [11]

Dans un DODAG, les nœuds peuvent être regroupés en trois types en fonction des tâches qu'ils assument

II.7.1. Le nœud Racine (root):

Encore appall LBR (Low Power and Lossy Network Border Router), il joue un rôle très important dans le réseau. La racine est à l'origine de l'envoi des messages DIO servant à la construction du DODAG, Il maintient les tables de routage permettant d'atteindre chaque nœud au sein d'un DODAG.

II.7.2. Les nœuds parents ou routeurs :

Un nœud parent possède au moins un fils (un nœud de rang inférieur). Le nœud parent assure la circulation des informations entre la racine et ses fils.

Les nœuds feuilles : ce sont des nœuds qui ne possèdent aucun fils. Ils sont seulement connectés à leur nœud parent qui envoie les informations en provenance de la racine du DODAG.

Le graphe DODAG formé par le protocole RPL. Est une topologie logique de routage construite à partir d'un réseau physique en utilisant un certain nombre de critères fournis par l'administrateur de système.

Ci-dessous nous présentons les étapes principales de construction d'un DODAG, à partir d'un réseau physique donné :

1.Première étape :

La racine ou LBR diffuse le message DIO dans son voisinage à un saut (Figure 18).

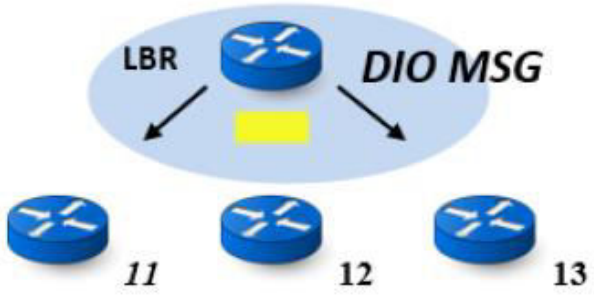

Figure 18:Diffusion des messages DIO par le LBR.

2. Deuxième étape :

Chacun des nœuds ayant reçu le DIO envoyé par le LBR, va utiliser la fonction objectif contenu dans le message DIO et l'algorithme de choix du parent préféré. Une fois choisi, le nœud envoi à ce parent un message DAO pour signaler son choix (voir figure 19). Ces messages DAO contiennent des informations permettant à la racine de construire des chemins vers les nœuds feuilles du DODAG.

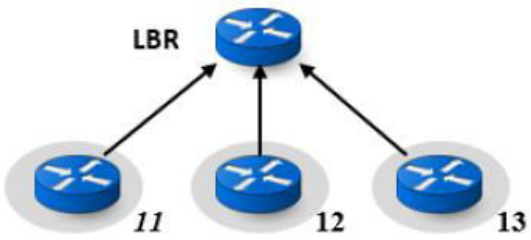

Figure 19:Choix du LBR comme nœud parent.

3. Troisième étape :

Les nœuds ayant choisi le LBR comme parent vont à leur tour diffuser des messages DIO en multicast dans leur voisinage. Les nœuds ne faisant pas encore parti du DODAG qui recevront ces messages DID, vont donc faire un choix de nœud parent puis répondre par des messages DAO indiquant leur attachement au DODAG (figure 20).

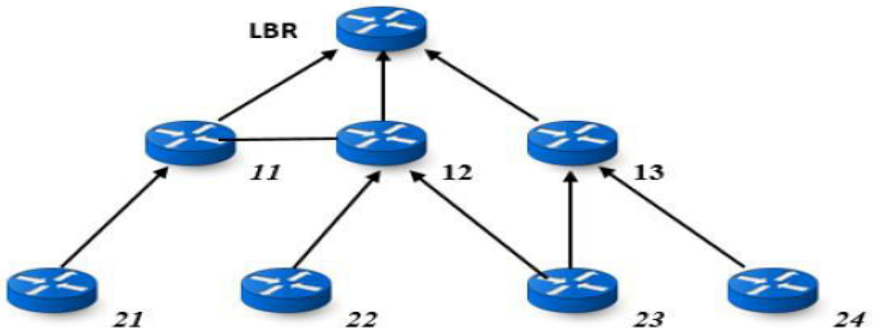

Figure 20:Poursuite de la construction du DODAG.

4. Quatrième étape :

Le processus de l'étape 3 se poursuivra jusqu'à ce que tous les nœuds du réseau se connectent au DODAG. La figure ci-dessous (figure 21) montre le DODAG final.

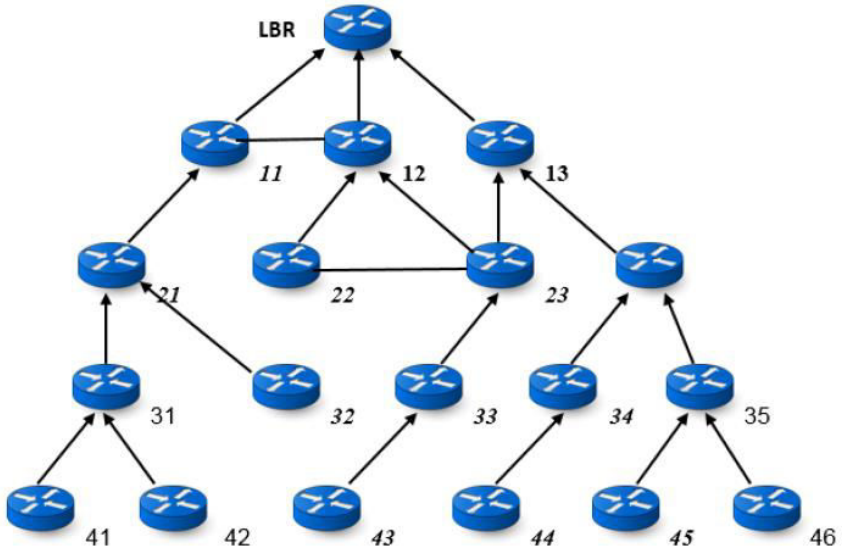

Figure 21:DODAG final construit par RPL.

Dans la construction du DODAG présentée ci-dessus, la fonction objective utilisée est la qualité du lien (LQ). A la réception du premier message DIO, chaque nœud va prendre sa décision de se joindre ou de ne pas se joindre au DODAG en construction. Cette décision s'effectue en se basant sur la fonction objectif diffusée dans le DODAG mais aussi :

- Des caractéristiques du DODAG indiquées par la racine dans les champs du message DIO
- De la valeur de la métrique évaluée entre la racine et le nœud

La construction du DODAG s'effectue par des messages de contrôles transportés dans des paquets ICMPV6 .

II.8. Les modes de fonctionnement du protocole RPL : [5]

En fonction de la capacité des nœuds en terme de mémoire et la taille éventuelle du réseau. Le protocole RPL offre deux modes de fonctionnement qui sont :

II.8.1. Mode Non-Storing :

Dans le fonctionnement en mode Non-Storing, seul la racine est en mesure de stocker des informations de routage. Les autres nœuds du réseau conservent uniquement les adresses de leur parent direct. Toutes les informations sur la structuration du DODAG sont transmises à la racine dans les messages DAO. En cas de besoin de router des données vers une destination quelconque, les nœuds transmettent ces données à la racine en passant par leur parent. La racine effectuera un routage à la source vers la bonne destination.

Dans ce mode, chaque nœud génère son message DAO puis l'envoie à la racine du DODAG à travers des communications multi-sauts. Le message DAO envoyé est une réaction à un message DIO reçu. Le temps entre la réception du message DIO et l'envoie du DAO dépend des paramètres de l'implémentation. Cependant, les spécifications du protocole RPL exigent que le délai entre l'émission de deux messages DAO successifs doit être inversement proportionnel au rang du nœud dans le DODAG. Selon cette exigence, les nœuds les plus proches de la racine vont générer moins de message DAO comparativement aux nœuds les plus éloignés.

La figure ci-dessous (Figure 22) présente un fonctionnement en mode non Storing :

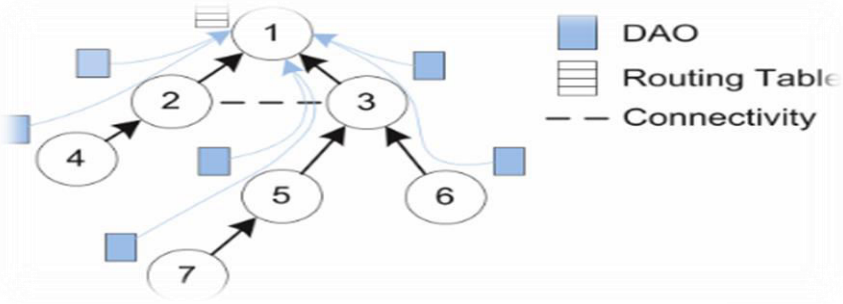

Figure 22:fonctionnement en mode non Storing.

A la réception d'un message DAO émis par un de ses fils, les nœuds intermédiaires vont inspecter le message pour s'assurer qu'il indique un chemin correct en direction du fils lui ayant transmis le message avant de le faire suivre vers la racine. La racine va collecter toutes les informations nécessaires pour la construction des tables de routage en direction des feuilles du DODAG. Après la Construction de la table de routage par la racine, tous les nœuds seront accessibles grâce à mécanisme de routage à la source.

Le routage à la source consiste à insérer les adresses des nœuds par les quels transitera le paquet dans l'entête

II.8.2 Mode Storing :

Dans ce mode, les nœuds intermédiaires sont en mesure de garder en mémoire des informations de routage puis de rediriger les données reçues vers la bonne destination en consultant les informations du routage.

Contrairement au mode Non-Storing, dans le fonctionnement en mode Storing, les messages DOA ne sont pas tous transmis à la racine. Chaque nœud transmet son message à son parent direct (Parent à un saut) qui maintient une table de routage à son niveau.

La figure Ci-dessous (figure23) présente le fonctionnement en mode Storing :

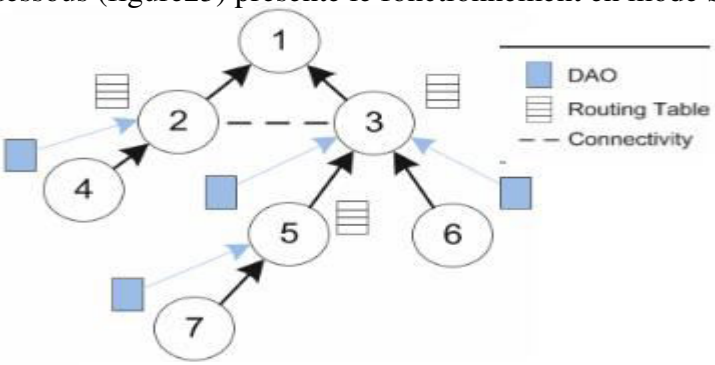

Figure 23:Fonctionnement en mode Storing.

Dans ce mode, lorsque la racine veut envoyer des données à un nœud de rang inférieur, elle n'effectue pas de routage à la source mais il envoie les données à ses fils situés à un saut qui se charge d'effectuer le routage jusqu'à la bonne destination. Pour envoyer des données à la racine, les nœuds feuilles envoient les données à leur parent direct qui effectuera un routage en se servant de la table de routage construite à l'aide des messages DAO. Il est également possible de faire une combinaison de ces deux modes de fonctionnement. Dans un tel scénario, les nœuds avec une faible capacité mémoire seront regroupés pour fonctionner en mode Non Storing.

II.9. conclusion :

Ce chapitre a été consacré à la description détaillée du protocole RPL. Après l'avoir défini, on a expliqué la procédure de construction du DODAG, la fonction objective, les types de messages de contrôles utilisés (DID, DAO et DIS) ainsi que le fonctionnement de l'algorithme Trickle utilisé pour minimiser ces messages en cas d'inconsistance dans le réseau. A la fin de ce chapitre, nous avons exposé les deux modes de fonctionnement (Non-Storing et storing).

Chapitre III : Simulation du protocole RPL et résultats

III.1. Introduction :

Dans ce chapitre, nous donnons une description de l'outil de simulation qui sera utilise pour simuler et analyser les performances du protocole RPL. Ce chapitre fait le bilan des scénarios de simulation et des résultats obtenus les.les derniers serons le sujet d'une discutions.

III.2. Systèmes embarqués pour internet des objets :

Les systèmes embarqués sont des systèmes d'exploitation prévus pour fonctionner sur des machines de petite taille, telles que des nœuds de capteurs. Les systèmes d'exploitation pour les nœuds de RCSF sont généralement moins complexes que les autres systèmes d'exploitation. Ceci à cause des exigences particulières des applications de réseau de capteurs et des contraintes de ressources des nœuds de capteurs. Il existe plusieurs systèmes d'exploitation pour les réseaux de capteurs sans fils comme : TinyOS [28], Contiki [14], MANTIS OS [3], LiteOS, RETOS, Nano- RK. Il y a certaines caractéristiques qui font la différence entre ces systèmes d'exploitation, par exemple : l'architecture, le modèle de programmation, la gestion de la mémoire, le langage de programmation. Le Tableau 1 fait une comparaison entre les caractéristiques de quelques systèmes d'exploitation. [20][12]

Parmi les systèmes d'exploitation actuels, nous décrivons les OS les plus utilisés dans le domaine scientifique qui sont : TinyOS, Contiki et MANTIS OS.

Caractéristique/OS	Architecture	Modelé de programmation	Gestion de la mémoire	Langage de programmation
TinyOS	Monolithique	Evénementielle	Mémoire statique	NesC
Contiki	modulaire	Evénementielle et multitâche	Mémoire Dynamique	C
MANTIS	Sous forme des couches	Multitâche	Mémoire dynamique	C

Tableau 1:Comparaison et caractéristiques de quelques systèmes d'exploitation [21].

III.2.1. Contiki :

Contiki est également un système d'exploitation open source. C'est un système configurable modulaire pour les réseaux d'internet des objets. L'architecture hybride du noyau Contiki autorise deux modes de fonctionnement : soit multitâche, soit basé sur les événements. Contiki est un système d'exploitation conçu pour prendre le moins de place possible, avec une faible empreinte mémoire. Pour cela, le code est écrit en langage C. Un système utilisant Contiki contient des processus, qui peuvent être des applications ou des services, c.à.d. un processus proposant des fonctionnalités à une ou plusieurs applications. La communication entre processus se fait par l'envoi d'événements. Le noyau Contiki reste, nativement, un système d'exploitation basé sur les événements. Pour obtenir le mode multitâche, une bibliothèque doit être installée. Les fonctions associées à cette bibliothèque n'accèdent pas directement à l'ensemble des ressources des objets sans fil. Elles doivent, dans certains cas, faire appel à la partie du noyau dédié à la gestion des événements. Cette structure à deux niveaux a pour conséquence une dégradation des performances du système quand le mode multitâche est activé.[14]

III.2.2. TinyOS :

TinyOS est un système d'exploitation open source pour les réseaux de capteurs sans fil et internet des objets qui trouve sa genèse au sein du laboratoire d'informatique de l'université de Berkeley et qui a été l'un des premiers systèmes d'exploitation conçus pour les réseaux de capteurs miniatures. En effet, TinyOS est le plus répandu des OS pour les réseaux de capteurs sans-fil. Il est capable d'intégrer très rapidement les innovations en relation avec l'avancement des applications et des réseaux eux-mêmes tout en minimisant la taille du code source en raison des problèmes inhérents de mémoire dans les réseaux des objets. Les applications pour TinyOS sont écrites en langage de programmation NesC (Network Embedded System C), une extension du langage programmation C. L'utilisation du langage NesC permet l'optimisation du code et par suite réduit l'usage de la mémoire à accès aléatoire (RAM). Un programme sous TinyOS ne doit comporter que les composants nécessaires à son exécution, ce qui réduit la taille du programme à insérer dans l'unité de traitement du des objets.[28]

III.2.3 MANTIS OS :

MANTIS (MultimodAl NeTworks of In-situ micro Sensor) OS apparu en 2005, a été conçu par l'université du Colorado. C'est un système d'exploitation léger et multitâche pour les capteurs, adapté aux applications où plusieurs traitements, chacun associé à un ou plusieurs processus, sont en concurrence pour accéder aux ressources du capteur sans fil. Il dispose d'un environnement de développement Linux et Windows. La programmation d'application sur MANTIS OS se fait en langage C. Son empreinte mémoire est faible : 500 octets en mémoire RAM et 14 kilo-octets en mémoire flash. C'est un système modulaire dont le noyau supporte également des entrées/sorties synchrones et un ensemble de primitives de concurrence. L'économie d'énergie est réalisée par MANTIS à l'aide d'une fonction de veille appelée sleep fonction qui désactive le capteur lorsque toutes les taches actives sont terminées. MANTIS est un système dynamique ; les modifications applicatives peuvent être réalisées pendant le fonctionnement. MANTIS apporte une compatibilité avec le modèle événementiel TinyOS à travers TinyMOS (MOS est la contraction de MantisOS), dont son noyau est équipé.[3]

III.3. Environnement de travail :

III.3.1. Contiki :

Contiki est un système d'exploitation open source, léger et flexible, qui est spécifiquement conçu pour les RCSFs et de petits dispositifs embarqués dédié à l'Internet des objets. Il a été conçu et mis au point par l'Institut suédois d'informatique (BICS) avec la collaboration du groupe de développeurs du monde universitaire et de l'industrie.

Contiki fournit des mécanismes qui aident à la programmation des applications d'objets intelligents. Il fournit. Des bibliothèques pour l'allocation mémoire, la manipulation des listes chainées et l'abstraction de communication. Développé en C avec toutes ses applications, et il est portable vers différentes architectures [29].

III.3.1.1. Architecture :

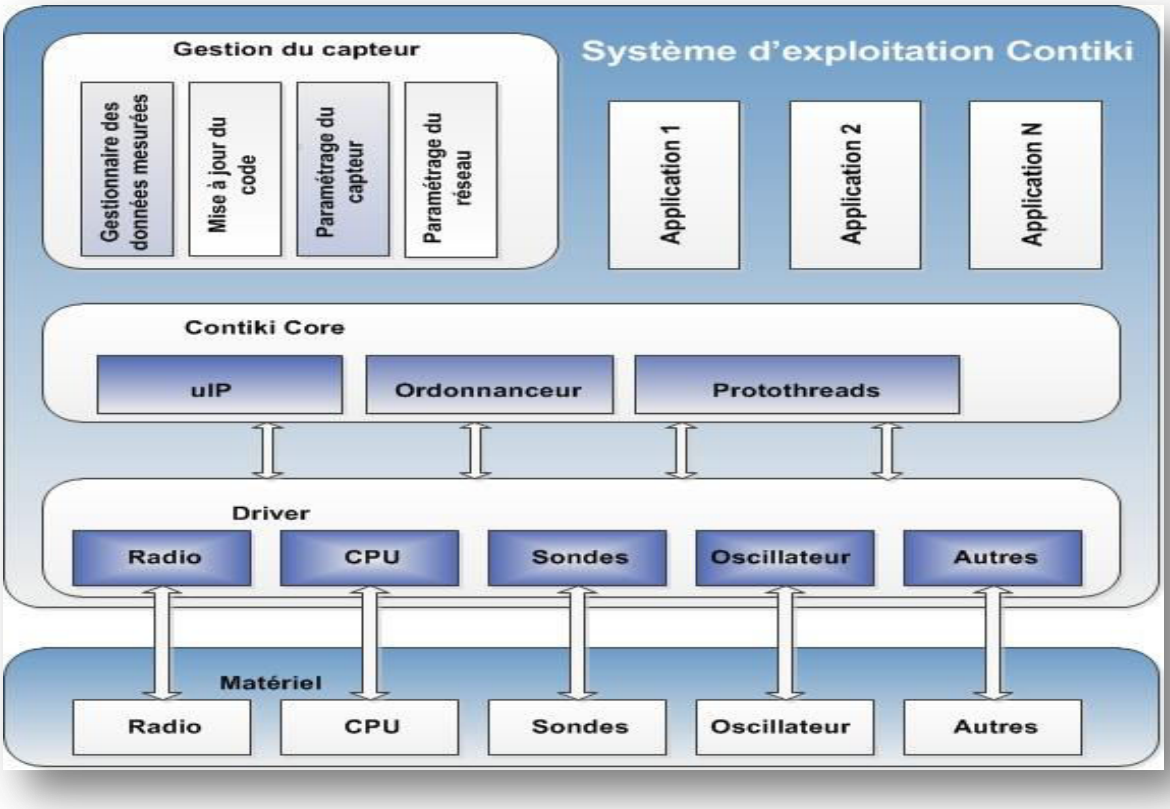

Figure 24:Architecture du système d'exploitation Contiki.

Contiki est constitué d'un noyau, de bibliothèques, d'un ordonnanceur et d'un jeu de processus. Comme tout système d'exploitation, son rôle est de gérer les ressources physiques tôles que le processeur, la mémoire, les périphériques informatiques (d'entrées/sorties) (Figure 24).

II fournit ensuite aux applications informatiques des interfaces permettant d'utiliser ces ressources. Conçu pour les modules de capteurs sans fil miniatures, il occupe peu d'espace en mémoire et permet une consommation électrique très faible [29]

III.3.2. ContikiRPL :

ContikiRPL est une implémentation du protocole RPL qui sépare la logique du protocole, les fonctions objectives de la construction du message et l'analyse des différents modules. Le module logique contikiRPL maintient et gère les informations du graphe DAG, construit et maintient un ensemble de parents pour chaque nœud dans un réseau, valide les messages RPL selon la spécification RPL et communique avec les fonctions objectives.

ContikiRPL crée les tables de transfert pour ulPv6 et ne participe pas dans les décisions de transfert effectuées seulement par ulPv6. La mise en œuvre de ContikiRPL fonctionne dans un émulateur de COOJA sans aucune modification de la plate-forme [16].

La structure de ContikiRPL est schématisée dans la figure suivante

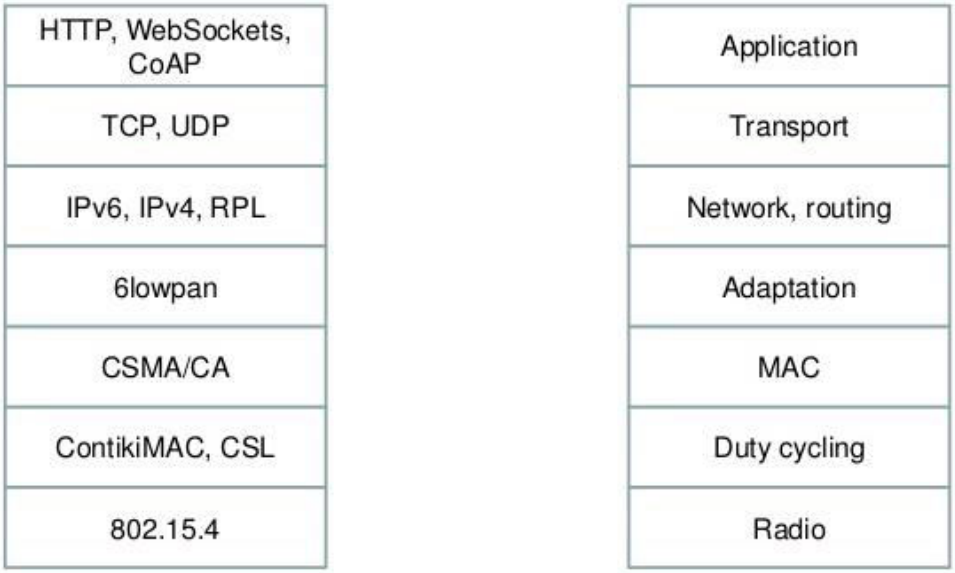

Figure 25:La structure de ContikiRPL.

Le répertoire RPL contient plusieurs fichiers d'exécution pour différentes fonctions RPL :

- RPL-timers.c : responsable de l'envoi des mises à jour périodiques.
- RPL-dag.c : contient les fonctions nécessaires pour les manipulations de DAG.
- rpl.c : comporte les principales fonctions de RPL pour modifier les entrées de routage IPV6 (purger un itinéraire, ajouter une route, etc.).
- Rpl-icmp6.c : fonction de gestion de messages ICMP pour le protocole RPL.
- Rpl-of0.c : est la fonction objective« FO » (modèle de fonction objective fonction implémentant la fonction de routage basée sur les sauts).
- Rpl-mrhof.c : contient la fonction objective basée sur les métriques ETX et batterie (énergie).
- Rpl-ext-header.c : pour la gestion des-en-tètes d'extension pour ContikiRPL .

III.3.3. Cooja :

Cooja est un simulateur de réseau pour Contiki. Il permet de simuler les réseaux de capteurs (Internet des objets) quel que soit leur taille. Les capteurs peuvent être émules : au niveau du matériel, ce qui est plus lent, mais permet un contrôle précis du comportement du système, ou à un niveau moins détaillé, ce qui est plus rapide et permet la simulation de grands réseaux (réseaux plus large) [27].

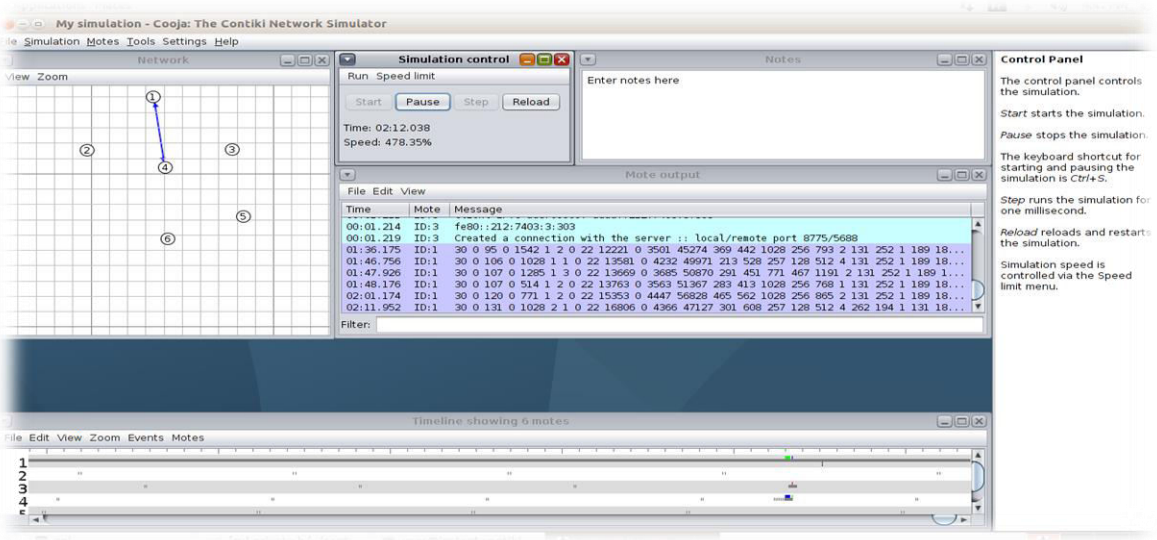

Figure 26:Capture d'image de Cooja.

III.4. Les différents outils nécessaires :

Nous allons travailler sous Ubuntu, pour cela, nous utilisons les outils suivants :

III.4.1. Ant :

Ant est un logiciel créé par la fondation Apache qui vise à automatiser les opérations répétitives du développement de logiciel telles que la compilation, la génération de documents (Javadoc) ou l'archivage au format JAR, à l'instar des logiciels Make .

Pour installer ce premier paquet, il faudra utiliser la commande suivante dans un terminal :

```
sudo apt-get install ant
```

III.4.2. Le compilateur MSP430 :

Cooja simule les communications réseau en utilisant l'émulateur MSPSim pour émuler finement (au niveau des instructions) l'exécution d'un programme sur une plateforme basée sur un processeur MSP430.

Il nous faudra donc un compilateur adapté à cette architecture.

La commande suivante permet son installation :

```
sudo apt-get install gcc-msp430
```

III.4.3. Le JDK :

Apache Ant étant écrit en Java, il a besoin d'une machine virtuelle (JVM : Java Virtual Machine) pour fonctionner.

Nous installerons donc Open-JDK (Java Développement Kit) pour pouvoir l'utiliser.

```
sudo apt-get install openjdk-8-jdk-headless
```

III.4.4. Téléchargement et installation de Contiki :

Contiki 3.0 peut être téléchargé à l'adresse suivante :

```
https://sourceforge.net/projects/contiki/files/Instant%20Contiki/Instant%20Contiki%203.0/InstantContiki3.0.zip/download
```

III.5. Changer les paramètres dans la simulation Cooja :

Pour lancer un mode non-storing dans cooja , nous remplaçons "RPL_CONF_MOP" par "RPL_MOP_NON_STORING" Dans le fichier " rpl-private " , qui est situé dans le chemin suivant : contiki/core/net/rpl/rpl-private.

III.6. Démarrage de l'application:

Accédez au dossier Contiki (contiki 3.0), puis accédez au répertoire / tools / cooja.

Exécutez la commande sudo ant run pour ouvrir une interface graphique cooja.

```
$ cd contiki3.0/tools/cooja
```

```
$ sudo ant run
```

Puis, on va créer une nouvelle simulation à partir du menu File / NewSimulation

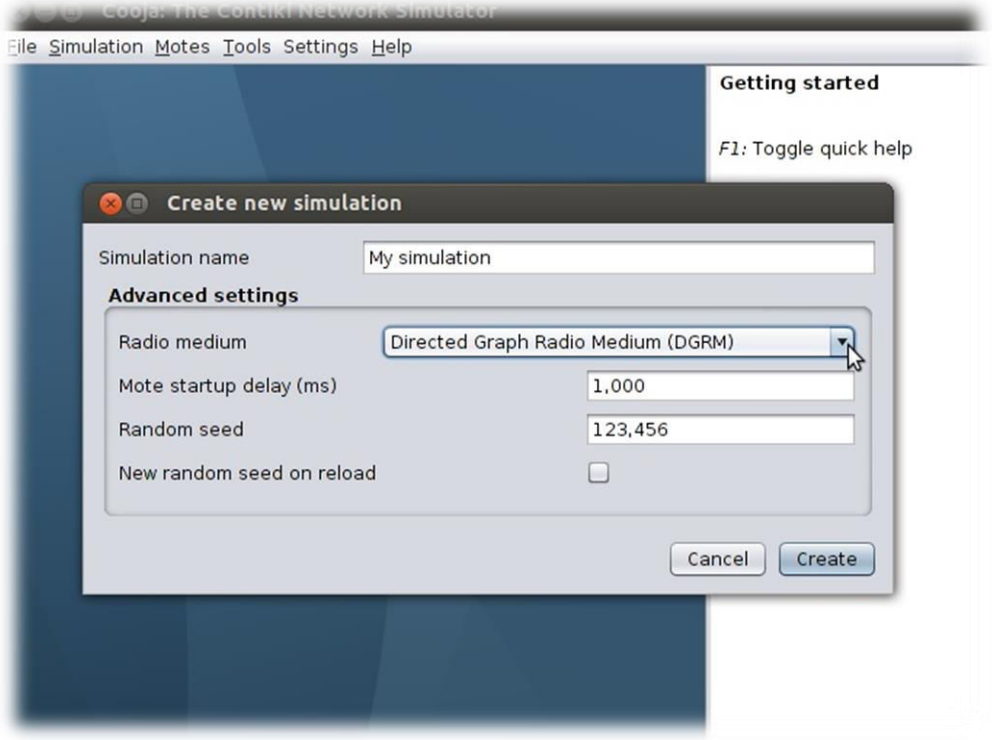

Figure 27:Créer une nouvelle simulation.

Donnez un nom à la simulation dans le champ Nom de la simulation. Choisissez le média radio à diagramme dirigé (Directed Graph Radio Medium) (DGRM) dans le menu déroulant de Radio Medium sous Paramètres avancés. Cliquez sur le bouton Créer.

La nouvelle simulation est créée et elle ouvrira plusieurs fenêtres comme indiqué.

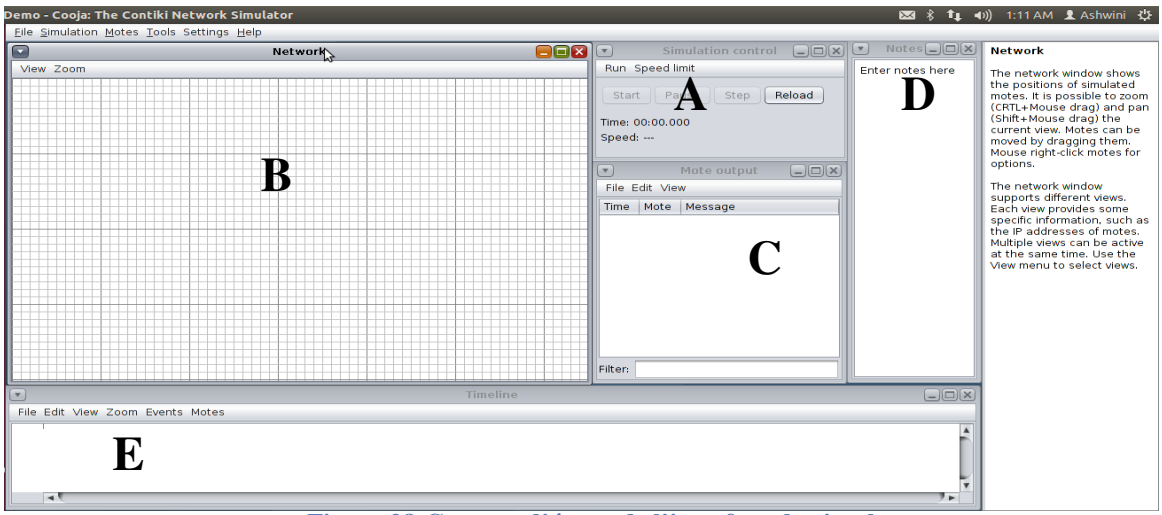

Figure 28:Capture d'écran de l'interface du simulateur.

A. La fenêtre Timeline : affiche tous les événements de communication dans la simulation, très pratique pour comprendre ce qui se passe dans le réseau.

B. La fenêtre Network : en haut à gauche de l'écran, nous montre tous les nœuds dans le réseau simulé.

C. La fenêtre Mote Output, sur le coté droit de l'écran, nous montre toutes les impressions sur le port série de tous les nœuds.

D. La fenêtre Notes en haut à droite est I 'endroit où nous pouvons mettre des notes pour notre simulation.

E. La fenêtre Simulation control : est où nous pouvons lancer, mettre en pause et charger notre simulation.

III.7. Ajouter des nœuds pour créer un réseau :

III.7.1. Ajouter le Nœud "Sink" :

Pour ajouter un Nœud de type Sink, le code udp-sink de l'exemple rpl-collect (/ examples / ipv6 / rpl-collect /) est utilisé. Cependant, vous pouvez télécharger n'importe quel code que vous souhaitez implémenter en fonction de votre application. Cliquez sur le bouton Compiler. Un bouton "Creat" apparaît sur une compilation réussie qui ajoute un nombre de Nœuds comme souhaité dans le réseau.

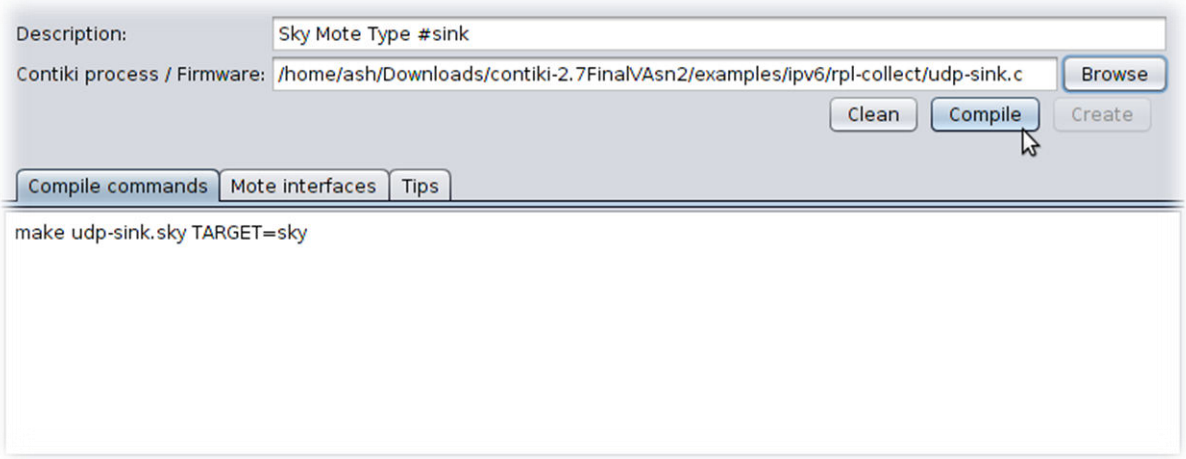

Figure 29:Ajouter un nœud de type « Sink ».

III.7.2. Ajouter des Nœud "Sender" :

Pour ajouter un Nœud de type Sender.est utilise, le code udp-sender de l'exemple rpl-collect (/ examples / ipv6 / rpl-collect /)

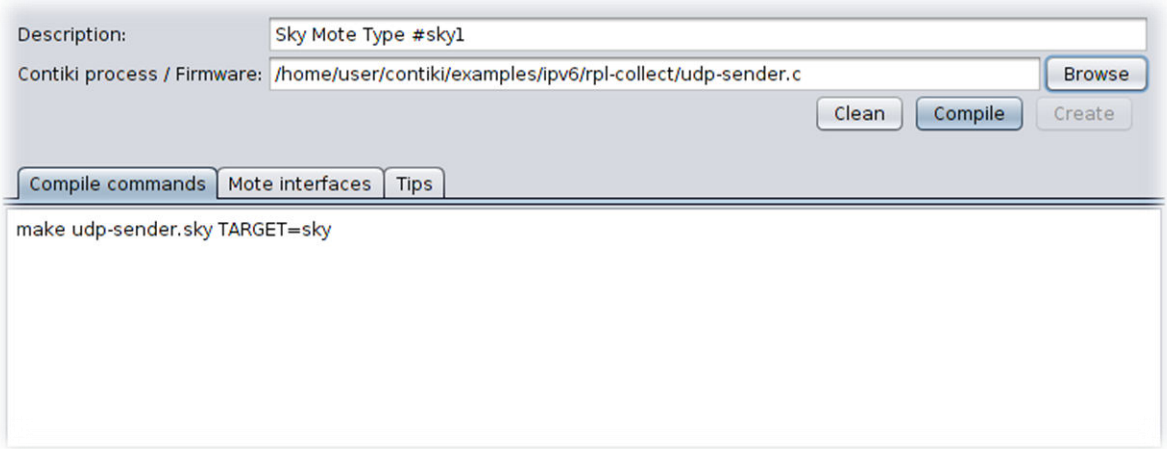

Figure 30:Ajouter un nœud de type « Sender ».

III.8. Paramètres estimés (mesurés) :

Nous ferons les études suivantes en changeant le nombre de nœuds sur les outils fournis par cooja.

- L'inertie : La consommation énergique de l'énergétique de tous les nœuds du réseau.
- Transmission (TX) : Nombre de messages envoyés à travers tous les nœuds.
- Réception (RX) : Nombre de messages reçus à travers tous les nœuds.

III.8.1. Énergie :

Nous étudions ici les performances des objets par rapport à l'énergie utilisée et consommée tout au long de la simulation.

Pour faire cela, nous avons un outil qui est PowerTracker, permettant de connaître le taux et la durée d'utilisation des fonctions radio des objets.

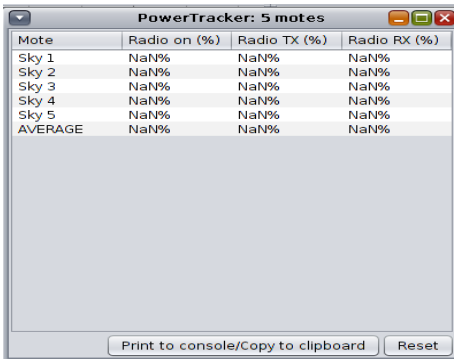

Figure 31:Outil d'interface PowerTracker.

La consommation d'énergie a un lien direct avec la puissance. Dans notre cas la puissance dépend d'un paramètre de temps T. Ce dernier est sous forme de pourcentage lié à l'utilisation de la radio en émission et en réception. La consommation d'énergie reste proportionnelle à l'utilisation de la radio. Les équations suivantes expriment l'utilisation de la puissance et de l'énergie dans le cas de ce réseau créée par Contiki [8]:

Puissance (mW) = (Ton * ON + Trx *RX + Ttx * TX) * 3V

ON = 2 mW.

RN = 17.7 mW.

XN = 20 mW.

Énergie (Joule) = Puissance x temps de l'exécution de la simulation (secondes).

T (secondes) = temps total de l'utilisation de la fonction radio voulu (radio on ; rx ; tx).

La tension opérationnelle des nœuds Tmote Sky est de 3V.

III.8.2. Transmission(TX) et Réception (RX) :

Les envois et la réception des paquets se font grâce au routage RPL qui permet entre autres de relayer les paquets entre capteurs. Ce routage se fait grâce aux différents paquets RPL que l'on peut citer :

Message d'information DIO

Message d'information DAO

Message d'information DIS

Ce routage permet à tous les équipements de connaître l'ensemble des routes disponibles pour acheminer un paquet d'un capteur à un autre.

On peut calculer le nombre total des messages envoyés/réceptionnés, grâce à l'outil "Mote output du simulateur Cooja

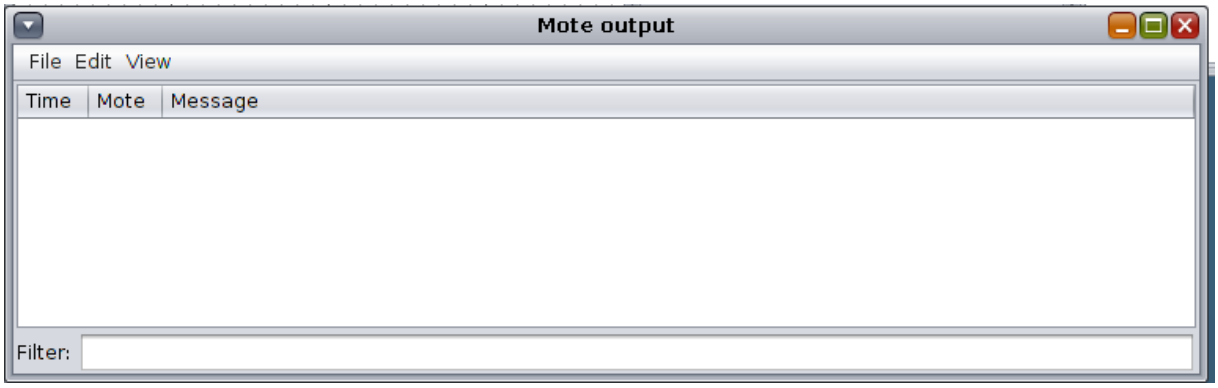

Figure 32:Outil d'interface Mote output.

III.9. Scénario :

III.9.1. Paramètres de simulation :

paramètres	valeurs
simulateur	Cooja
temps	5 min
Nombres des nœuds	5, 10, 15, 20, 30
Type de nœuds	SKymote
TX range	32 m
RX range	32 m
Consommation énergétique	CPU 1.8 mW
	Écouter 20.0 mW
	Transmettre 17.7 mW

Tableau 2:Paramètres de simulation .

III.9.2. Résultats :

▪ **Energie Vs nombre de nœuds :**

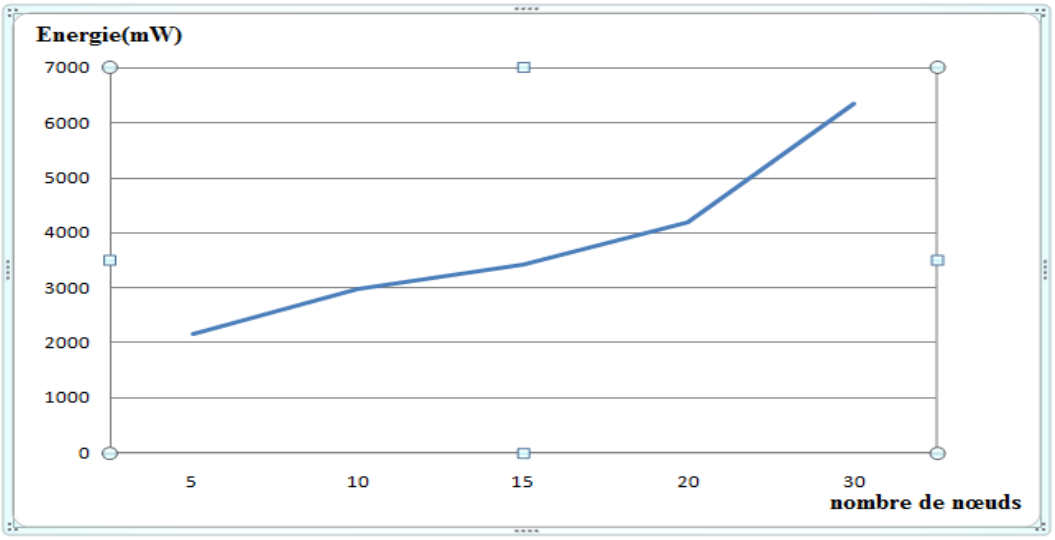

Figure 33:Consommation énergétique totale Vs Nombre de nœuds.

Cette figure représente la consommation énergétique totale pour le mode non-storing en fonction du nombre de nœuds. Nous cotations que la consommation augmente considérablement en utilisant 30 nœuds ou plus. Ceci peut être interprété par un taux supérieur du trafic de contrôle dans le mode non storing supérieur, pour gère les changements peuvent se produiront (dans la topologie).

▪ **TX Vs nombre de nœuds :**

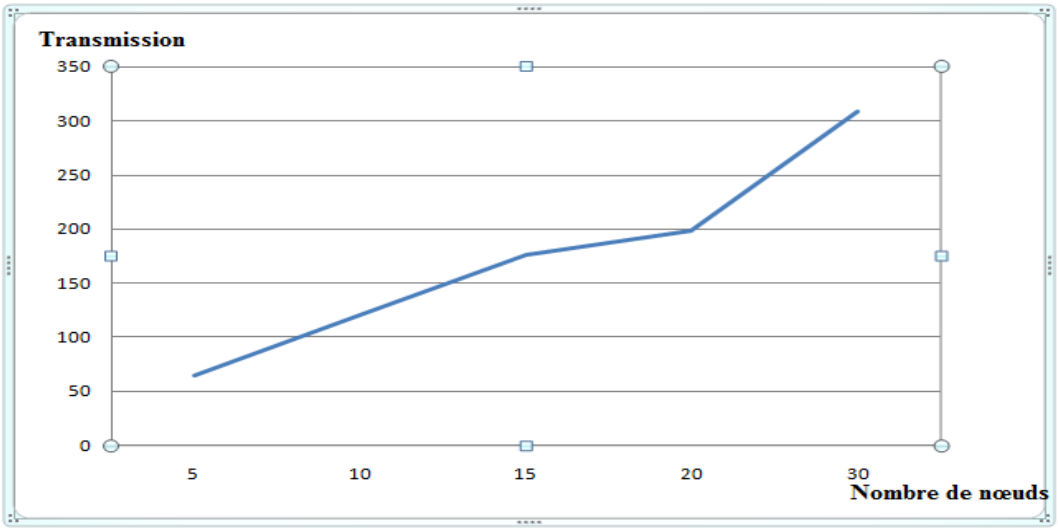

Figure 34:Transmission Vs Nombre de nœuds.

Cette figure représente le nombre de transmission (TX) pour le mode non-storing en fonction du nombre de nœuds. Nous remarquons que ce nombre est proportionnellement moyen par rapport du temps pris dans la simulation. Ceci peut être expliqué par la présence de plus de retransmission des paquets.

▪ **RX Vs nombre de nœuds :**

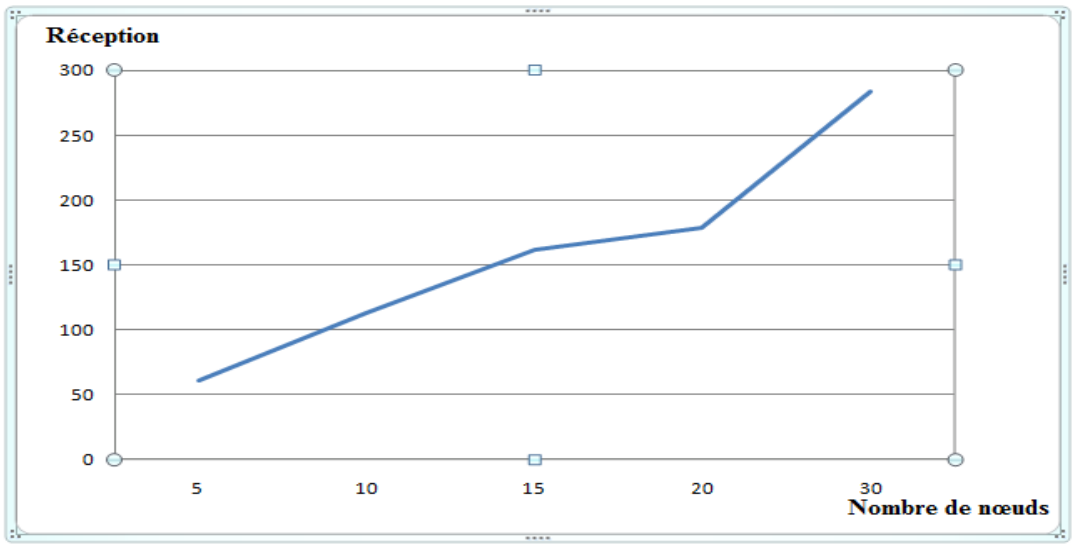

Figure 35:Réception Vs Nombre de nœuds.

Cette figure représente le nomre de réceptions (RX) pour le mode non-storing en fonction du nombre de nœuds. On remarque que ce nombre dans le mode non-storing est moyen eu proportion du temps pris dans la simulation. Ceci peut être expliqué par présence de plus de ré-réceotion des paquets.

III.10. Conclusion :

Dans ce chapitre, nous avons expliqué comment installer et madéfier les paramétrer du simulateur Cooja (disponible pour le système Contiki). De même, nous avoir crée un scénario pour le protocole RPL, afin d'étudier les performances de ce protocole par rapport à : temps, Nombres des nœuds, TXrange, RX range et Consommation énergétique (CPU, Écouter, Transmettre).

Nous avoir obtenu des résultats, qui out été tracés et qui étaient le sujet d'une discutions. Ces même résultats vont être comparés avoir les résultats obtenus su utilisant l'outil UPPAAL, le chapitre suivant, décrire la modélisation et la simulation du protocole RPL sous UPPAAL.

Chapitre IIII : Modélisation et vérification du protocole RPL sous UPPAAL

IIII.1. Introduction :

L'objectif de ce chapitre est la modélisation du protocole RPL sous UPPAAL, afin d'étudient ses performances. Cette modélisation est basée sur les automates temporisés. Après la modélisation de ce protocole, nous simulons le fonctionnement de ce dernier, et nous utilisons les logiques temporelles pour vérifier certaines propriétés. Nous terminons le chapitre par une étude de performances qui sera basée sur : la consommation d'énergie, le transmission, la réception.

IIII.2. UPPAAL : [2]

UPPAAL: est un outil de modélisation et de vérification de systèmes temps-réel qui peuvent être représentes sous forme de processus non-déterministes et interactifs ayant une structure de contrôle fine et des horloges avec des valeurs réelles. Un modelé d'un système dans UPPAAL [4] se compose d'un réseau de processus décrits par des automates temporisés étendus communiquant par des canaux ou des structures de données partagées. La conception de l'outil UPPAAL est basée sur l'architecture Client/serveur, comme le montre la Figure (Architecture de UPPAAL) Le serveur est représenté par la machine UPPAAL développée en C++ et le client est représenté par l'interface GUI développée en JAVATM. La communication entre la machine UPPAAL et son interface graphique est établie a travers des protocoles internes.

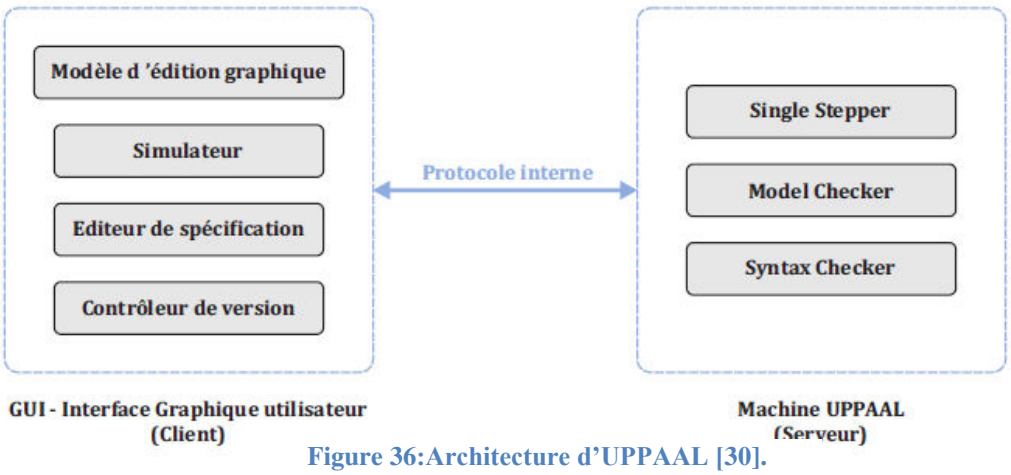

Figure 36:Architecture d'UPPAAL [30].

L'outil UPPAL assure plusieurs fonctionnalités [30] :

- atg2ta : C'est un compilateur qui traduit la représentation graphique (.atg) d'un réseau d'automates temporises a une représentation textuelle (.ta).
- Hs2ta : Ce programme permet de transformer des automates hybrides linéaires, ou la valeur d'horloges sont définie par un intervalle, en réseaux d'automates temporises.
- checkta : Etant donne une représentation textuelle (.ta), ce programme permet d'appliquer une vérification syntaxique de la description du système.
- verifyta : Ce programme est le noyau de vérification UPPAAL. Il permet de décider si une propriété donnée est satisfaite ou non et générer un contre-exemple dans le cas de violation de Cette propriété.

IIII.2.1. Syntaxe :

Un modèle UPPAAL est basé sur les automates temporisés. La description d'un modèle comporte de trois parties [25]:

- Les déclarations globales et locales.
- Les processus sous forme d'automates temporisés.
- La définition du système.

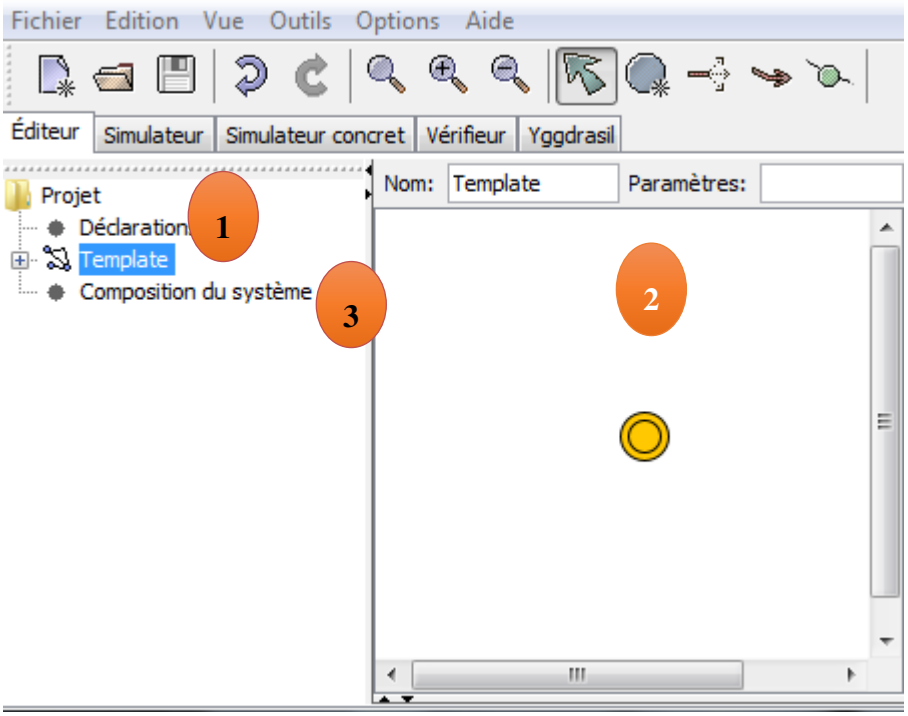

Figure 37:Interface Uppaal.

IIII.2.2. Déclarations :

Les déclarations peuvent être locales ou globales. UPPAAL permet de déclarer des horloges (type clock), des entiers (type int), des booléens (type bool), des canaux de synchronisation (type chan), des tableaux, des structures de données (type struct), et de définir des types de données (typedef).

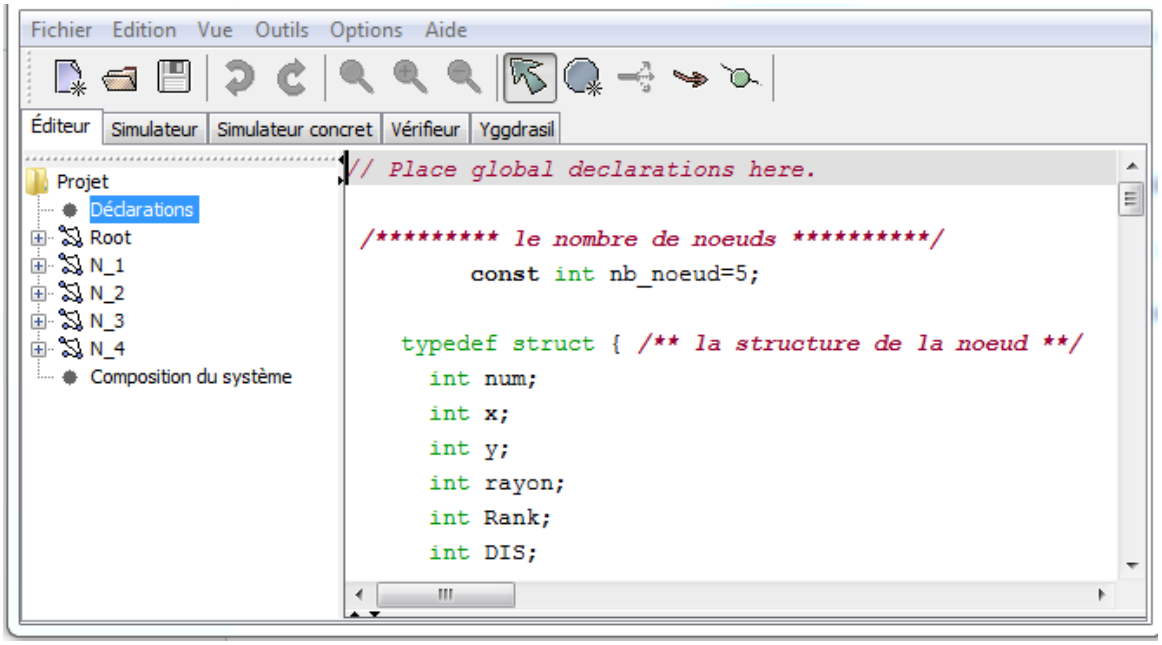

Figure 38:Interface Déclarations pour Uppaal.

IIII.2.3. Locations :

Un automate temporisé est considéré comme un graphe à états. Les locations représentent les états dans ce graphe. Elles se communiquent via les transitions. Un identifiant optionnel peut être associé à chaque location afin de l'identifier. Elles peuvent être étiquetées par des invariants. Un invariant peut être une expression composée d'un ensemble de contraintes sur des horloges, une différence entre des horloges ou des expressions booléennes ne faisant pas intervenir des horloges.

UPPAAL distingue trois types de locations :

- ✓ Location initiale : Chaque automate doit avoir une location initiale représentée pas un double cercle.
- ✓ Location urgente : La location urgente ne permet pas de faire passer le temps quand processus s'y trouve. Sémantiquement, une location urgente est équivalente a une

location dont l'invariant est x <= 0 ou x est une horloge remise à zéro sur toutes les transitions entrantes de la location en question.

- ✓ Location engagée : La location engagée ne permet pas l'écoulement du temps quand un processus s'y trouve. Elles sont utiles pour assurer l'atomicité de la synchronisation entre trois processus ou plus. Elles sont plus prioritaires que les locations urgentes.

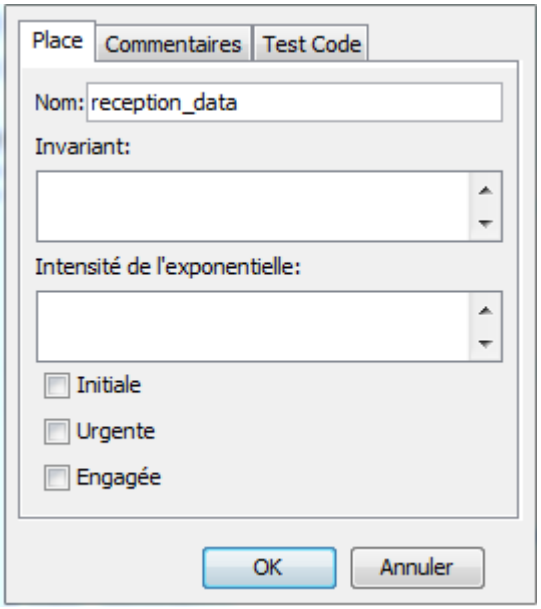

Figure 39:Interface Locations pour Uppaal.

IIII.2.4. Transitions :

Les locations sont connectées entre elles par les transitions. Ces dernières sont étiquetées par des gardes, des synchronisations et des mises à jour des variables et des fonctions.

IIII.2.5. Garde :

Une transition est franchie si et seulement si la garde spéciale est satisfaite.

Les gardes expriment des contraintes d'horloges ou de données nécessaires pour franchir une transition. Formellement, les gardes peuvent être spéciales comme suit : Soient x et y sont deux horloges ou deux variables de données. Une gardées de la forme x r n ou (x - y) r n ou r appartient {≤, ≥, =, >, <} et n est une expression entière.

IIII.2.6. Synchronisation :

Les différents automates dans un modelé UPPAAL peuvent se synchroniser via des canaux de synchronisation permettant l'envoi et la réception d'un message. Deux transitions dans deux processus différents peuvent se synchroniser si et seulement si les gardes des deux transitions sont satisfaites et elles admettent deux étiquettes de synchronisation a1 ! (envoi) et a2 ? (réception) ou a1 et a2 représentent le même canal de synchronisation. Quand deux processus se synchronisent, les deux transitions sont franchies au même temps. Les mises à jour de la transition portant l'étiquette a1 ! Se font avant celles de la transition portant l'étiquette a1 ?*.

Il existe deux types de canaux de synchronisation :

- les canaux de synchronisation binaires qui permettent de synchroniser deux processus via un seul canal de synchronisation a (au moyen de deux actions a! et a?)
- les canaux de synchronisation de diffusion (broadcast) qui permettent de synchroniser un processus avec plusieurs autres processus. Une transition avec une étiquette de synchronisation a!, avec a est un canal broadcast, peut synchroniser avec chaque transition ayant une étiquette de synchronisation a? Et dont les gardes sont satisfaites.

IIII.2.7. Mise à jour :

L'exécution de la mise à jour sur une transition permet de changer l'état du système. Une mise à jour est une liste d'expressions séparées par des virgules et exécutées dans un ordre séquentiel.

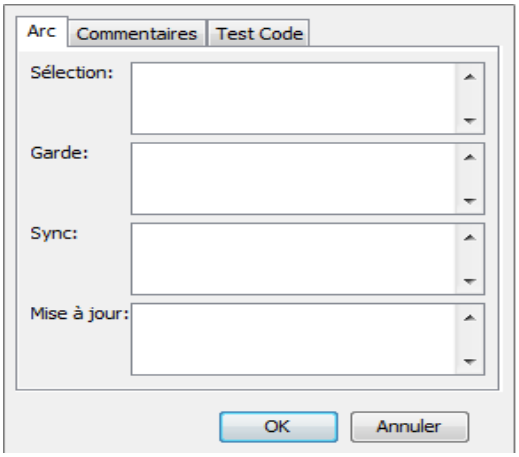

Figure 40:Interface Transitions pour Uppaal.

IIII.3. Modélisation le protocole RPL :

Nous proposons un modèle formel conforme au fonctionnement de protocole RPL (expliqué dans deuxième chapitre), il est claire que le fonctionnement du "Sink" se diffère du fonctionnement du "Sender". Par conséquent, nous avoir crée deux modèles (automates) qui modélisent le fonctionnement du Sink d'un coté et du Sender(s) de l'autre coté.

IIII.3.1. Modélisation du Root (Sink):

Les variables et les méthodes utilisées pour modéliser le root(Sink) par un automate temporisés sont :

IIII.3.1.1. Les variables:

Nous avons défini un ensemble de variables comme suit :

Const Int Root_num: Numéro du root.

Int Root_x: Coordonnée de la x.

Int Root_y: Coordonnée de la y.

Int Root_rayon: Rayon du root pour calculer la zone de transmission.

Int Root_Rank: Le rang du root est toujours zéro.

Int Root_DIS: le numéro du Sender qui a envoyé le message DIS .

Int Root_DIO: le numéro du Sender qui a envoyé le message DIO.

Int Root_DAO: le numéro du Sender qui a envoyé le message DAO.

Int Root_DAO_ACK: le numéro du Sender qui a envoyé le message DAO_ACK.

Int Root_DATA: le numéro du Sender qui a envoyé le message DATA.

Int Batterie : capacité de la batterie.

Int tab voisins [nb_noeud] : table de voisinage.

Int nb_voisins : Nombre de voisins.

Bool Root_connecte: Variable pour vérifier la connexion du root au réseau.

IIII.3.1.2. Les méthodes :

Void routage () : La fonction responsable de trouver le meilleur chemin vers la destination qui est basée sur le paramètre (distance).

Void rece_data () : Une fonction qui reçoit un message DATA et identifie la destination.

Void Mise_a_jour () : Une fonction pour savoir quels messages ont atteint le root.

Void prep_diffuse_Root () : Une fonction qui calcule la distance entre le root et les nœuds pour crée la table de voisinage.

IIII.3.1.3. L'automate temporise :

- **La Modélisation générale (root):**

La Figure Suivant décrire un modèle globale du root.

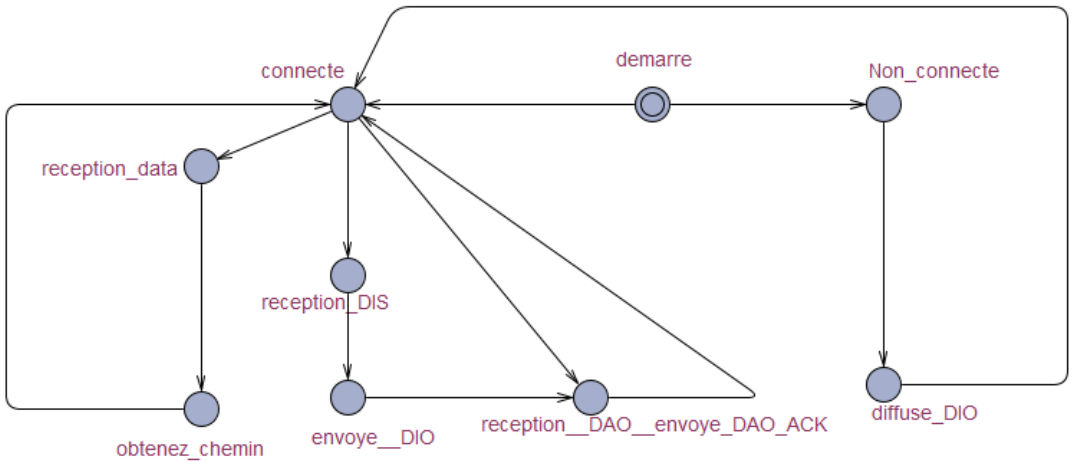

Figure 41:Modélisation générale de l'automate temporise pour le root.

Lors du démmarge du root (état "demarre"), le root a deux possibilités soit il se connecte (il passe à l'état "Connect"),ou il reste déconnecte (et il passe à l'état "Non_connect").

- État "Non_connecte" :Lorsque le root est déconnecté, il diffuse un message DIO pour connaitre ses voisins. la reconnaissance des voisins se faut par la méthode Void prep_diffuse_Root ().

  Il change son état vers l'état connecté (état connect) tout eu changeant la valeur de la variable Bool Root_connecte de "false" à "true".
- État "connecte" : Quand il est connecté à un réseau, il peut effectuer trois fonction:

1. Recevoir le message DATA et connaître sa destination en utilisant la fonction « Void rece_data () »,et exécute la fonction «Void routage () » pour trouver le meilleur chemin vers la destination, puis il revient à l'état "connecte"
2. Recevoir le message DIS des nœuds qui viennent de se connecter au réseau ou qui viennent de rejoindre le réseau après une déconnexion, Et envoyez-leur un message DIO, puis recevez un message DAO et envoyer un message DAO_ACK pour se connecter.
3. Recevoir le message DAO et envoyer un message DAO_ACK.

La modélisation finale (détaillée du root) :

La Figure suivante illustre la modélisation détaillée dans le cas non connecté pour le root :

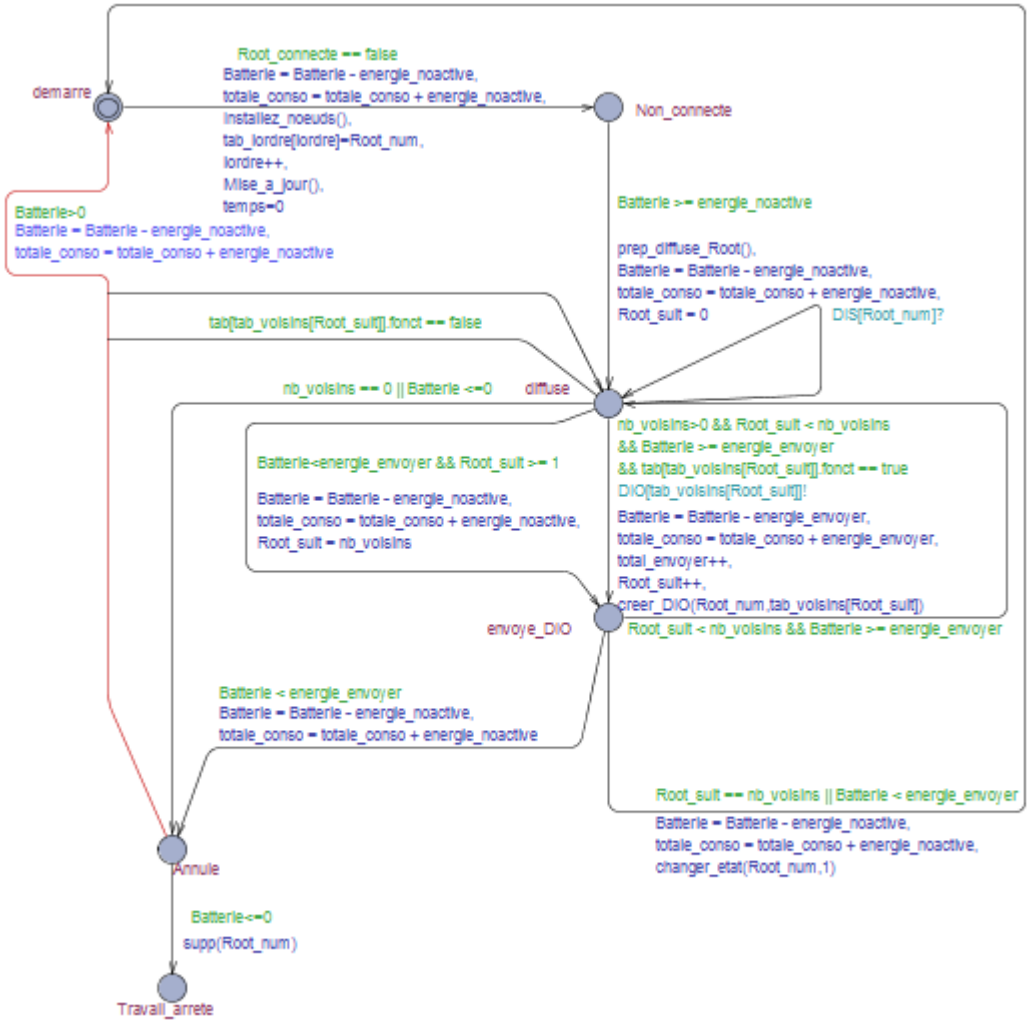

Figure 42: La modélisation (détaillé) dans le cas non connecté pour le root.

La Figure suivante illustre la modélisation détaillée dans le cas connecté (le root) :

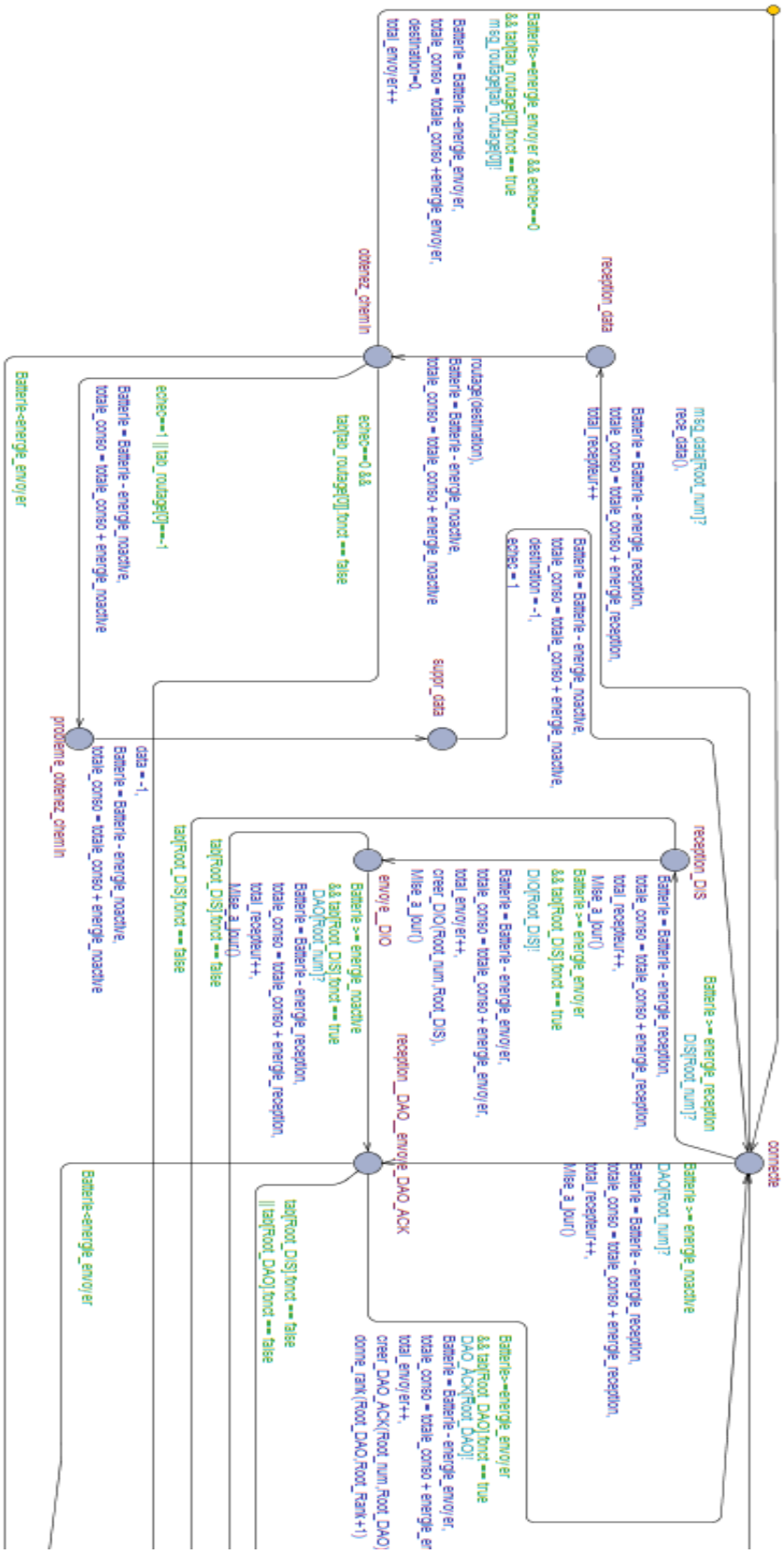

Figure 43:La modélisation (détaillé) dans le cas connecté.

Pour avoir une modalisation du protocole nous ajoutons des élément à la modélisation globale déjà vue ces élément sont :

- ✓ Garde : elles expriment les contraintes sur les transition. par exemple :

Avant qu'une émission ou une réception se faite, énergie doit être considérée pour savoir si le taux de la batterie est suffisait pour une telle opération. Cette contrainte, peut être exprimée par la garde: Batterie >= energie_reception. Un autre exemple, illustre les interaction avec les autres nœuds peut être exprimé par la garde: tab [tab voisins [Root_suit]].fonct == true. Cette garde a pour rôle de vérifier si le nœud en question est fonctionnel ou non .

- ✓ Synchronisation :

En cas d'envoi ou de réception, vous devez ajouter une synchronisation

Exemple : DAO [Root_num]? , DIO [Root_DIS]!.

- ✓ Mise à jour :

Si vous modifiez les valeurs des variables ou exécutez des fonctions

Exemple :prep_diffuse_Root (),

Batterie = Batterie - energie_noactive,

Totale_conso = totale_conso + energie_noactive,

Root_suit = 0

IIII.3.2. Modélisation du nœud :

Les nœuds(Sender) seront aussi modélisés par des automates temporisés. Nous devons aussi définir les variables et les méthodes utilisées :

IIII.3.2.1. Les variables:

Const int node_num : Numéro du nœud.

Int node_x : Coordonnées x du nœud.

Int node_y : Coordonnées y du nœud.

Int node_rayon : Rayon du nœud pour calculer la zone de transmission.

Int node_Rank : Le rang du nœud.

Int node_DIS : le numéro du Sender qui a envoyé le message DIS.

Int node_DIO : le numéro du Sender qui a envoyé le message DIO.

Int node_DAO: le numéro du Sender qui a envoyé le message DAO.

Int node_DAO_ACK: le numéro du Sender qui a envoyé le message DAO_ACK.

Int node_DATA: le numéro du Sender qui a envoyé le message DATA.

Int tab voisins [nb_noeud] : table de voisinage.

Int nb_voisins : Nombre de voisins.

Int Batterie : capacité de la batterie .

Bool node_connecte : Variable pour voir si la racine est connectée au réseau.

Int tab parent [nb_noeud] : Une table avec des parents (nœuds) que vous pouvez transmission.

Int nb_parent : Nombre de parents.

IIII.3.2.2. Les méthodes :

Voidcreation_data ():Créer un message DATA.

Void Mise_a_jour () : Une fonction pour savoir quels messages ont atteint le noeud.

Void prep_diffuse_Root () : Une fonction qui calcule la distance entre le nœud et les nœuds pour spécifier les voisins.

Void Verifiez_destination () :Lorsque un message arrive à un nœud ,Cette fonction vérifie si le ce nœud est le nœud de destination finale ou non.

IIII.3.2.3. L'automate temporise :

- **La Modélisation générale (nœud):**

La Figure Suivant montre une modélisation générale de l'automate temporise pour un nœud :

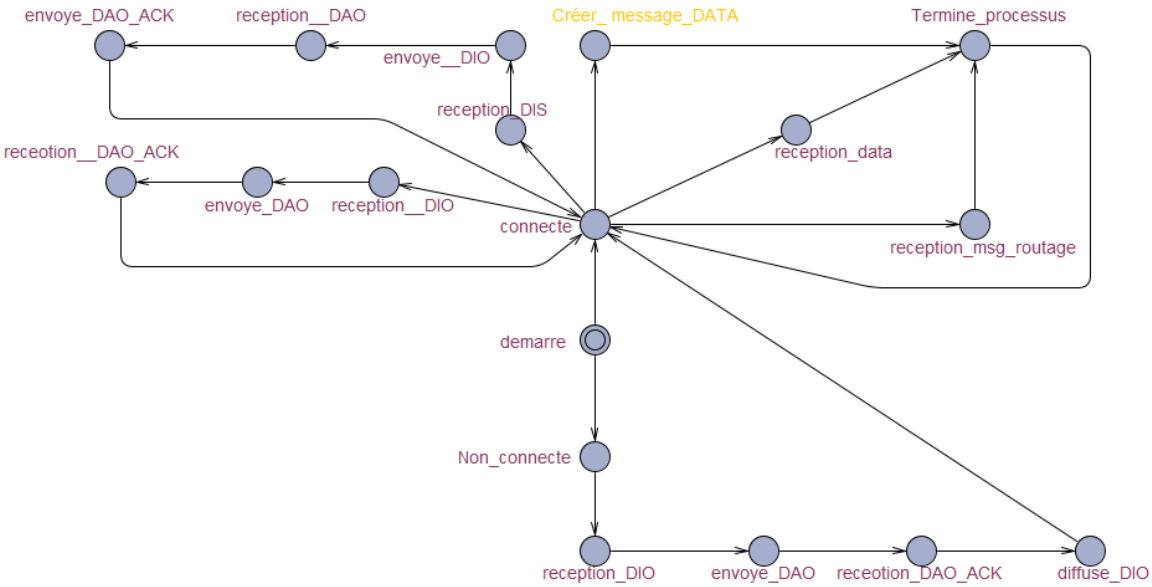

Figure 44:Modélisation générale de l'automate temporise pour nœud.

Lors du démarrage d'un nœud (état "demarre") ce nœud doit passer par l'état "non_connecte" pour recevoir message DIO de ses pères. Une fois ce message est réceptionné, le nœud envoie un acquittement, et diffuse lui même un message DIO à ses enfants et il passe à l'état "connecte".

Si un nœud arrive à se connecter (état "connecte"), il peut faire:

1. Créer un message data, et ajoutez le numéro de destination dans ce message.
2. Recevoir un message data de ses enfants et l'envoie à ses parents.
3. Recevoir un message DIS à partir des nouveaux nœuds, des messages de contrôle (DIO, DAO, DAO_ACK) seront échangés pour communiquer avec eux
4. Recevoir un message de routage qui vient du root, ensuite, il vérifie s'il s'agit de la destination ou non. Si c'est la destination, il supprime le message, et s'il n'est pas la destination, il termine l'opération de routage.
5. Recevoir un message DIO à partir des nœuds qui veulent se connecter avec lui.

La modélisation finale (détaillée un nœud) :

La Figure suivante illustre la modélisation détaillée dans le cas "non connecté "(nœud) :

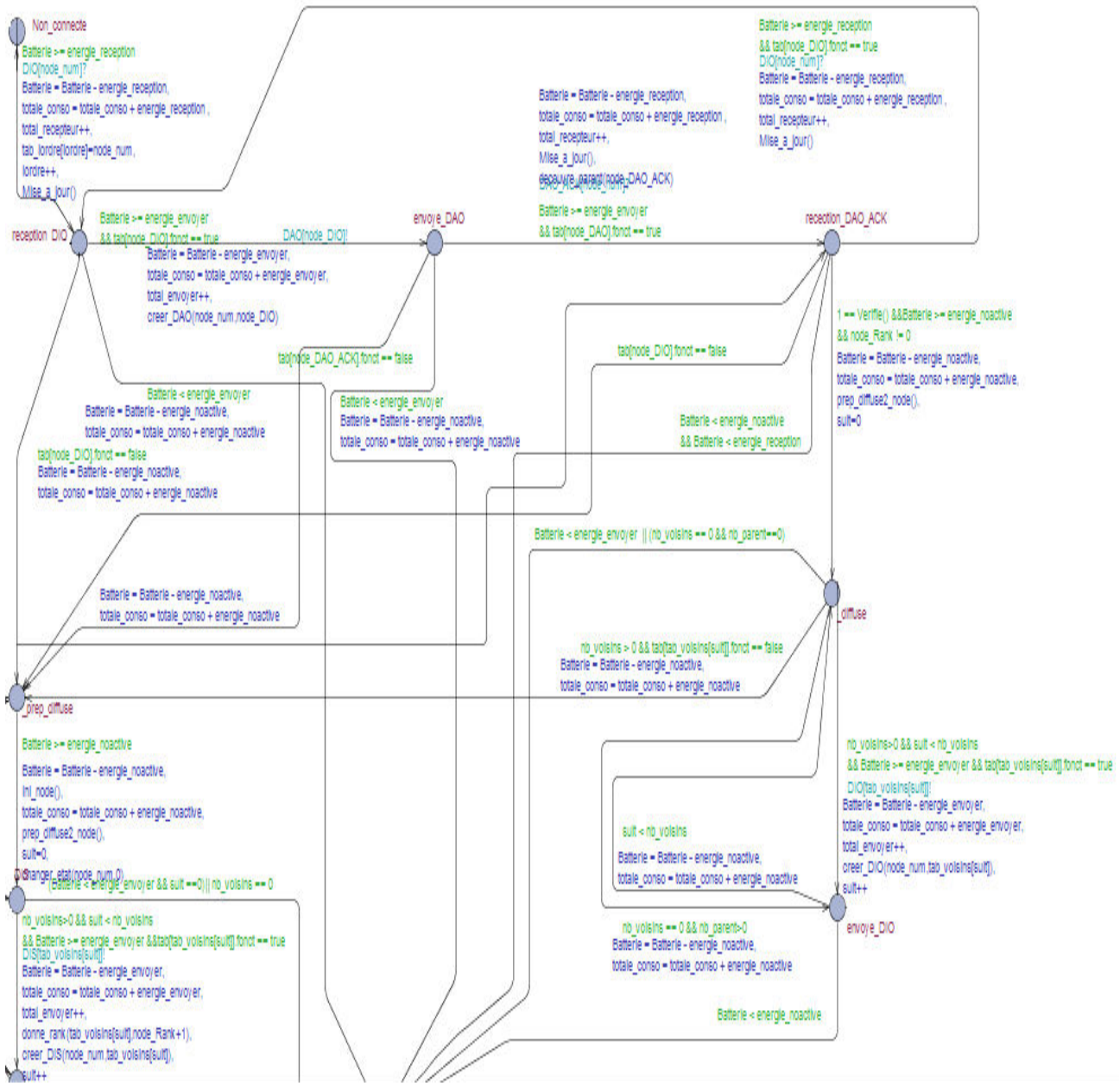

Figure 45:Modélisation détaillée dans le cas non connecté (nœud) .

La Figure suivante illustre la modélisation détaillée dans le cas routage (nœud) :

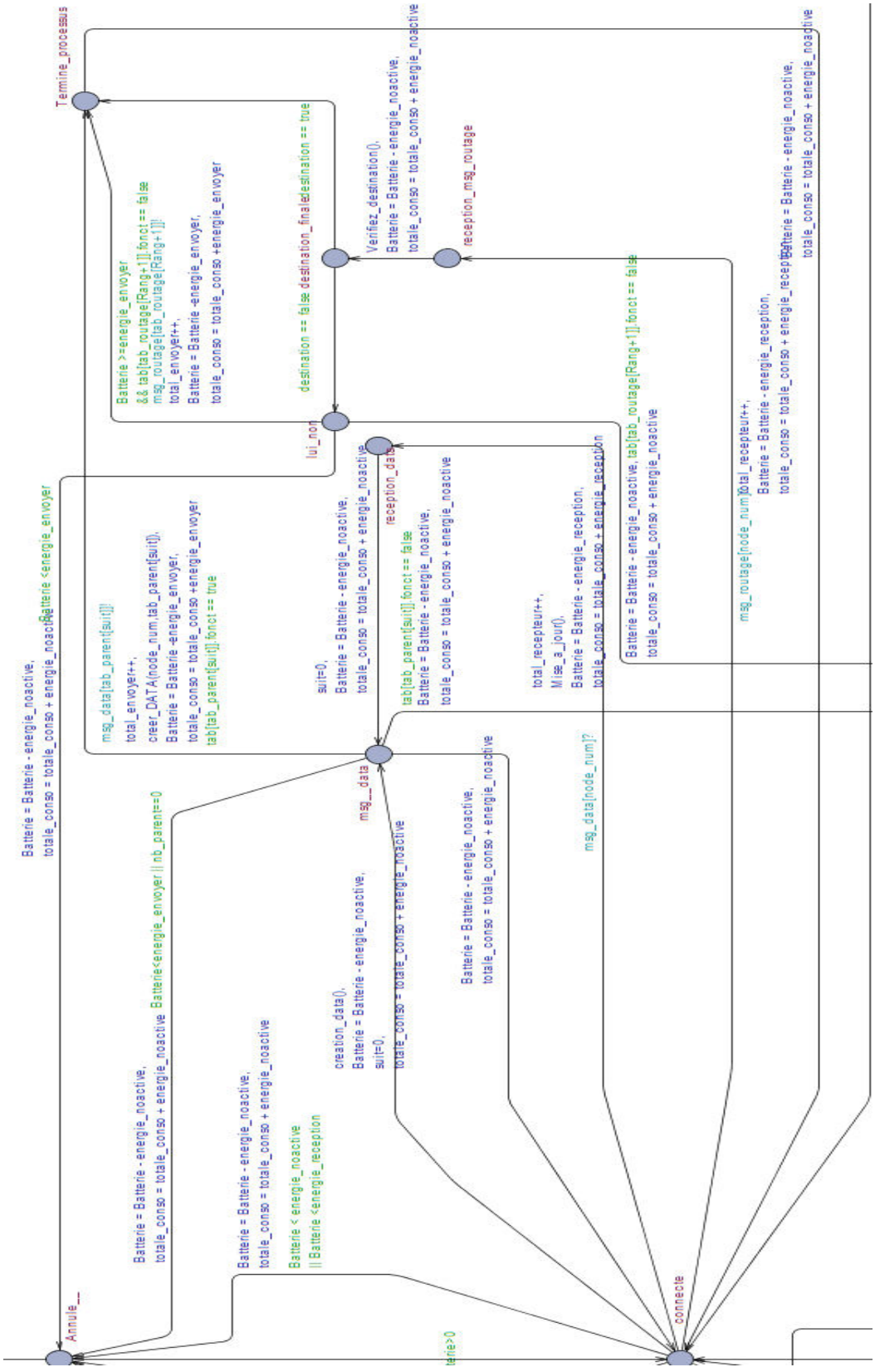

Figure 46:Modélisation détaillée dans le cas routage (nœud).

Le Figure suivante illustre la modélisation détaillée dans le cas connecté recevoir les messages de contrôle (DIS, DIO) (nœud) :

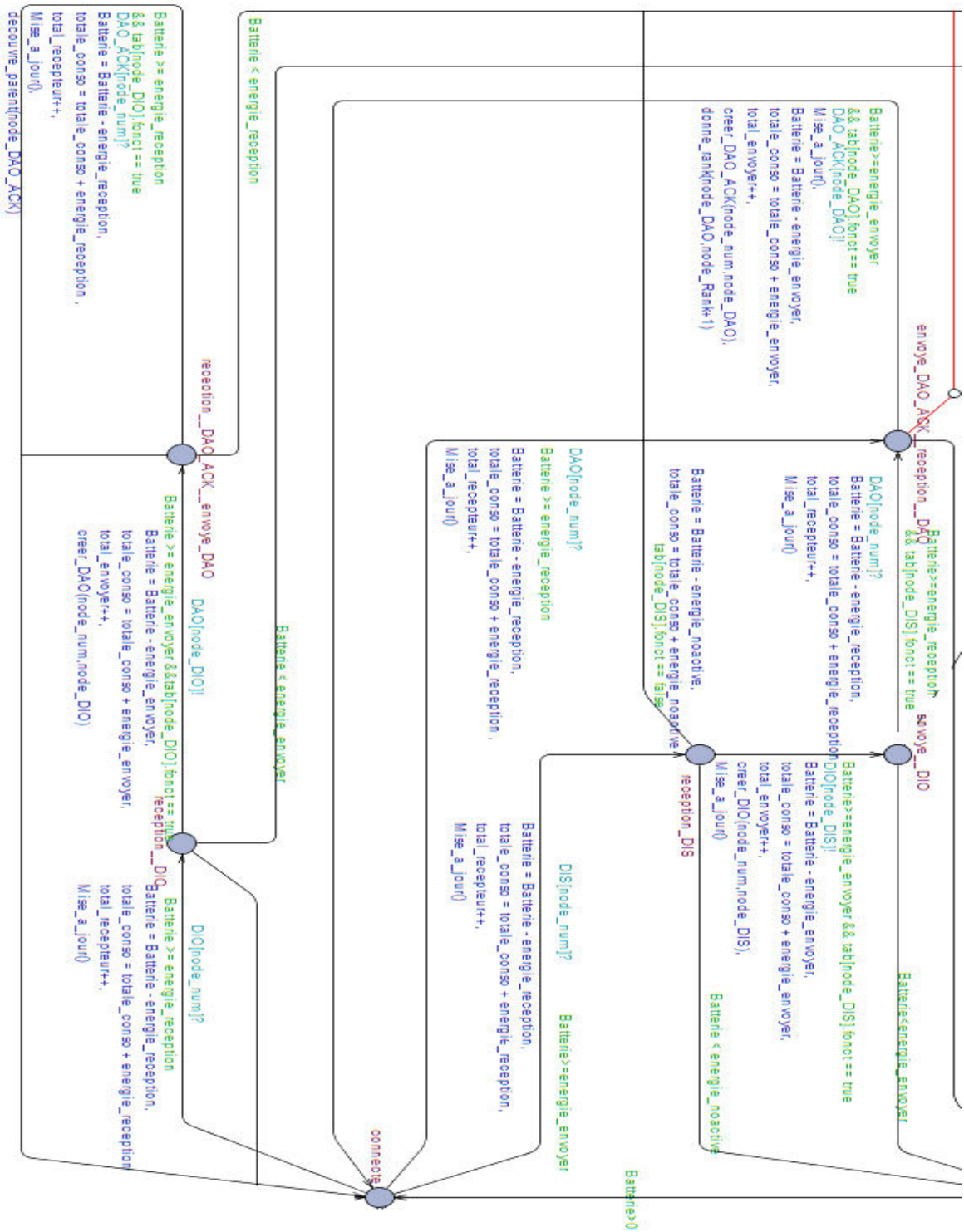

Figure 47:Modélisation détaillée dans le cas connecté recevoir les messages de contrôle (nœud).

IIII.4. Vérification UPPAAL :

UPPAAL permet de vérifier des propriétés d'accessibilité, de sureté, de vivacité et d'absence de blocage. Il utilise un sous ensemble étendu de la logique TCTL (Timed Computation Tree Logic). Soient p et q deux propositions atomiques. Les propriétés seront définies comme suit[25][18] :

IIII.4.1. L'atteignabilite :

Elle peut être exprimée par A <> p (tous les chemins d'exécution possibles atteignent éventuellement un état satisfaisant la propriété p).

IIII.4.2. La sureté :

Elle peut être exprimée par E [] p (il existe au moins un chemin dont tous les états satisfont la propriété p) ou par A [] p (tout état accessible satisfait la propriété p).

IIII.4.3. La vivacité :

Elle peut être exprimée par E <> p (Il existe au moins un chemin D'exécution qui mène éventuellement a un état satisfaisant la propriété p) ou par q-> p (q toujours mène vers p : tout chemin qui débute par un état qui satisfait q atteint éventuellement un état qui satisfait forcement p).

IIII.4.4. L'absence de blocage :

Elle est vérifiée en utilisant la proposition "deadlock". Appliqua sur un état, deadlock retourne "vrai" si et seulement si aucune évolution du système n'est possible à partir de cet état. E <> deadlock est utilisé pour vérifier s'il existe un deadlock et A [] not deadlock est utilisé pour vérifie qu'il n'y a aucun deadlock dans le système.

IIII.5. Vérification du protocole RPL :

Pour ce faire, nous avons détermine un ensemble des cas permettant d'analyser le fonctionnement du protocole RPL. Les propriétés ont été exprimées en logique temporelle à l'UPPAAL. Elles sont résumées dans le tableau suivant :

	Propriétés	Syntaxe	Résultats
P0	L'absence des deadlocks	*A [] not deadlock*	La propriété n'est pas satisfaite.
P1	L'absence des deadlocks avant crée un graphe DAG	*E<>(not deadlock imply Root.Root_connecte ==false imply N_4.node_connecte == false)*	La propriété est satisfait .
P2	Le Root n'est pas connectée au réseau. Et le Nœud 4 est connecté	`A<> Root.Root_connecte == false imply N_4.node_connecte == true`	La propriété n'est pas satisfaite
P3	Le Root n'est pas le parent du nœud 4	*E<> (N_4.tab_parent[0] !=0 || N_4.tab_parent[1] !=0|| N_4.tab_parent[2] !=0|| N_4.tab_parent[3] !=0)*	La propriété est satisfaite.
P4	Tous les nœud sont connectés	*E<> Root.Root_connecte==true implyN_1.node_connecte == true imply N_2.node_connecte==true imply N_3.node_connecte==true imply N_4.node_connecte == true*	La propriété est satisfaite.

Tableau 3:Propriétés vérifiant le fonctionnement de Protocol RPL.

- P0 : Il est clair qu'il existe un deadlock, qui représente le cas où les nœuds tombent en panne, de telle sorte qu'il sera impossible d'acheminer de paquet dans le réseau.
- P1 : Il est important de vérifier qu'il est impossible d'avoir des problèmes de connexion avant la création de la table de routage.
- P2 : Effectivement, c'est un cas impossible.
- P3 : Pour vérifier la connectivité enter le root et le nœud 4.
- P4 : Pour vérifier si le réseau arrive à ses limites et quand ça sera le cas ?.

IIII.6. Résultats de l'analyse de performances de RPL :

Il existe plusieurs métriques pour analyser les performances d'un RCSF, par ni ces métriques bous avons utilisés :

1) consommation totale d'énergie.
2) Nombre de messages envoyés par tous les nœuds (transmission TX).
3) Nombre de messages reçus par tous les nœuds (réception RX).

Qui sont les métriques déjà utilisée pour analyser le protocole RPL par Cooja. Nous avoir utilisé les mêmes métriques avec Uppaal pour obtenir les résultats suivants :

▪ **Energie Vs nombre de nœuds:**

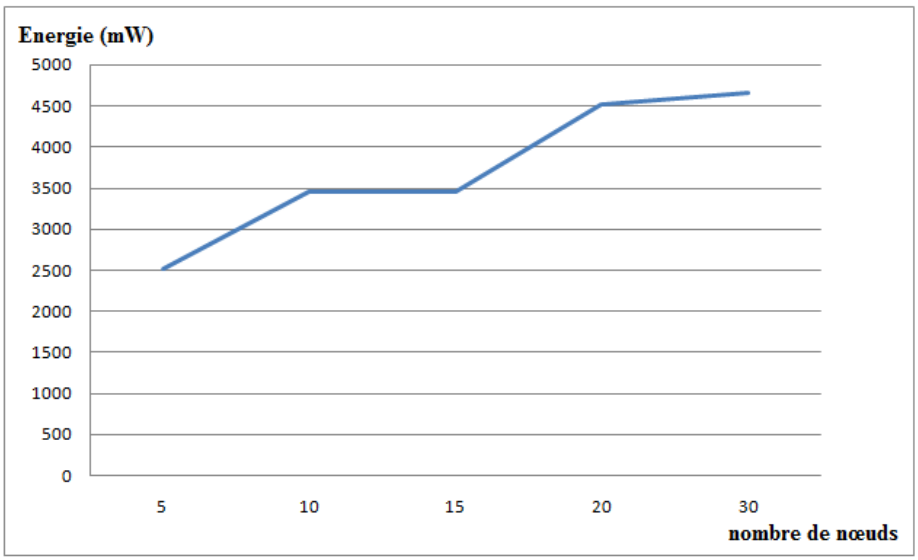

Figure 48:Consommation énergétique totale Vs Nombre de nœuds(UPPAAL).

Cette figure représente la consommation énergétique totale en fonction du nombre de nœuds. Nous remarquons que plus le nombre de nœuds est grand, la consommation d'énergie augment

▪TX Vs nombre de nœuds :

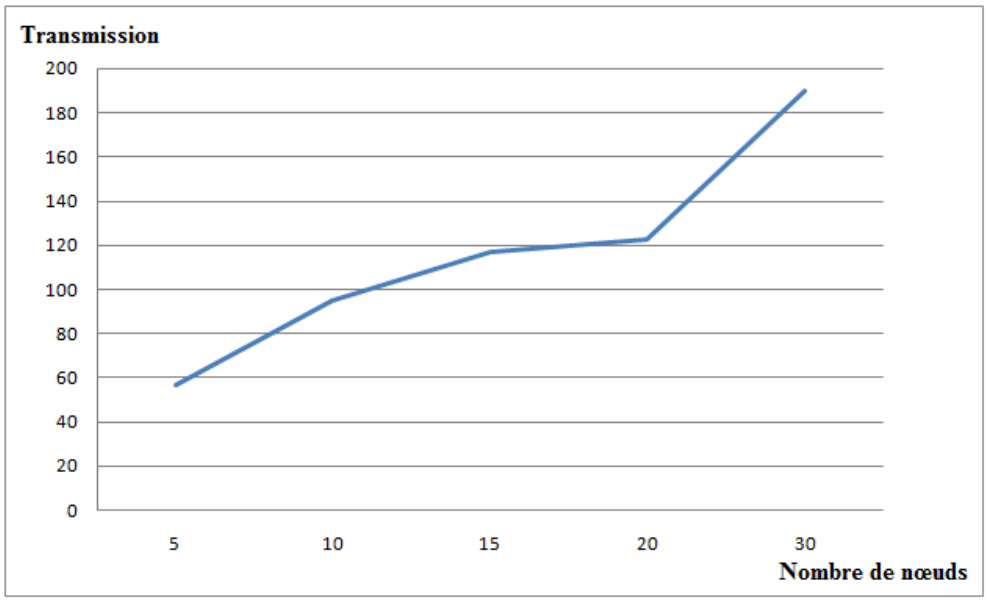

Figure 49:Transmission Vs Nombre de nœuds.

Cette figure représente le nombre de transmission (TX) en fonction du nombre de nœuds. Nous remarquons que. La courbe est Croissante. Il est claire qu'une augmentation du nombre de nœuds entraîne une augmentation du nombre de messages de contrôle, c.à.d. une augmentation du nombre de message envoyés .

▪RX Vs nombre de nœuds :

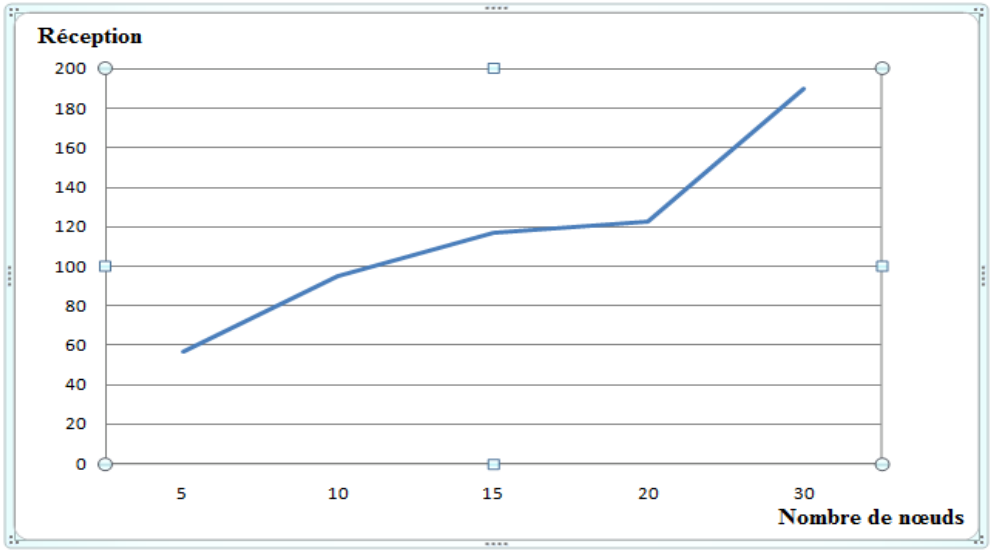

Figure 50:Réception Vs Nombre de nœuds.

Cette figure représente le ratio de réception (RX) en fonction du nombre de nœuds, Mêmes notes en cas de transmission(TX).L'augmentation de nombre de messages reçus peut être expliquée par l'augmentation du nombre de nœuds.

- Notez que dans le cas de l'outil UPPAAL, nous allons le même courbe pour la transmission TX et celle de réception RX. Cela est expliqué par le principe de synchronisation dans UPPAAL. Où chaque transmission (a !) nécessite une réception (a ?).

IIII.7. Comparaison des résultats :

Nous comparerons les résultats que nous obtenons d'UPPAAL avec les résultats de COOJA (Ce que nous avons obtenons dans le troisième chapitre) pour valider notre modélisation.

▪ **Comparaison de la consommation d'énergie :**

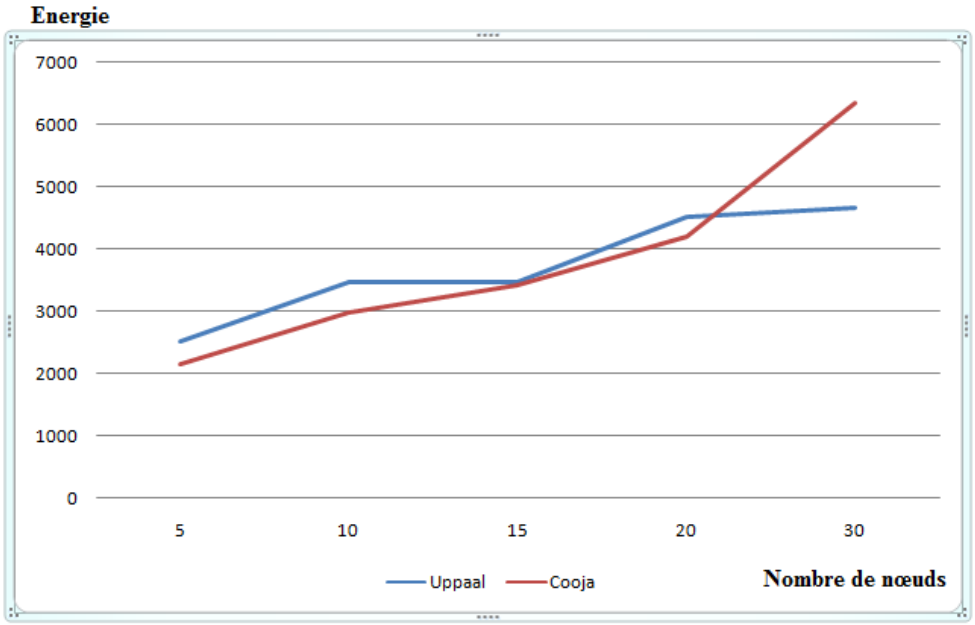

Figure 51:Comparaison de la consommation d'énergie Vs Nombre de nœuds.

La Figure 51 : représenta une comparaison des résultats obtenus sur les deux outils, en tenant compte la consommation total d'énergie Vs le nombre dès noeuds.IL est claire qu'il y a une similitude entre les deux courbes quand le nombre de nœuds varie entre 5 et 20.Nous remarquons que la consommation augmente considérablement que dans Cooja si le nombre de nœuds attient 30, mais ce n'est pas le vas pour l'outil UPPAAL .L'explication de ce phénomène est relié au comportement des nœuds en plus de leur nombre. Le pourcentage des nœuds actifs dans le réseau diminue par rapport le nombre total des nœuds. Les nœuds

non actifs passe en mode veille (de qui simule Cooja) après une attente d'un délai. Pendant ce délai, les nœuds en ce mode est considéré comme déconnecté ce qui entraine une modification dans la table de routage, ce qui implique plus de messages de contrôle(DIS).Une fois le délai de veille attenu sa fin, le capteur redevient actif, ce qui nécessite sa connexion au réseau ce qui implique plus des messages de contrôle aussi cette augmentation de nombre de messages de contrôle engendre une consommation de l'énergie d'où l'hausse de consommation à 30 nœuds. Nous n'avons pas pris en considération ce comportement (mode veille) lors de notre modélisation sur UPPAAL parce que nous avouas jugé que ce phénomène n'est pas lie aux règles du protocole RPL mais plutôt au matériel.

▪ **Comparaison du nombre de transmission :**

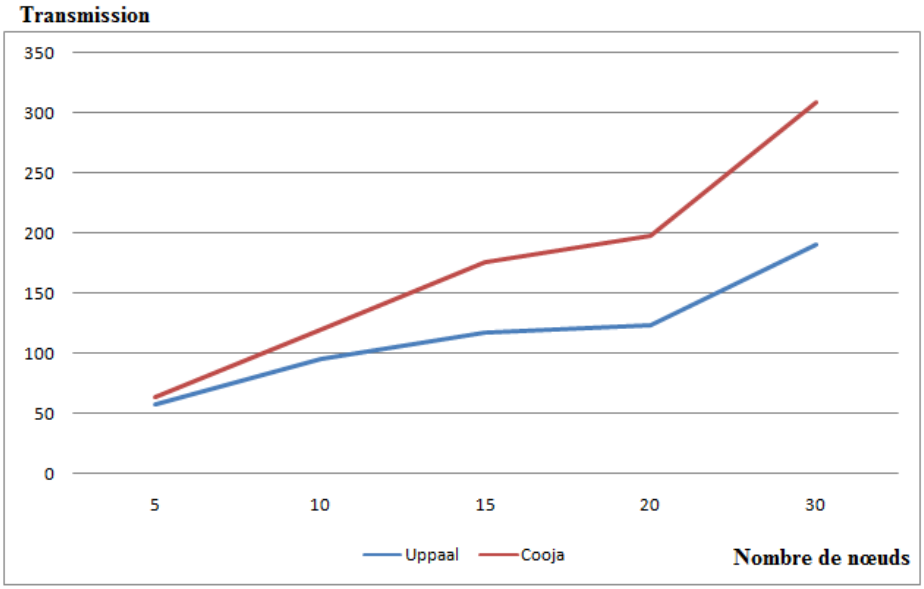

Figure 52:Comparaison du nombre de transmission Vs Nombre de nœuds.

Cette figure représente une Comparaison entre le nombre de transmissions en fonction du nombre de nœuds. Il est à noter qu'il y a un écart entre les deux traces qui est justifié par le phénomène de veille/réveil qui augmente le nombre de messages que ce soit en émission (TX) ou en réception (RX).

▪ **Comparaison du nombre de Réception (RX) :**

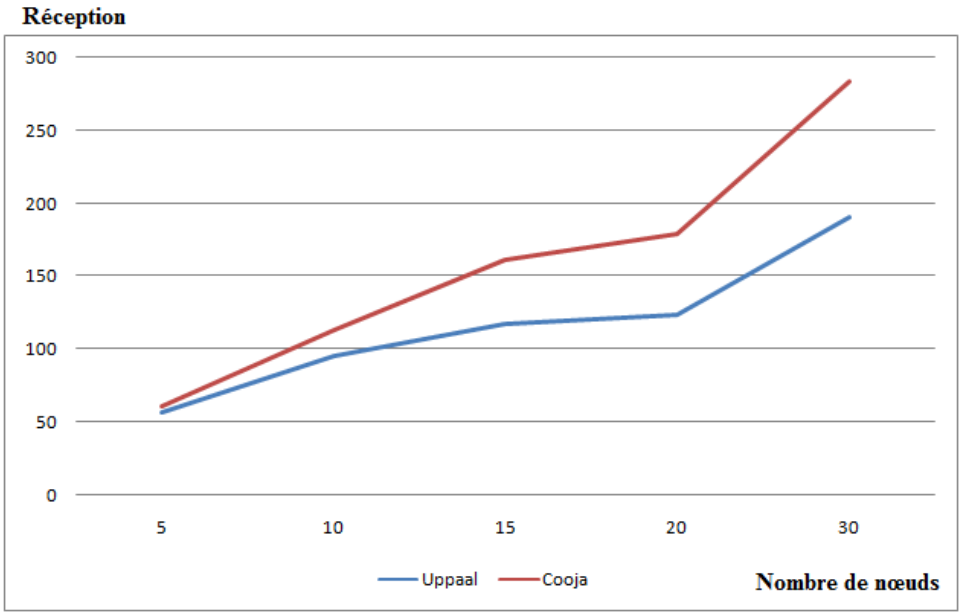

Figure 53:Comparaison du nombre de Réception Vs Nombre de nœuds.

Cette figure représente une comparaison entre les résulta obtenus par les deux outils sur le nombre de messages reçus par rapport au nombre de nouds. Nous soulignons la ressem blance entre cette figure et la figure précédent, car le nombre de messages émis égal au nombre de messages reçus et c'est pour ça les même explications sont valables pour cette figure.

IIII.8. Discussion:

Après avoir modélise le protocole RPL sous UPPAAL, et âpres avoir comparé les résultats obtenus avec ceux obtenus par Cooja, nous pouvons conclure ce qui suit :

- La modélisation faite sur UPPAAL est plus ou moins valide, vu que les traces établis sont presque similaires pour un nombre minium de objets .nous devons améliorer cette modélisation pour tenir en compte le mode veille , ce qui va nous permettre d'avoir des résultats proches à ceux de Cooja même pour un nombre important d'objets.

❖ Les avantages de l'utilisation d'UPPAAL par rapport à Cooja réside dans :

- Le modèle des automates temporises est plus au moins facile à assimiler par les professionnels qui travaillent sur les IoT.

- Le langage utilisé par UPPAAL est un langage similaire au langage C , aves de restriction , ce qui facilite la modélisation et la manipulation du code UPPAAL par rapport à Cooja , qui nécessite la maitrise de plusieurs langages (C,JAVA,XML...).
- Le nombre de fichiers à manipuler dans UPPAAL est minime par rapport à Cooja.
- L'utilisation de la logique temporelle sous UPPAAL nous permettre d'écrie de propriétés et des les vérifier par le model checker facilement, ce qui n'est pas le cas pour Cooja, sous lequel on doit écrire des codes pour vérifier telles propriétés.

❖ L'inconvenant de la modélisation sous UPPAAL est lié à la technique de model cheching, notamment l'explosion combinatoire qui engendre une consommation énorme de la mémoire et un temps de calcul plus élevé. Cet inconvenant peut être remède par l'utilisation de HPC(High Performance Computer).nous pouvons améliorer notre travail en tenant en compte les spécificités matérielles (mode veille) et par la transformation de notre modèle (automates temporises) vers un modèle (automate) statistique sous UPPAAL-SMC pour tirer des statistique (et des probabilités) sur le fonctionnement du RPL tout en bénéficiant des avantages offerts par UPPAAL.

IIII.9. Conclusion :

Dans ce chapitre, nous avons présenté notre modalisation sous UPPAAL et nous avons représente les résultats obtenus par des courbes. nous avons commenté ces courbes et donné des explication au tour de traces .nous avons validé notre modélisation par une comparaison entre nos résultats et ceux de Cooja a la fin de cd chapitre nous avons discuté le travail fait en détaillons les avantages et les inconvenant.

Conclusion générale :

Notre travail rentre dans le coudre de la vérification formelle des protocole de communication. nous avons choisi de travailler sur le protocole RPL qui est répondu dans l'IoT. L'IoTde nos jour ne cesse à être omni présente dans notre vie quotidienne et qui nécessite dé garante une assurance du bon fonctionnement surtout avec l'implication des "Objets". nous a vous détaillé les règles du RPL avant de le simuler par Cooja, qui est un simulateur dentine à l'IoT . L'utilisation d'UPPAAL est motivé par la technique de model checking qui est une technique automatique et exhaustive pour vérifier tels systèmes, ainsi que les outils nécessaires pour modéliser et simuler le protocole en question. La validation de notre travail est faite sous fourme d'une comparaison entre les résultats obtenus par les deux outils ce qui nous a permet de mieux présenter les avantages de notre choix (l'utilisation d'UPPAAL) et de tirer les inconvenants et les lacunes pour améliorer notre travail.

Bibliographie:

- [1] Ali Hazret "A Performance Evaluation of RPL in Contiki,"2012
- [2] : BEHRMANN, G., DAVID, A. et LARSEN, K. G. (2004). A tutorial on uppaal. Formal Methods for the design of real-time systems, Springer. 200{236.
- [3] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, And R. Han. Mantis OS: an embedded multithreaded operating system for wireless Micro sensor platforms. Mob. Netw. Appl., 10(4) : 563-579, 2005.
- [4]: BENGTSSON, J., LARSEN, K., LARSSON, F., PETTERSSON, P. et YI, W. (1996). UPPAAL| a tool suite for automatic verification of real-time systems. Springer.s
- [5] F.Akyildiz, WeilianSu, Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," Aug 2002.
- [6] A. Hakin, A. Gokhale, P. Berthou, D. C. Schmidt, T. Gayraud, Software-Defined Networking: Challenges and research opportunities for Future Internet, Computer Networks 75 (part A) (2014) 453–471
- [7] G. Aceto, A. Botta, W. Donato, A. Pescapè, Cloud monitoring: A survey, Computer Networks 57 (9) (2013) 2093–2115.
- [8] Z. Shelby (Sensinode), K. Hartke, C. Bormann (Universitaet Bremen TZI), Constrained Application Protocol (CoAP),Juin 2014
- [9] Serigne Makhtar Dioum, "Evaluation de l'impact du, mécanisme du « dutycycic » surla qualité de service," Universitet' de Lorraine, France.
- [10]Anuj Sehga, Remi Badonnel, lsabelle Chrisment Anthéa Mayzaud, Gestion de risques appliquée aux réseaux RPL.
- [11] Saad Moustapha, Al-halqi Bassem. Thème: *GESTION DES CLES DANS LES RESEAUX DE CAPTEURS CORPORELS SANS FILS (WBAN).* Année : 2016-2017
- [12] Thème GESTION DE CLES DANS LES RESEAUX DE CAPTEURS CORPORELS SANS FIL (WBAN) Réalisé par : - SEBBAH ABDERREZAK,- CHERRAK SOUFIANE

- [13] D. E Vans, The Internet of things: how the next evolution of the internet is changing every thing, Cisco Internet Business Solutions Group (IBSG), 2011.
- [14] Dunkels, A., B. Grönvall, et T. Voigt. Contiki: a Lightweight and Flexible Operating System for Tiny Networked Sensors. In Proceedings of the FirstIEEEWorkshop on Embedded Networked Sensors, pages 455-462, Tampa, Florida, USA, 2004.
- [15] Elnaz Rezaei, (Master's Seminar). ENERGY EFFICIENT RPL ROUTING PROTOCOL IN SMART BUILDINGS.
- [16] N. Tsiftes, J. Eriksson, and A. Dunkels, "Low-power wireless IPv6 routing with ContikiRPL," in Int Conf on Information Processing in Sensor Networks. ACM/IEEE, 2010.
- [17] Jose A. Gutierrez, Edgar H. Callaway, Raymond L. Barrett, "Low-Rate Wireless Per-sonal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4" in, IEEE Press, 2003.
- [18] ALUR, R., COURCOUBETIS, C. et DILL, D. (1993). Model-checking in dense real-time. Information and computation.
- [19] E. FELEMBAN, C.-G. LEE, and EKICI, FELEMBAN, E. MMSPEED) Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks. 2006.
- [20] Ali MAKKE Thèse Présentée pour obtenir le grade de docteur de l'Université Paris Descartes Spécialité : Informatique et Réseaux Détection dattaques dans un système WBAN desurveillancemédicale à distance
- [21] Bardh Prenkaj. RPL: Routing for IoT. Dept. Of Computer Science, université de ROMA
- [22] J. Granjal, E. Monteiro, J. Sá Silva, Security in the integration of low-power Wireless Sensor Networks with the Internet: A survey, Ad Hoc Networks 24 (2015) 264–287.
- [23] L. Atzori, A. Lera, G. Morabioto, The Internet of Things: a survey, Computer Networks 54 (15) (2010) 2787–2805.
- [24] D. Miorandi, S. Sicari, F. De-Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad Hoc Networks 10 (7) (2012) 1497-1516.

- [25] MAJDA MOUSSA. (2014). Mémoire : Vérification et configuration automatiques de pare-feu par model chocking et synthèse de contrôleur. Université de montreal.Ecole Polytechnique de Montréal
- [26] J. Decuir, "Bluetooth: Low Energy", Presentation slides, 2010, http://chapters.comsoc. org/vancouver/BTLER3.pdf
- [27] Contiki, “Contiki: The Open Source Operating System for the Internet of Things,” 2015. [Online]. Available: http://www.contiki-os.org/.(2018.Janvier).
- [28] Tinyos. Http: /www.tinyos.net/, 2010.] (2018.Janvier)
- [29] Get Started with Contiki. [Online]. http://www.contiki-os.org/start.html.] (2018.Janvier)
- [30] GitHub. [Online].https://github.com/Riot-os/riot/pull/2754. (2018.Janvier)
- [31] Smart grids.Les object connectés http://www.smartgrids-cre.fr/index.php?p=objets-connectes-technologies. (2018.Janvier)
- [32]"ZWaveProtocolOverview,"v.4,May2007, https://www.openhabfoundation.org/documents/201710_Chris_Jackson_A_Deep_Dive_into_Z-Wave.pdf