UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

ÉCOLE DOCTORALE SCIENCES DU NUMÉRIQUE ET DE L'INGÉNIEUR n°620

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Discipline : INFORMATIQUE

Spécialité : informatique

Présentée et soutenue publiquement Par

HÉRARD JOFFREY

Le 25 février 2022

Structuration autonome des réseaux IoT de type LoRa / LoRaWAN

Thèse dirigée par **OLIVIER FLAUZAC ET FLORENT NOLOT**

JURY

Mme Isabelle GUÉRIN-LASSOUS,	Professeur,	Université Claude Bernard Lyon 1,	**Présidente**
M. Olivier FLAUZAC,	Professeur,	Université de Reims Champagne-Ardenne,	**Directeur de thèse**
M. Florent NOLOT,	Maître de Conférences HDR,	Université de Reims Champagne-Ardenne,	**Co-Directeur de thèse**
M. Fabrice VALOIS,	Professeur,	INSA Lyon,	**Rapporteur**
M. Antoine GALLAIS,	Professeur,	INSA Hauts-de-France,	**Rapporteur**
Mme Anne-Elisabeth BAERT,	Maitre de conférences,	Université de Montpellier ,	**Examinateur**

"Tout le monde savait que c'était impossible.
Il est venu un imbécile qui ne le savait pas et qui l'a fait."
- Marcel Pagnol

"À la mémoire de mon grand père, Alexandre Doré"

Remerciements

Quand j'ai entrepris de commencer cette thèse, je n'aurais jamais imaginé ne serait-ce que pouvoir la commencer tant ce passage fut compliqué. Quand la thèse fut enfin démarrée, j'étais enthousiaste, j'étais en train d'accomplir ce que je voulais faire depuis mon entrée au lycée : de l'informatique, de la recherche et des horaires d'encadrements. Le triptyque parfait pour moi. Cependant trois ans de travail ne se résume pas que à une seule personne. C'est pourquoi il me semble bien naturel de remercier aussi bien la chance d'avoir pu entamer cette thèse, que le hasard de m'avoir mis sur le bon chemin avec les bonnes personnes. Mais surtout les personnes qui m'ont accompagné, et celles qui ont contribué à la réalisation de cette thèse

Je décerne mes remerciements à mes directeurs de thèse *Olivier Flauzac* et *Florent Nolot* respectivement Professeur et Maître de conférences à l'Université de Reims Champagne-Ardenne. Merci pour la confiance qu'ils m'ont donnée, merci de m'avoir offert cette opportunité de faire une thèse. Merci à eux de m'avoir offert l'opportunité de travailler dans un projet industriel ambitieux. Merci pour leur gentillesse, leur bienveillance, leur bonne humeur. Merci de s'être rendu très disponibles pendant ces trois années, d'avoir été à l'écoute, d'avoir été des guides dans la forêt sombre qu'est celle de la recherche. Merci de m'avoir offert ce que je ne pouvais pas m'offrir seul. J'espère avoir été à la hauteur de leurs attentes.

Quand je parlais de projet industriel, je parlais de celui qui encadre tous les travaux présentés dans ce manuscrit. Ceux-ci ont été réalisés au sein d'une collaboration étroite avec *Objenious* la marque de Bouygues Telecom. Merci à *Objenious* , dont le soutien financier a permit à ce travail d'être possible. Merci à la personne qui a encadré tout cela pendant ces 3 années, *Philippe Cola*. Merci à lui d'avoir eu confiance en Olivier, Florent et moi pour mener à bien ces travaux de recherches.

Je remercie *Cyril Rabat*, Maître de Conférences à l'Université de Reims Champagne-Ardenne, dont les conseils m'ont particulièrement aidé à plusieurs étapes de mes travaux. Je remercie *Christophe Jaillet*, Maître de Conférences à l'Université de Reims Champagne-Ardenne, pour ces conseils sur la visualisation des données de sorties de toutes les différentes simulations réalisées.

Mes remerciements vont également à Ida Lenclume, secrétaire du CReSTIC, pour sa disponibilité et toute l'aide apportée durant ces années de thèse.

À *Océane Cornevin* de m'avoir soutenu pendant toutes ces années avant même que la thèse ne soit commencée jusqu'à aujourd'hui. Merci d'avoir été là pour moi à tout les niveaux pendant tout ce temps.

À ma mère *Béatrice Hérard* et mon père *Joël Hérard* que je ne saurais jamais assez remercier pour votre persévérance, et soutien quotidien. Je vous exprime ma plus grande gratitude pour tout ce que vous avez fait pour moi jusqu'à aujourd'hui. Merci pour vos conseils et votre amour

réciproque.

Il y a malgré tout, une personne qui ne pourra plus jamais avoir l'occasion de savoir ce que j'ai fait pendant 3 ans. J'ai le profond regret de citer *Alexandre Doré* feu mon grand-père que je remercie de m'avoir inspiré à être le plus rigoureux, bienveillant, curieux. Un modèle de vie pour moi. Merci d'avoir assuré une partie de mon éducation. Repose en paix.

Merci à mes amis qui m'ont apporté leur soutien, *Emmanuel Pluot*, *Loïc Perrin*, et *Nicolas Bouème* qui m'ont soutenu par leur bonne humeur, leur gentillesse, leurs regards sur l'aspect théorique et nos échanges, leur aide sur la relecture, *Erwann Ancelin* pour son soutient dans les moments plus difficiles.

Mes remerciements vont également à tous ceux qui ont contribué de près ou de loin à l'élaboration de cette thèse, qu'ils trouvent ici l'expression de ma profonde sympathie.

En plus des personnes avec lesquelles j'ai pu coopérer sur ce travail, je remercie celles qui ont assisté à ma soutenance.

Table des figures

Liste des tableaux

Liste des définitions

Liste des publications

[FHN20] Olivier Flauzac, Joffrey Herard et Florent Nolot. “Synchronization Solution to Optimize Power Consumption in Linear Sensor Network”. In : *International Conference on Fog and Mobile Edge Computing (FMEC)*. Paris, France, 2020, p. 295-300. doi : 10.1109/FMEC49853.2020.9144887. url : https://hal.archives-ouvertes.fr/hal-02909570.

[Fla+20a] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. “A Fault Tolerant LoRa/LoRaWAN Relay Protocol Using LoRaWAN Class A Devices”. In : *International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)*. Bari, Italy, 2020, p. 295-302. doi : 10.1007/978-3-030-61746-2_22. url : https://hal.univ-reims.fr/hal-02978424.

[Fla+20b] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. “A low power LoRa-LoRaWan relay function with a single input, single output device”. In : *International Conference on Embedded Wireless Systems and Networks (EWSN)*. Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks. Lyon, France, 2020, p. 283-288. url : https://hal.univ-reims.fr/hal-02852171.

[Fla+20c] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. “LLWURP : LoRa/LoRaWAN Uniform Relay Protocol with a Single Input, Single Output Device”. In : *Future Technologies Conference (FTC) 2020*. T. 2. San Francisco, United States, 2020, p. 862-876. doi : 10.1007/978-3-030-63089-8_56. url : https://hal.univ-reims.fr/hal-02989181.

[Fla+20d] Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. “Comparison between different platform using a Uniform Relay Protocol in Linear Sensor Network”. In : *International Conference on the Internet of Things Companion (IoT)*. Malmö, Sweden : ACM, 2020, p. 1-4. doi : 10.1145/3423423.3423462. url : https://hal.univ-reims.fr/hal-02978428.

[Fla+20e] Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. “Comparison Between Different Types of Sensor Architecture Using a Uniform Relay Protocol”. In : *International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2020 8th International Conference on Wireless Networks and Mobile Communications (WINCOM). Reims, France : IEEE, 2020, p. 1-6. doi : 10.1109/WINCOM50532.2020.9272428. url : https://hal.univ-reims.fr/hal-03042336.

Table des matières

Introduction

Depuis les années 2000 la communication en informatique a évolué autant du côté des particuliers que du côté des domaines dans lesquels ils s'ancrent. Cette connectivité devient omniprésente avec la création de nombreuses plateformes d'échanges, de messagerie, d'achats et de partage d'informations.

Parallèlement, la communauté scientifique étudie les réseaux de capteurs sans fil pour diverses raisons et applications. L'intention est donc d'utiliser les données récoltées pour les interpréter ou s'en servir. Pour ce faire ils doivent communiquer entre eux avec un protocole sans fil. Les protocoles utilisés par les particuliers ont été en premier lieu le Bluetooth et le Wi-Fi avec l'avènement des smartphones. Par la suite vient l'émergence des capteurs sportifs, des casques audio, des capteurs de surveillance de la maison ou des plantes, ou encore des interactions via des QR codes, NFC, RFID. De plus en plus de protocoles avec de plus en plus de manière d'interagir avec différents objets passifs ou actifs. La connectivité étant différente en fonction des objets mais basée sur une infrastructure de récolte de données, cela débouche sur une sorte d'Internet. Tout ceci assemblé fait donc naître le concept d'Internet des Objets. Un réseau mondial où les objets, les humains et les applications informatiques peuvent communiquer ensemble. Ainsi naquirent les réseaux LPWAN pour "Low Power Wide Area Network" qui visent à être l'équivalent des réseaux mobiles cellulaires mais adaptés aux objets . La problématique de la couverture existe aussi comme pour les réseaux mobile, où la couverture doit être le plus grande possible afin d'assurer une qualité de service sur la présence de la connectivité. Pour cela, les réseaux LPWAN sont définis afin d'envoyer des messages courts, sur une longue distance et avec un bilan énergétique très faible.

Les technologies radios doivent pour exister faire des compromis très importants au niveau de l'économie d'énergie, du débit, de la portée, de la sécurité ou encore de la fiabilité. Chaque aspect est une possible problématique pour tout problème autour des réseaux de capteurs sans fil.

Voici le contexte dans lequel les travaux issus de cette thèse s'intègrent. L'approche de l'Internet des Objets ne peut se faire sans cette vision globale des enjeux et de l'historique du développement des technologies. Nous avons fait le choix de garder cette approche globale, tout en implémentant nos contributions sur une technologie en particulier, les réseaux LoRaWAN et la technologie LoRa. Nous abordons le problème de la connectivité, la sécurité, la fiabilité, la portée et l'économie d'énergie. Nous proposons une solution de relais avec une construction incrémentale à travers chaque chapitre une notion va être ajoutée pour améliorer la solution de base.

C'est ainsi que la contribution de ce manuscrit à travers 7 chapitres sera organisée :

Chapitre 1 : Nous présentons et détaillons les différents concepts autour des systèmes distribués qui sont le cadre général de nos travaux de recherche. Puis nous abordons leur architecture et topologie avec quelques algorithmes associés.

Chapitre 2 : Nous décrivons d'abord le modèle de nos travaux de recherche, l'Internet des objets(IoT ou Ido). Ceci à travers la présentation de ce paradigme, les architectures des applications, les solutions de communications employées.

Chapitre 3 : Nous abordons les divers protocoles utilisés dans le cadre de nos recherches et surtout le cas spécifique du LoRa/LoRaWAN que est détaillé dans ce chapitre. Après la présentation du LoRaWAN nous soulèverons notre problématique .

Chapitre 4 : Nous détaillons les spécifications de la première version de notre algorithme de relais accompagné d'une campagne de simulation pour calculer l'impact sur les messages et la consommation énergétique d'une version dite relais. Puis avec un deuxième aspect dit d'usurpation. Nous comparons également notre solution de relais avec différents protocoles comme le LoRa, Wi-Fi, Bluetooth afin d'estimer l'impact énergétique d'un protocole radio.

Chapitre 5 : Nous commençons l'évolution de la solution présentée au chapitre précédent vers une passerelle LoRa/ LoRaWAN uniforme et sécurisée. Pour cela nous mettons en place un système d'authentification et de synchronisation. Nous étudions et comparons les performances ainsi que l'impact de cette version de notre solution. Nous soulevons les différents problèmes d'implémentation que nous avons eue afin de réaliser la mise en place de notre solution.

Chapitre 6 : Nous détaillons dans ce chapitre la mise en place et la considération des aspects de dynamicité et de sûreté de fonctionnement de notre solution. Nous soulevons les différentes évolutions que peuvent subir les différents acteurs. Un nouvel environnement de simulation qui permet une étude plus approfondie est présenté. Nous présentons également une expérimentation dans le monde réel.

Chapitre 7 : La dernière version de cette solution est détaillé dans ce chapitre. Le principe de maillage est ajouté. Nous détaillons l'algorithme, les simulations, l'implémentation, le passage au monde réelle et le passage à l'échelle.

Chapitre 1
Système Distribué

Résumé : *Le cadre général de nos travaux de recherche est présenté : les systèmes distribués. Nous présentons la définition, son modèle d'exécution, ses modèles de communication. Nous abordons les différentes architectures et topologies, pour finir ce chapitre sur les réseaux ad hoc et plus particulièrement les réseaux de capteurs sans fils*

Sommaire

1.1 Généralités sur les systèmes distribués

Dans cette partie, nous présentons d'abord les systèmes distribués avec plusieurs définitions de ceux-ci, les différents modèles d'exécution, de synchronisation et modèles de communication. Après cette partie, différentes topologies autour des systèmes distribués, architectures sont présentés.

1.1.1 Définitions

Un système distribué est défini comme étant un ensemble d'acteur ayant une capacité de traitement et de stockage, qui sont autonomes, inter-connectés et peuvent communiquer entre eux comme décrit par Raynal dans [Ray85].

Un système distribué a pour but de fournir un service qui ne pourrait pas être réalisé par un seul acteur en termes de fonctionnalité, disponibilité, puissance de traitement ou de stockage, temps de réponse, fiabilité, etc.

Les acteurs peuvent être des processus, des processeurs ou des ordinateurs, etc. Ils sont généralement appelés nœuds du système distribué. Pour être inter-connecté, chaque nœud doit disposer de l'usage d'un médium pour assurer la communication. Ainsi, les nœuds s'échangent des informations selon un modèle de communication.

Un système distribué peut être qualifié d'homogène ou d'hétérogène. Dans le premier cas tous les nœuds sont identiques. Mais dans le deuxième les nœuds du système peuvent être de divers types.

Les nœuds coopèrent à l'aide d'un modèle de communication pour accomplir la tâche globale du système dont le déroulement est induit par un algorithme distribué. Un algorithme distribué est constitué de l'ensemble des algorithmes des différents nœuds du système distribué. Au niveau de chaque nœud, l'exécution d'algorithme distribué est déclenchée par un événement de type réception de message ou l'épuisement d'un timer. Cet événement entraine donc une exécution d'une instruction en interne qui peut modifier l'état du nœud. Puis, cette modification de l'état du nœud peut conduire à l'émission d'un message ou au déclenchement d'un timer. L'algorithme distribué s'exécute ainsi pour accomplir la tâche du système.

Les systèmes distribués offrent plusieurs utilisations qui ont favorisé leur développement massif. Ils peuvent permettre le partage de ressources, l'augmentation de la fiabilité des données par le biais de réplications, la haute disponibilité, l'amélioration des performances par le parallélisme, la simplification de la conception d'un système par une structuration modulaire etc. Il existe plusieurs ouvrages dans la littérature traitant des systèmes distribués ainsi que les algorithmes distribués [Lav95][Ray92][Tel01][AW04][Ray13][Lyn96] [Ray85][RH88].

1.1.2 Quelques notions sur les graphes

Pour modéliser un graphe, des terminologies et notations sont employées, celles-ci sont tirées de la théories des graphes [Gib85] [LPS03] [GM90] [BFH12]. Un système distribué est défini comme un graphe connexe :

1. Le graphe est noté G .
2. V décrit l'ensemble des sommets (nœuds, sites, processeurs etc..) du graphe du système.
3. E décrit l'ensemble des connexions existantes entre tous les nœuds du graphe.
4. Un graphe est connexe si et seulement s'il existe un chemin entre tout couple de nœuds du système E.
5. G peut être orienté ou non.

— Le graphe G est dit orienté(Figure 1.1) si et seulement si un ensemble de couples ordonné distincts communément appelé (u,v) de sommets appelés arcs où u est l'origine et v la destination.

— Le graphe G est dit non-orienté (Figure 1.2) si et seulement si un ensemble de couples ordonnées distincts (u,v) de sommets appelés arrêtes où (u ;v) = (v ;u).

6. G peut être anonyme, les nœuds de V n'ont donc pas d'identité. Cependant lorsque tous les nœuds de V sont distingués par un identifiant unique, on parle alors de graphes avec identité.

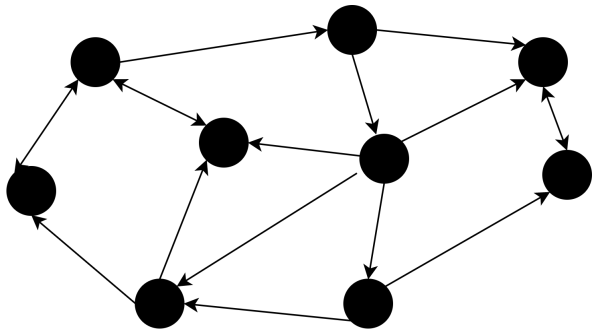


FIGURE 1.1 – Graphe orienté

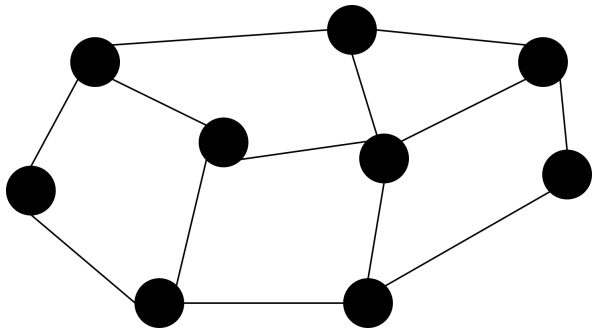


FIGURE 1.2 – Graphe non orienté

1.1.2.1 Modèle adopté

Dans nos travaux de recherches, nous prenons comme modèle des systèmes distribués représentés par des graphes :

— connexes

— non-orientés

— avec identités

Comme il s'agit d'un graphe avec identités, chaque nœud du graphe possède un identifiant unique : son identité est notée id_u . Cette entité peut être un nombre ou une chaine de caractères. La valeur de cette identifiant provient d'un ensemble I muni d'un ordre total qui peut être l'ordre numérique sur N, de l'ordre alphabétique ou lexicographique. Par la suite nous utiliserons les mots : voisinage, degré, distance définis ainsi :

Définition 1.1.1 (Distance)
La distance entre deux nœuds u et v quelconques dans le graphe G est noté $D_{(u,v)}$. Elle représente la longueur du plus court chemin, correspondant au nombre d'arêtes minimal entre u et v.

Définition 1.1.2 (Voisinage)
Le voisinage d'un nœud u, noté V_u, représente l'ensemble des nœuds $v \in V|(u,v) \in E$

Définition 1.1.3 (Degré)
Le degré d'un nœud noté Δ_u représente le nombre de nœuds dans le voisinage de V_u de u. Soit $\Delta_u = |N_u|$.

1.1.3 Modèle d'exécution

Après avoir abordé quelques notions sur les graphes ainsi que la description du modèle que nous avons choisi, nous allons présenter le modèle d'exécution associé au modèle de représentation. Ainsi Tel modélise dans [Tel01] l'exécution d'un système distribué par un *système à transitions* introduit par Arnold dans [Arn92].
Avant de présenter un système de transitions et le modèle d'exécution d'un système distribué, nous allons rappeler les définitions des notions d'*état* et de *configuration*.

Définition 1.1.4 (État)
L'état d'un nœud est défini par ses variables locales ainsi que celles de ses canaux.

Définition 1.1.5 (Configuration)
Une configuration est une instance de tous les composants du système, c'est à dire l'état global du système.

Un systèmes de transition se définit alors ainsi :

Définition 1.1.6 (Système de transitions)
Un système de transition est un triplet S = (C,→ , I) où :

1. *C est l'ensemble de toutes les configurations possibles ;*
2. *→ est une relation binaire dans C ;*
3. *I un sous ensemble de C de toutes les configurations initiales.*

Un système de transitions représente toutes les évolutions possibles d'un système distribué. Une transition est appelée *round* ou *étape*. Un système distribué suivant un système de transitions peut être maximal ou fini.

Définition 1.1.7 (Exécution)
Une exécution d'un algorithme distribué est une séquence E = (i_0,i_1,i_2,i_3,i_4...) issue du système de transitions.

Définition 1.1.8 (Exécution finie)
Une exécution est une séquence E = (i_0,i_1,i_2,i_3,i_4...) avec $i_i \rightarrow i_{i+1}$ pour tout i, elle est dite finie si elle se termine par une configuration terminale.

Définition 1.1.9 (Exécution maximale)
Soit $S = (C,\rightarrow , I)$ un système de transitions. Une exécution maximale de S est une séquence maximale $E = (i_0,i_1,i_2,i_3,i_4...)$ où :

- $i_0 \in I$
- $\forall i \geq 0, i_i \rightarrow i_{i+1}$

Définition 1.1.10 (Configuration terminale)
Une configuration terminale est une configuration i pour laquelle il n'existe aucune autre configuration j telle que $i \rightarrow j$.

L'exécution sur un système distribué, qui suit un système de transitions, d'un algorithme s'exécutant au niveau de chaque nœud est une collection de règles disjointes de la forme suivante :

$$Algorithme \equiv < Regle_1 > | < Regle_2 > |..| < Regle_n >$$

Où une règle est définie ainsi :

$$Regle_i ::< garde > \rightarrow < action >$$

Pour chaque règle, $< garde >$ est un prédicat booléen qui en fonction de l'état local du site/nœud et de ceux de ses voisins, exécute une $< action >$ en interne qui peut modifier son état local.

1.1.4 Synchronisation

Comme le présentent Raynal et Helary dans [RH88], l'exécution d'un système distribué peut suivre deux modes de synchronisation différents : synchrone ou asynchrone.

1.1.4.1 Système synchrone

Dans un système synchrone, les horloges locales de tous les nœuds sont synchronisées et possèdent la même valeur. Tous les nœuds commencent leurs activités au signal d'un initiateur global appelé *synchroniseur* qui synchronise cycle après cycle chaque événement et chaque action de tous les nœuds du système. C'est ce qui est appelé l'ordonnancement de tâches. L'exécution d'un système synchrone est partitionnée en rounds.

Définition 1.1.11 (Round)
Un round correspond à un envoi et à une réception de tous les messages.

1.1.4.2 Système asynchrone

Un système asynchrone est caractérisé par l'absence d'horloge globale. Contrairement à un système synchrone le concept d'instant est étranger à un système asynchrone. Les seuls événements perçus par un nœud sont soit les événements produits par lui même, soit des événements causés par d'autres nœuds. La spécificité asynchrone d'un système possédant n nœuds implique qu'il n'existe aucune horloge globale mais n horloges locales aux nœuds sans relation entre elles. L'évolution de l'état global du système est induite par un événement de type réception d'un message ou par le déclenchement de "timer" autonomes dont l'exécution est indépendante de tout mécanisme externe.

Raynal et Helary dans [RH88] ont présenté plusieurs algorithmes de synchronisation

permettant de simuler et de contrôler l'exécution d'algorithme synchrone dans un contexte asynchrone.

1.1.5 Modèles de communication

Nous avons montré qu'un système distribué pouvait être représenté par des graphes et que son exécution peut être tantôt synchrone tantôt asynchrone, tous deux suivant un système de transitions. Nous passons maintenant à la présentation des différents modèles de communication qui peuvent être utilisés pour les échanges d'informations.

Dans un système distribué, les communications entre les nœuds peuvent se faire suivant 3 modèles :

1. le modèle à états
2. le modèle à registres
3. le modèle à passage de messages

Le modèle à états et à registres sont deux modèles qui se basent sur la mémoire partagée. Chacun de ces trois modèles de communication présente des spécificités qui lui sont propres et font l'objet des points suivants. Comme présenté par les auteurs dans [GM91] [KP93] [DIM91].

1.1.5.1 Modèle à état

Le modèle à états a été introduit par Dijkstra en 1974 dans [Dij74]. Ce modèle a pour particularité que chaque nœud peut lire l'état de ses voisins (Figure 1.3).

Un système distribué est caractérisé à états si et seulement si pour tous les couples de voisins (u,v), u est le seul à disposer du privilège d'écriture sur son état et peut lire les informations sur l'état de v. De même v peut lire l'état de u et dispose exclusivement du privilège d'écriture sur son état. Avec ce système, chaque nœud connait l'état de chacun de ses voisins.

Dans un modèle à états, un ordonnanceur est utilisé pour diriger les différents changements d'états du systèmes. Celui-ci est appelé démon. Un démon est un mécanisme qui durant chaque configuration i_i, choisit à partir de l'ensemble de tous les nœuds déclenchables, tel qu'un sous ensemble Q $\subseteq$ N. Un nœud est déclenchable si il est en capacité d'exécuter une action. Parmi tout les nœuds de l'ensemble Q, le démon choisit une règle qui va s'exécuter puis le nœud est libéré.

Concernant le démon il peut être centralisé voire distribué ou bien totalement distribué.

Définition 1.1.12 (Démon centralisé)
Un démon dit centralisé (Central Daemon) si et seulement si à chaque configuration $|Q| = 1$. Autrement dit, à chaque configuration i_i, parmi l'ensemble de tous les nœuds déclenchables, le démon en choisit un seul qui sera assuré d'exécuter une règle.

Définition 1.1.13 (Démon distribué)
Un démon est dit distribué (Distributed Daemon) si et seulement si $1 < |Q| \leq |N|$. Ce qui veut dire qu'à chaque configuration i_i, un sous ensemble de l'ensemble des nœuds déclenchables est sélectionné par le démon pour exécuter une règle applicable. Dans ce cas, l'exécution s'effectue de manière synchrone. Tous les nœuds sélectionnés exécutent leurs règles en même temps.

Définition 1.1.14 (Démon totalement distribué)
Un démon dit totalement distribué (Fully Distributed Daemon) est semblable à un démon distribué à la différence que l'exécution des règles est asynchrone. Dès que l'autorisation est donnée par le démon, chaque nœud est libre d'exécuter sa règle quand il le souhaite.

Il subsiste un détail concernant le démon quelque soit sa spécificité. Si un nœud continuellement déclenchable n'est jamais choisi par le démon, alors l'exécution de l'algorithme peut ne pas se dérouler correctement. Pour résoudre ce problème, la notion d'équité a été introduite. L'équité du démon peut être forte ou faible.

Définition 1.1.15 (Démon faiblement équitable)
Le démon est dit faiblement équitable (unfair). Si un nœud u est continuellement déclenchable ($u \in N$) à partir d'une configuration i_i, alors il existe une configuration i_j telle que $i_i \rightarrow i_j$ où le démon est contraint de le choisir ($u \in Q$) pour exécuter une règle.

Définition 1.1.16 (Démon fortement équitable)
Le démon est dit fortement équitable (fair) si $\forall u \in N$ infiniment déclenchable à partir d'une configuration i_i, alors il existe une configuration i_j telle que $i_i \rightarrow i_j$ où u est choisi par le démon pour exécuter une règle.

Si aucune des deux formes d'équité précédemment décrites n'est prise en compte alors le démon est dit inéquitable.

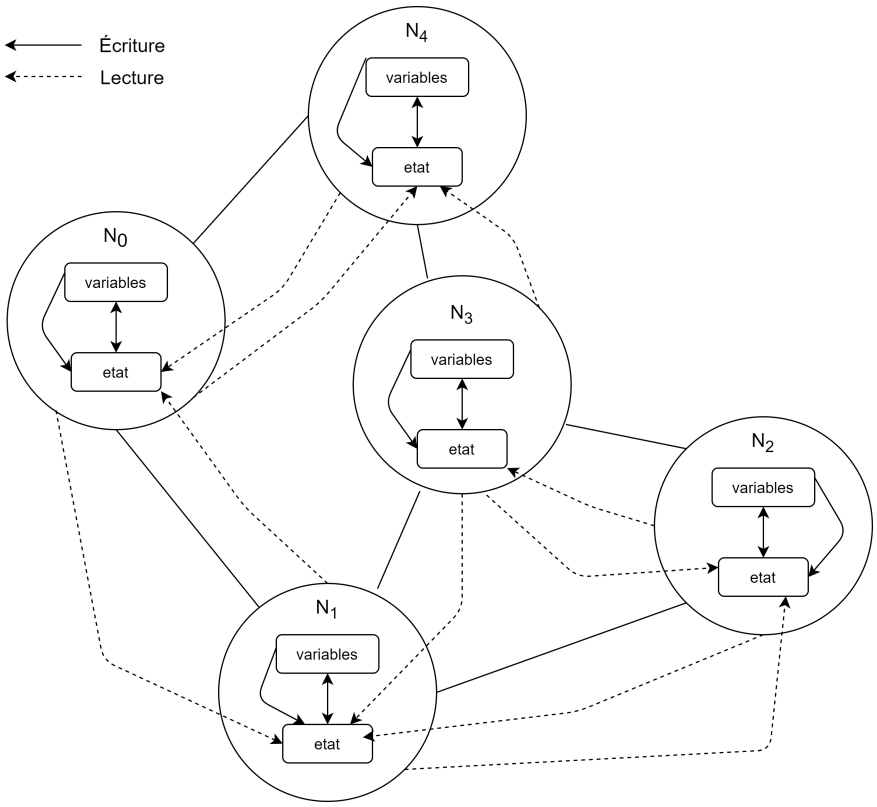


FIGURE 1.3 – Modèle à états

1.1.5.2 Modèle à registres

Le modèle à registres est le deuxième modèle de communication utilisant la mémoire partagée avec le modèle à états. Il présente les mêmes caractéristiques que le modèle à états à la différence que les échanges d'informations entre les nœuds se fait via des registres associés aux canaux de communication. Ainsi pour tout couple (u, v) de nœuds communiquant, il est associé un couple de registres $R_{(u,v)}; R_{(v,u)}$ tel que u dispose des droits d'écriture et de lecture sur son propre registre et uniquement du droit de lecture sur le registre du nœud v. Il en est de même pour v qui a les droits en écriture et lecture sur ses registres, et de lecture sur le registre du nœud u.

Comme le montre la figure 1.4, le nœud n_0 écrit sur son registre pour envoyer une information au nœud voisin et lit les données sur les registre du nœud voisin. Les algorithmes s'écrivent de façon similaire à ceux du modèle à états mais avec la caractéristique de considérer des registres.

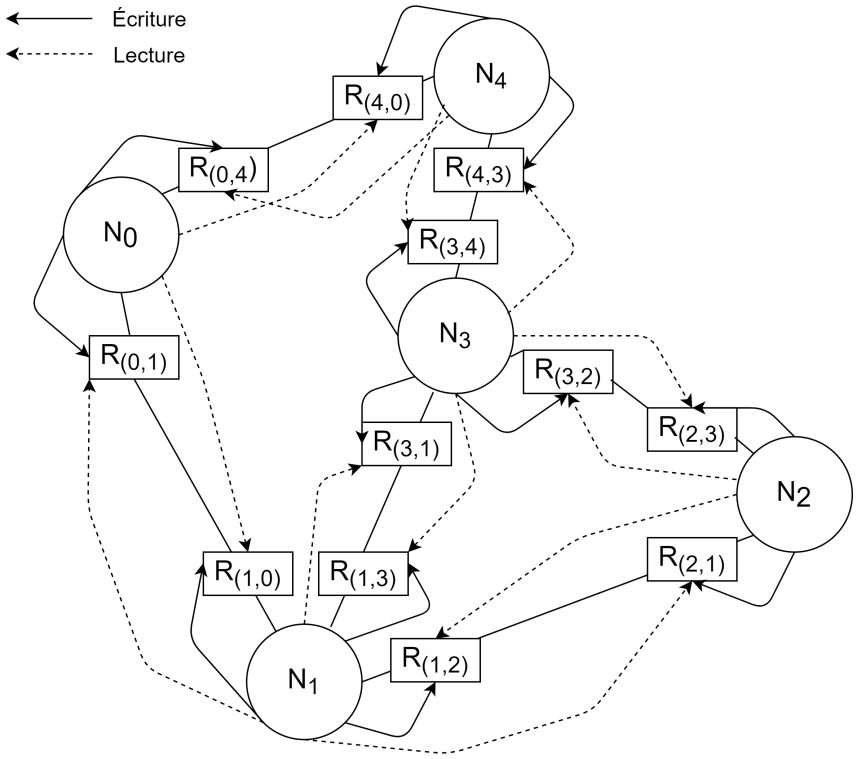


FIGURE 1.4 – Modèle à registres

1.1.5.3 Modèle à passage de messages

Dans le modèle à passage de messages, les nœuds communiquent entre eux par émission et réception de messages via des canaux de communication. Dans un tel modèle comme le montre la figure 1.5 chaque couple de nœuds (u, v) est relié par un canal. Les canaux de communication d'un nœud u sont étiquetés par un identifiant compris entre $[0, \gamma]$ ou γ représente le degré du nœud, autrement dit le nombre de voisins. Il y a plusieurs façons de décrire la taille des canaux de communication, elle peut être bornée, finie, non bornée, infinie. Si la taille des canaux est bornée alors l'écriture sur un canal est bloquante si la file d'attente est remplie. Pour tout couple de nœuds communiquant dans ce modèle, l'ordre de réception des messages peut être identique à leur ordre de transmissions. Dans ce cas, les transmissions respectent la caractéristique dite **F**irst **I**n **F**irst **O**ut(FIFO). Autrement dit, il y a un ordre sur les messages. Le premier émis est le premier reçu. Cependant cela peut ne pas être le cas et l'ordre des messages reçus peut être différent de l'ordre des messages transmis. Dans cette hypothèse, les transmissions via les canaux de communications ne sont pas FIFO.

Le modèle à passage de messages peut être synchrone ou asynchrone.

1. Dans un modèle à passage de messages synchrone, tous les nœuds envoient simultanément leurs messages via le synchroniseur d'horloges. Tout message envoyé à un temps t à un voisin est reçu et traité par ce voisin au temps $t + 1$, le temps global du système, commun à tous les nœuds. Ce qui ramène à supposer que l'horloge globale du système synchronise cycle par cycle chaque événement et chaque action.
2. Un modèle à passage de messages est dit asynchrone si et seulement si les délais d'acheminement des messages via les canaux de communication sont finis mais non bornés. Un message prend alors un temps fini mais non borné dans un canal de communication reliant un nœud à son voisin.

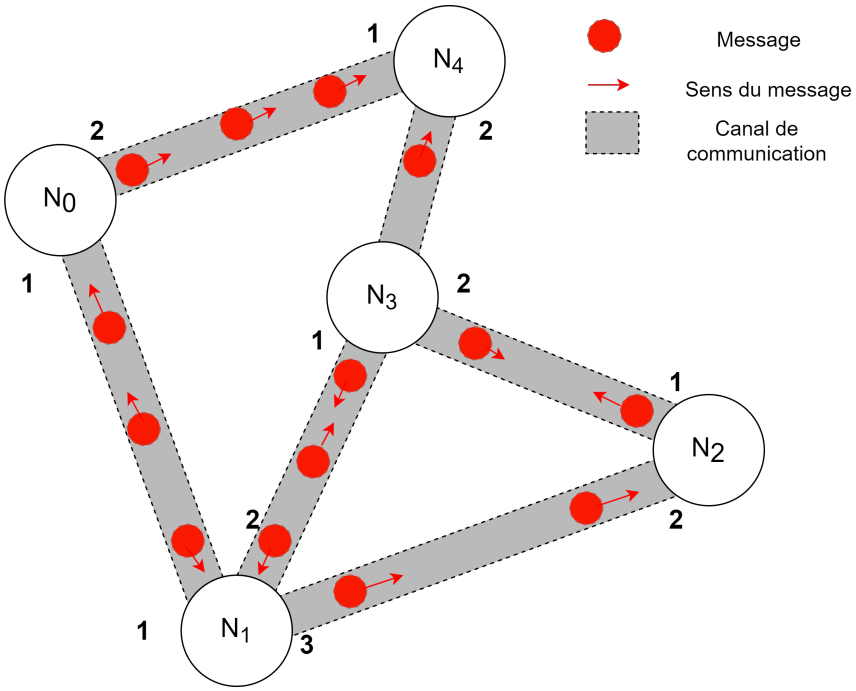


FIGURE 1.5 – Modèle à passage de messages

1.1.5.4 Choix du modèle à passage de message

Chacun des trois modèles que nous venons de présenter est une abstraction qui peut être utilisée comme modèle de communication dans la conception de l'écriture d'algorithmes distribués. Il présente chacun des propriétés qui leur sont propres et peuvent être adéquates pour une situation donnée.

Le modèle à états permet de faire abstraction des problèmes liés à la communication et d'éviter l'utilisation de protocoles de communication. Un tel modèle offre des caractéristiques intéressantes comme : tous les nœuds peuvent lire la mémoire de leurs voisins. Le modèle à registres permet une première approche vers l'utilisation et la gestion de messages ainsi que l'échange d'informations entre les nœuds.

Contrairement à un modèle à états, dans un modèle à registres, un nœud choisi les données qu'il souhaite transmettre à un voisin et les met à disposition de ce dernier grâce au registre qu'ils partagent ensemble. Ces deux modèles qui constituent la communication en mémoire partagée modélisent parfaitement des systèmes distribués comme les machines parallèles et sont adaptés et utilisables dans le contexte de la programmation parallèle multi-threads.

Le modèle à passage de messages prend en considération l'ensemble des spécificités relatives aux échanges de messages entre différents nœuds du système : incohérence entre les données reçues et celle présentes dans la mémoire de l'émetteur, l'altération possible du message lors de sa transmission et la perte ou la réplications de messages. Nous nous intéressons à ce modèle car il est plus réaliste et plus adapté aux contraintes des systèmes distribués que nous traitons dans nos travaux de recherche. Nous cherchons à nous rapprocher le plus possible des conditions réelles, nous optons pour un modèle asynchrone à passage de messages.

Cependant il faut souligner que la transformation d'un modèle de communication à un autre reste possible bien que cela entraine des surcouts supplémentaires. En effet, la mémoire partagée peut être vue comme un lien de communication acceptant un seul message à la fois. De nombreux travaux proposent des méthodes pour adapter un algorithme conçu dans un modèle particulier vers un autre [GM91][KP93][DIM91].

1.2 Les différentes topologies classiques

Après avoir présenté les généralités autour des modèles de communication, d'exécution et de synchronisation, nous allons voir dans quelle topologie un système distribué peut évoluer. L'intérêt de savoir dans quelle topologie un système distribué évolue est que l'on peut choisir des algorithmes plus approprié pour répandre une information. Chaque topologie étant différente le temps de transmission d'un information, d'organisation va varier en fonction de celle-ci. Il est important de connaitre les topologies existantes, les contraintes, les avantages de chacune afin d'évaluer l'impact quelle va avoir sur une solution.

1.2.1 En étoile

Dans un réseau en étoile, chaque nœud est connecté à un nœud central. Dans sa forme la plus simple, un nœud central sert de conduit pour transmettre les messages comme décrit par Roberts et al. [RW70]. Le réseau en étoile est l'une des topologies de réseau informatique les plus courantes. Il a pour avantage d'être fiables. Si une connexion échoue, cela n'affecte pas les autres nœuds. Il y a peu de collisions de données car chaque nœud dispose de son propre canal de communication vers le nœud central. Cette architecture prend la forme d'une étoile comme le montre la Figure 1.6.

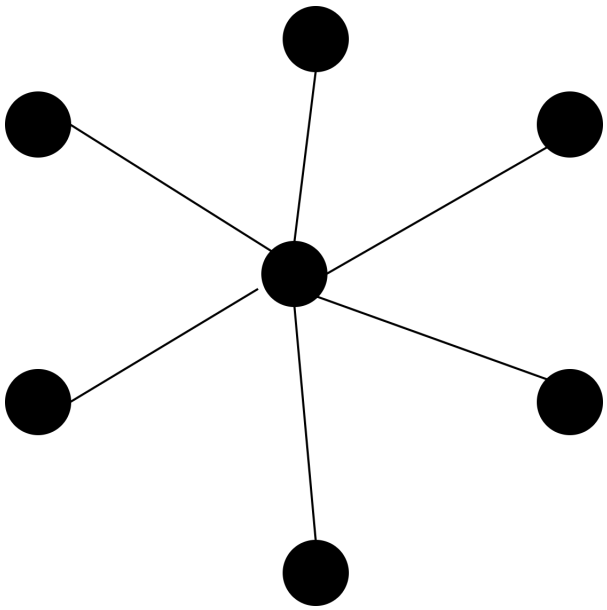


FIGURE 1.6 – Topologie en étoile

1.2.2 Linéaire et semi-linéaire

Une topologie linéaire met en forme les nœuds du système en forme de ligne chaque nœud à deux voisin celui de "gauche" et celui de "droite". Le lien peut être bidirectionnelle. Une topologie semi-linéaire est basé sur une topologie linéaire ou chaque nœud peut être le début d'une topologie linéaire comme le montre les figures 1.7 et 1.8.

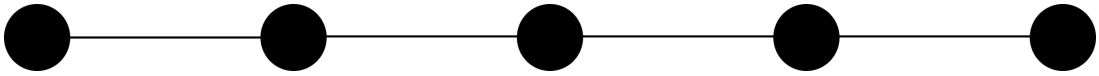


FIGURE 1.7 – Topologie linéaire

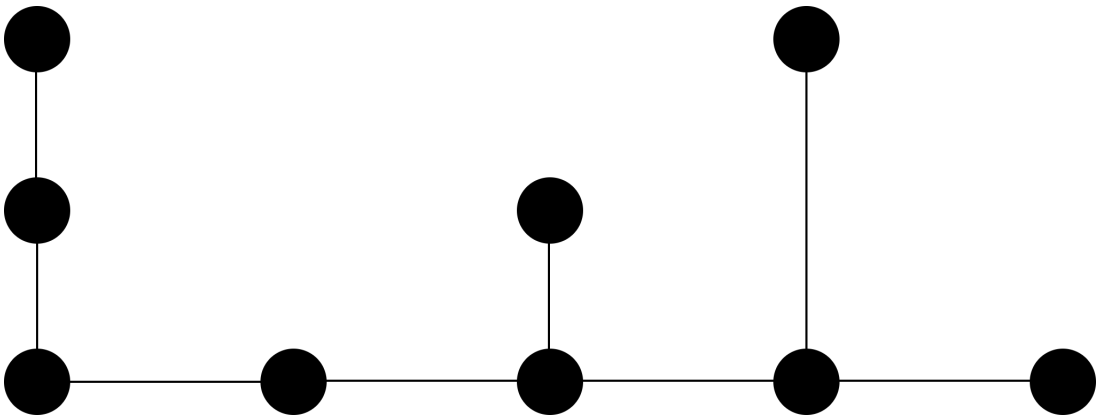


FIGURE 1.8 – Topologie semi-linéaire

1.2.3 En anneau

Une topologie en anneau est une topologie particulière d'une topologie linéaire. Le premier et le dernier nœud sont lié entre eux. Un anneau de taille n est un réseau où les sites numérotés de 0 à $n-1$ sont tels qu'il n'existe un canal de communication qu'entre i et $i+1$ modulo n comme décrit par Tel dans [Tel01]. Les anneaux sont souvent utilisés pour des calculs de

controle distribué. Il existe aussi des réseaux physiques comme des *Token Rings* comme le montre Tanenbaum dans [Tan96].

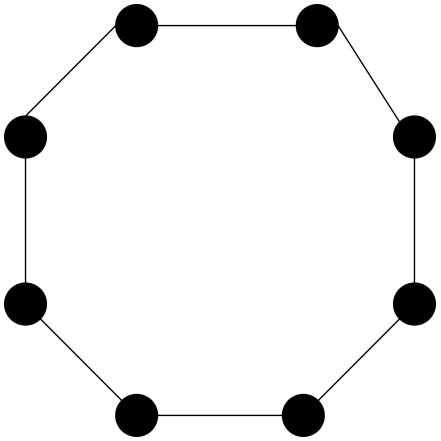


FIGURE 1.9 – Topologie en anneau

1.2.4 Arbre

Un arbre de n nœuds est un réseau connexe sans cycle contenant exactement $n-1$ liens de communication. Il n'existe alors entre deux sites qu'un unique chemin simple. Il est possible de construire un arbre couvrant sur tout réseau arbitraire connexe.

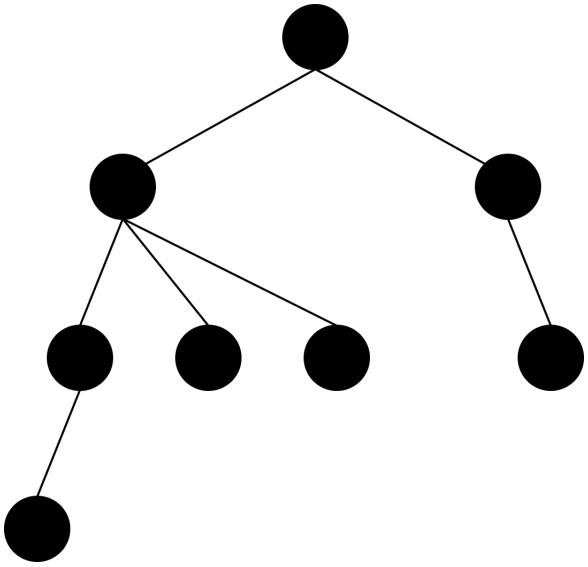


FIGURE 1.10 – Topologie en arbre

1.2.5 Grille

Dans une topologie de grille régulière, chaque nœud du réseau est relié à deux voisins selon une ou plusieurs dimensions. Si le système est unidimensionnel et que la chaîne de nœuds est connectée pour former une boucle fermée, la topologie résultante est appelée anneau. (figure 1.9) Ici, nous avons une grille 4x5 dans la figure 1.11. Formellement une grille de $n*m$ nœuds est un réseau pour lequel chaque nœud est étiqueté (i, j) tel que $i < n, j < m$. Deux nœuds (i, j) et (i_0, j_0) sont voisins si : $((i = i0) \wedge (j = j_{0+1})) \vee ((i = i_0) \vee (i = j_0 - 1)) \vee ((i = i_0 + 1) \wedge (j = j_0)) \vee ((i = i_0 - 1) \wedge (j = j_0))$.

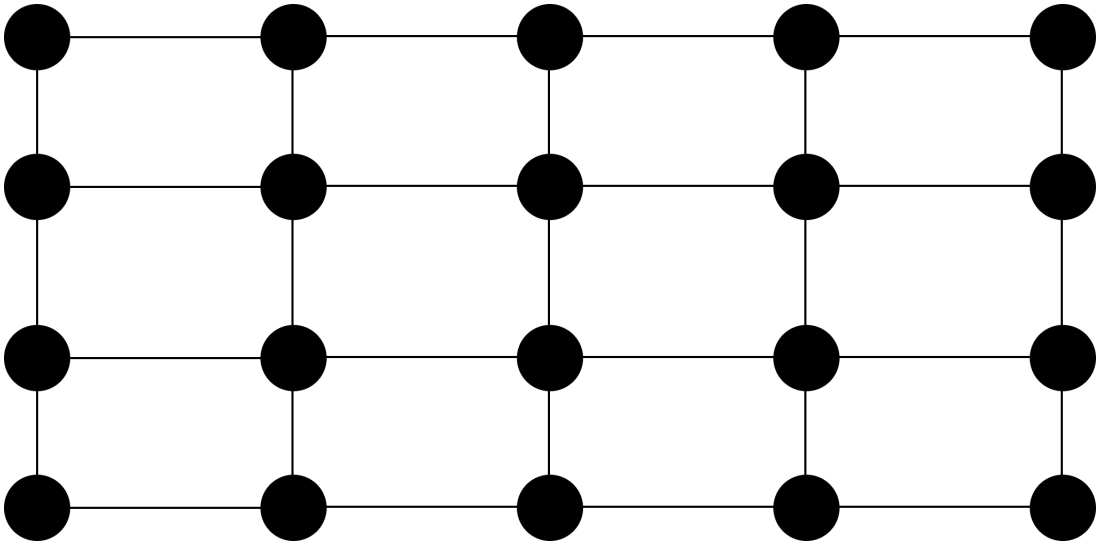


FIGURE 1.11 – Topologie en grille

1.2.6 Graphe complet ou clique

En théorie des graphes, un graphe complet, ou "clique", est un graphe simple dont tous les sommets sont adjacents deux à deux, c'est-à-dire que tous couples de sommets disjoints est relié par une arête. Si le graphe est orienté, on dit qu'il est complet si chaque paire de sommets est reliée par exactement deux arcs (un dans chaque sens) comme décrit par Diestel dans [Die97] (Figure 1.12).

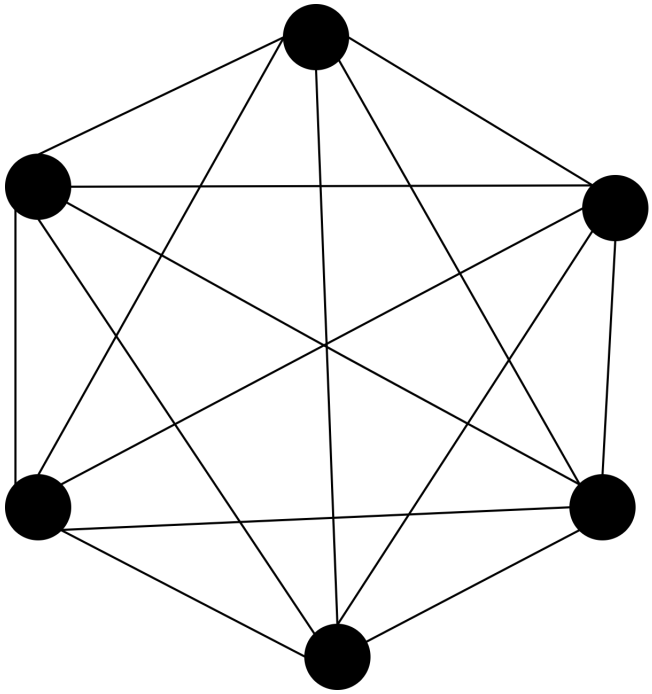


FIGURE 1.12 – Topologie en graphe complet

1.2.7 Algorithmes classiques

Dans le cadre des systèmes distribués, plusieurs algorithmes constituent les bases et les principes pour l'élaboration d'autres solutions. Certains problèmes classiques sont étudiés dans plusieurs ouvrages parmi lesquels nous pouvons citer [Lav95][Ray85][Tel01][AW04]. Nous listons des types d'algorithmes.

- L'exclusion mutuelle : L'exclusion mutuelle est le nom donné aux différents problèmes de concurrence dans un système distribué. L'exclusion mutuelle a pour objectif d'éviter les interférences des accès des différents nœuds du systèmes à une ressource partagée entre ces nœuds. Cette solution consiste donc à attribuer équitablement une ressource critique qui ne peut donc pas être détenu par un même nœud du système. Les algorithmes de référence d'exclusion mutuelle qui ont été proposés se classent en deux grandes familles.
  1. La première est celle qui se base sur des solutions à base de jeton. Il existe une notion appelée *sureté* définie en 1.2.1.

     Cette propriété est trivialement assurée par l'unicité du jeton qui circule dans le système : le nœud possédant le jeton dispose du droit d'utiliser la section critique. Il y a une autre notion et propriété à respecter appelée la *vivacité* définie en 1.2.2.

     Ainsi un déplacement séquentiel du jeton entre les différents sites demandeurs de la section critique est imposé soit par un mouvement perpétuel, soit par des requêtes explicites. Dans cette famille d'algorithmes nous pouvons citer entre autres [Le 77][LT87] [HPR88].

2. La deuxième famille est à base de permissions. Tous les nœuds sont associés à un ensemble de nœuds auxquels il doit demander la permission avant d'utiliser la ressource critique/partagée. Ces demandes sont faites via des messages sous forme de requête. Dans cette famille d'algorithmes nous avons les ouvrages suivants [RA80] [Mae85] [Sin91].

Définition 1.2.1 (Principe de sureté)
Le principe de sureté assure qu'à tout instant, il y a au plus un seul nœud du système disposant de la ressource partagée/critique.

Définition 1.2.2 (Principe de vivacité)
Le principe de vivacité assure que tout autre nœud demandant la ressource partagée est servi au bout d'un temps fini.

- Le principe d'élection de leader : L'objectif est de choisir un nœud afin de lui conférer une élévation de privilèges. Ainsi, dans une configuration initiale du système, tous les nœuds actifs sont dans le même état appelé *candidat*. Dans une configuration finale, un seul nœud est dans un état appelé *élu*, tous les autres sont dans un état différent du précédent appelé *battu*. Le nœud élu à la fin d'une exécution est alors appelé *leader* et est donc l'élu de cette élection. Le problème de l'élection a été posé et résolu pour la première fois en 1977 par LeLann dans [Le 77], puis par Gallager dans [Gal77].
- Les algorithmes à vagues : Ces algorithmes sont employés dans la diffusion d'une information, la synchronisation, des calculs locaux, etc. Un algorithme à vagues peut être constitué d'une seule vague ou d'une succession de vagues. Dans cette dernière, l'algorithme fonctionne de façon redondante : à chaque fois une exécution se termine, une nouvelle commence. Une instruction particulière appelée *décision* est appelée pour terminer une exécution. Le concept de vague a été créé en tant que tel par Chang dans [Cha82] puis approfondie par Schneider et Lamport dans [SL85] puis par la suite par Helary et *al.* dans [Hél+86].
- La détection de propriété et de terminaison : c'est un mécanisme de contrôle qui permet de vérifier une propriété de stabilité globale. Les algorithmes de détection de terminaisons montrent ainsi qu'un ensemble de nœuds vérifie le prédicat de terminaison. Les solutions de référence proposées dans [DS80][DFG83][Mis83][Ran83] commencent par structurer cet ensemble de façon à pouvoir le parcourir pour vérifier la propriété de terminaison. Les structures de base utilisées sont l'anneau [DFG83], un arbre couvrant [DS80], un circuit quelconque tiré du graphe de base [Mis83][Ran83]. Pour la structure d'arbre couvrant, le mode de parcours utilisé est le calcul diffusant alors que pour l'anneau ou le circuit, un jeton circulant est utilisé pour parcourir la structure.

1.3 Réseaux avec infrastructure et sans infrastructure

Dans les réseaux informatiques, nous distinguons deux grandes familles de réseaux, les réseaux avec infrastructure et les réseaux sans infrastructure comme décrit par Kumar dans [Kum08].

1.3.1 Réseaux avec infrastructure

Dans les réseaux avec infrastructure, il existe une entité centrale, appelée coordinateur qui a pour rôle d'organiser toute l'activité du réseau. L'infrastructure la plus commune est celle en

étoile ou d'étoile en étoile (Figure 1.13). Les réseaux avec infrastructure sont séparés en cellules, appelées zones. Chaque zone dispose d'un coordinateur appelé station de base qui joue le rôle de l'entité centrale. Toutes les communications passent par la station de base. Les nœuds peuvent être fixes ou mobiles, connectés par une interface filaire ou sans fil. La particularité de cette famille de réseau est que toutes les communications passent obligatoirement par la station. Le système de communication est basé sur l'utilisation de réseaux filaires mais aussi de stations de base, avec une longue portée pour couvrir l'ensemble des nœuds mobiles. Ainsi les réseaux avec infrastructure nécessitent une importante logistique pour leur déploiement [Hec07] [IMM08] [Ini18].

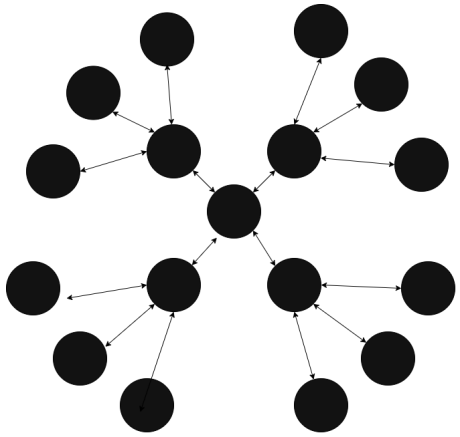


FIGURE 1.13 – Modèle avec infrastructure

1.3.2 Réseaux sans entité centrale / infrastructure

Dans les réseaux sans infrastructure, il n'y a pas d'entité centrale comme dit par Ilyas dans [Ily17]. Les nœuds communiquent directement entre eux. L'absence d'infrastructure oblige les nœuds du réseau à participer à la découverte et à la maintenance des routes dans le but d'acheminer les informations. Un réseau sans infrastructure peut être constitué de nœuds mobiles avec des interfaces sans fil. Tout nœud peut communiquer avec l'ensemble des autres nœuds se trouvant dans la zone de couverture représentée par sa portée radio. Les nœuds du réseaux peuvent être de même type, dans ce cas on parle de réseau homogène. Mais le réseau peut aussi être constitué de nœuds de types différents. Dans ce cas, le réseau est dit hétérogène. Les réseaux sans infrastructure sont plus contraignants que les réseaux avec infrastructure. La mobilité des nœuds conduit à de fréquents changements topologiques, d'où la nécessité de recalculer fréquemment les tables de routage.

Remarque Les nœuds d'un réseaux avec ou sans infrastructure peuvent avoir des ressources limitées comme la capacité de calcul, de stockage et énergétique, ce qui est le cas dans nos recherches. Toutes ces contraintes rendent difficile l'établissement de protocoles de communication.

1.4 Caractéristiques des réseaux ad hoc

Les réseaux ad hoc sont des réseaux sans infrastructure [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13]. Ce sont des systèmes distribués complexes qui rassemblent un important nombre de nœuds qui peuvent être mobiles et équipés d'interfaces sans fil pour les communications. Dans le cas où les nœuds sont tous mobiles et munis d'interfaces de communication sans fil, on parle alors de réseau *Mobile Ad hoc Network* (MANET).

1.4.1 Les différentes possibilités de communication

Dans les réseaux ad hoc il y a plusieurs techniques de communication

- Unicast
- Multicast
- Broadcast

Dans le broadcast, appelée également technique de diffusion, un message envoyé par un nœud est reçu par tous ses voisins. Dans une communication en multicast, un message envoyé par un nœud est reçu par un groupe de nœuds parmi ses voisins. Pour finir la technique de l'unicast décrit par Ilyas [Ily17], appelée aussi communication point à point, un message envoyé par un nœud est destiné à un seul de ses voisins. Toutefois, dans chacune de ces différentes techniques, les communications entre les nœuds ne nécessitent aucune infrastructure fixe. Ainsi, les réseaux ad hoc représentent une alternative aux réseaux avec infrastructure, notamment pour des applications qui requièrent un déploiement dynamique du réseau.

1.4.2 Spécificités des réseaux ad hoc

Dans cette partie nous présentons les réseaux ad hoc qui sont des cas particuliers dans le domaine des systèmes distribués [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13]. Les réseaux ad hoc présentent plusieurs spécificités qui rendent difficile l'établissement des protocoles de communication. Parmi ces spécificités, nous pouvons présenter celles-ci :

— **La topologie dynamique** : La topologie d'un réseau ad hoc est dynamique. Les nœuds peuvent se déplacer de façon libre et arbitraire. Un nœud peut se connecter ou se déconnecter du réseau de manière impromptu à tout moment. En conséquence, la topologie du réseau peut changer à des instants imprévisibles, d'une manière rapide et aléatoire. Cela affecte fortement la disponibilité des chemin de routage , les protocoles de routage doivent s'adapter à la mobilité des nœuds [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13].

— **La volatilité du médium de communication** : la communication dans un environnement sans fil est différente de celles des réseaux câblés par delà deux points. Le premier est que les ondes radio ne sont pas gérables comme dans un médium câblé, elles ont donc un comportement variable et imprévisible. Le deuxième point est qu'un médium sans fil est un médium de diffusion, tous les nœuds qui sont dans la portée de transmission de l'émetteur peuvent recevoir le message [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13].

— **L'hétérogénéité des nœuds** : Les nœuds dans un réseau ad hoc peuvent correspondre à une multitude d'équipements. Ces nœuds du réseau peuvent avoir des différences en termes de capacité de traitement liée à leur CPU, RAM, mais aussi vis à vis de leurs mobilités qu'ils soient lents ou bien rapides, etc. Cependant, quelle que soit l'hétérogénéité des nœuds, ils doivent être en mesure de communiquer et de coopérer pour accomplir les tâches requises dans le réseau [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13].

— **La bande passante limitée** : dans un réseau ad hoc, la bande passante est limitée. En effet, les nœuds du réseau se partagent le médium de communication pour les échanges et routage d'informations. Ce partage fait que la bande passante réservée à un hôte est réduite [Ily17] [CCL03][Agg04][Fri13][BK07] [SBP13].

1.5 Les réseaux de capteurs sans fil (RCSF)

Après avoir présenté les réseaux ad hoc, nous présentons dans cette section les réseaux de capteurs sans fil (RCSF) avec leurs caractéristiques et leurs contraintes.

1.5.1 Définition du concept des RCSF

Les réseaux de capteurs sans fil sont un type particulier de réseau ad hoc [Aky+02][AY07][KW07][CA06] [SMZ07][Dre08][LP09][XCL10][MSH11]. Ils consistent en un grand nombre d'appareils déployés dans une zone géographique souvent appelé une zone d'étude, dotés de la capacité de récupérer des informations sur l'environnement comme :

— Température

— Pression,

— Luminance

— Humidité

— Accélération

— etc.

Sur ces données récupérées le *device* peut faire des traitements sur celles-ci puis coopère avec d'autre appareils pour acheminer le résultat de ces traitements vers une station de base, antenne principale. Ces appareils sont appelés **capteurs** et ont connu un important développement grâce aux avancées récentes dans la technologie de la micro-électromécanique, de la micro-électronique et des communications sans fil.

Les principales caractéristiques d'un RCSF sont les suivantes :

— Des contraintes de consommation d'énergie pour les nœuds utilisant des batteries ou la collecte d'énergie
— Une certaine mobilité des nœuds
— Hétérogénéité des nœuds
— Homogénéité des nœuds
— Possibilité de déploiement à grande échelle
— Capacité à résister à des conditions environnementales difficiles
— Facilité d'utilisation
— Optimisation intercouche

Dans un réseau de capteurs (Figure 1.14), les données collectées par les capteurs sont remontées avec un routage multi-sauts de proche en proche vers un centre de collecte appelé station de base (base station, BS). Celle-ci est aussi désignée sous le nom de Puits (Sink) et est connectée via Internet ou par satellite à un centre d'administration où il y a des outils de visualisation et de contrôle sur l'activité du réseau.

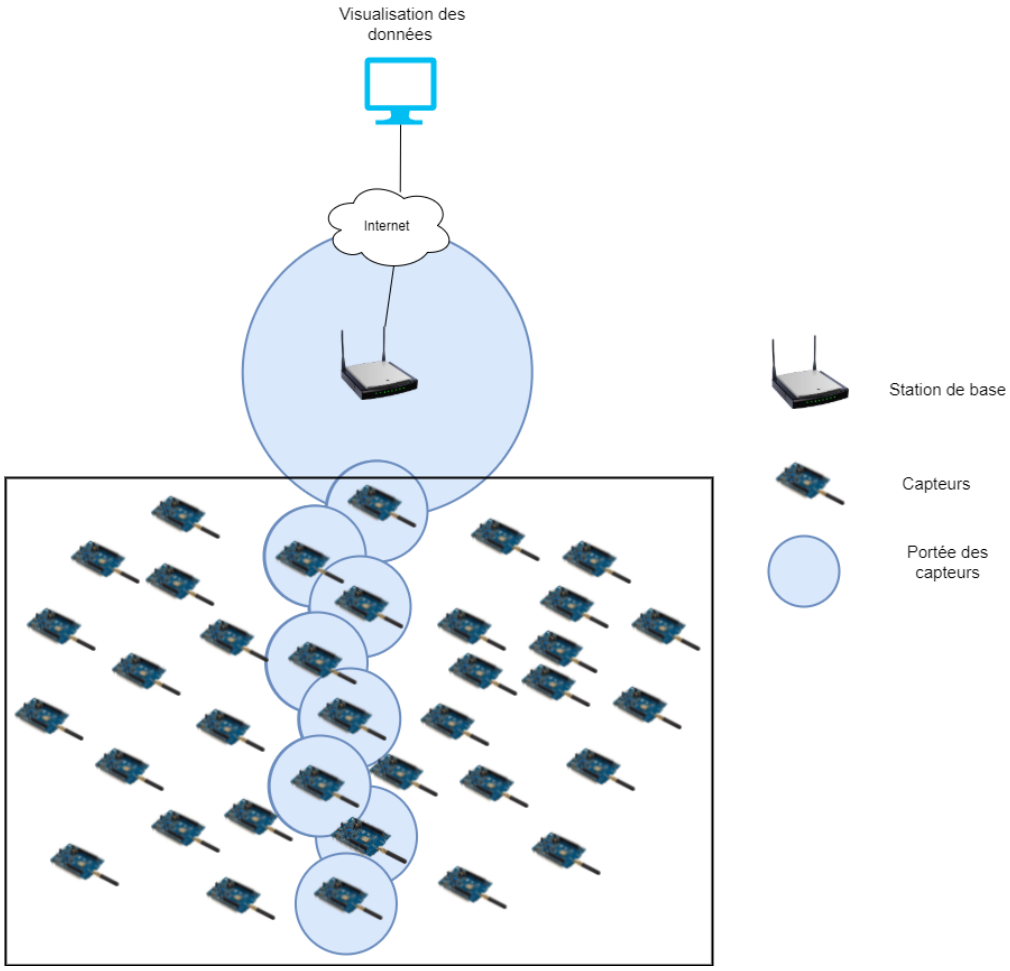


FIGURE 1.14 – Réseaux de capteurs sans fil

1.5.2 Applications des réseaux de capteurs

Les réseaux de capteurs sans fil du fait de la taille réduite des capteurs et leur faible coût de production, ainsi que de l'absence d'infrastructure fixe, etc. offrent de nombreuses applications pratiques. Les opérations tactiques comme les opérations militaires, de secours, ou d'explorations de zones hostiles trouvent dans les réseaux de capteur un potentiel intéressant. La technologie des capteurs intéresse également la recherche d'applications civiles.

— **Les services d'urgence** : Dans le cas de catastrophes naturelles tel que des feux de forêts, tremblements de terre ou inondations, la mise en place d'infrastructure demande un temps conséquent ou insuffisant. Les réseaux de capteurs offrent une solution alternative pour une mise en place et un déploiement rapide.

— **Applications militaires** : Dans le cas d'infrastructure fixe, la rapidité de déploiement est l'aspect recherché dans le cadre d'opérations militaires qui font que les réseaux de capteurs sans fil sont une solution efficace.

— **Applications médicales** : Les réseaux de capteurs peuvent être utilisés également pour la surveillance permanente ou momentanée d'organes vitaux grâce à des micro-capteurs implantés dans un patient.

— **Applications environnementales** : Dans le cadre de surveillance de zones tempérées, de la qualité de l'air atmosphérique ou le suivi d'animaux, les réseaux de capteurs offrent d'intéressantes applications.

— **Application domestiques** : Les RCSF présentent aussi des applications pratiques pour la surveillance et le contrôle d'équipements domestiques dans le cadre de la domotique à distance tels que les appareils de chauffage, contrôle de température de pièces, contrôle de ventilateurs, contrôleur d'ampoules connectées, etc.

1.5.3 Modèle énergétique du module radio

Dans les réseaux de capteurs sans fil, le modèle énergétique de référence a été proposé par Heinzelman et *al* dans [HCB00]. Actuellement c'est le modèle couramment le plus utilisé dans les travaux de recherche sur la consommation énergétique des capteurs. Pour cette raison, nous l'utilisons dans le chapitre quatre lors de nos mesures énergétiques.

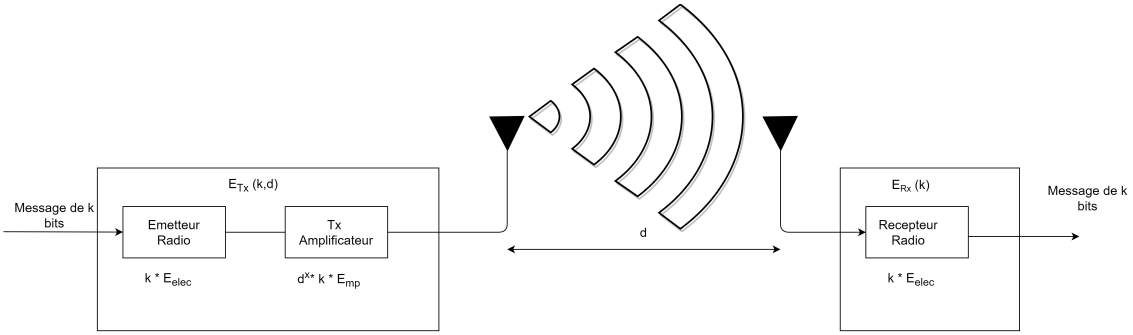


FIGURE 1.15 – Schéma du module radio de Heinzelman et *al*[HCB00]

Heinzelman et *al* décrivent un modèle radio, illustré dans la Figure 1.15 où un nœud capteur consomme une quantité d'énergie E_{Tx} pour transmettre un message de longueur l bits sur une distance d mètres. Comme décrit dans l'équation suivante, quand la distance dépasse un certain seuil d_0, le modèle de consommation est différent car l'énergie nécessaire est plus importante.

$$E_{\mathrm{Tx}}(k,d)\begin{cases}(E_{\mathrm{elec}}*l)+(\epsilon_{\mathrm{mp}}*l*d^4), d\leq d_0\\(E_{\mathrm{elec}}*l)+(\epsilon_{\mathrm{fs}}*l*d^2), d\geq d_0\end{cases} \tag{1.1}$$

d_0 est défini ainsi :

$$d_0 = \sqrt{\frac{\epsilon_{\text{fs}}}{\epsilon_{\text{mp}}}} \tag{1.2}$$

A chaque réception de message, un nœud consomme une quantité d'énergie E_{Rx} comme illustré dans l'équation suivante :

$$E_{\text{Rx}}(l) = E_{\text{elec}} * l \tag{1.3}$$

Dans le cas où il y a une agrégation de message l'énergie dépensée est décrite ainsi :

$$E_{\text{DA}}(l) = E_{\text{agg}} * l * m \tag{1.4}$$

Les valeurs des paramètres utilisés dans les équations précédentes sont mentionnées dans le tableau 1.1 E_{elec} représentent l'énergie nécessaire pour activer les circuits électroniques du module radio lors des communications au niveau de l'émetteur et du récepteur. En plus de cette quantité E_{elec}, l'émetteur consomme une énergie supplémentaire dépendant de la longueur l du message et de la distance de transmission d.

TABLE 1.1 – Paramètre du module radio

Paramètres	***Définition***	***Valeurs***
E_{elec}	Énergie pour faire fonctionner la radio	50 nJ/bit
ϵ_{fs}	Modèle d'espace libre de l'amplificateur de l'émetteur	10 $pJ/bit/m^2$
ϵ_{mp}	Modèle multi-path de l'amplificateur de l'émetteur	0,0013 $pJ/bit/m^4$
l	Taille d'un message	2000 bytes

1.5.4 Contraintes dans les réseaux de capteurs sans fil

En plus d'avoir toutes les spécificités des réseaux ad hoc, les RCSF présentent deux contraintes fondamentales qui sont au centre de nos travaux de recherche : la faible capacité énergétique et la vulnérabilité aux pannes. Nous allons séparer cette section en deux parties consacrées à ces problèmes respectifs.

1.5.4.1 Contrainte énergétique

Comme nous l'avons dit précédemment, du fait de sa taille réduite, un capteur est doté de batterie avec une quantité énergétique très limitée. De plus, l'hostilité qui caractérise souvent les zones de déploiement fait qu'il est difficile, voire même impossible de recharger ou de changer la batterie des capteurs. À cela s'ajoute l'énergie qu'un capteur consomme principalement par trois opérations :

— Capturer les données

— Les traitements en calcul sur le CPU

— Les communications

Les communications sont la source de consommation énergétique la plus importante par rapport au coût du CPU et les opérations de captage. Pottie et Kaiser dans [PK00], ont montré que la consommation énergétique requise pour la transmission d'un message de 1 Kb sur une distance de 100 mètres est équivalente à 3 millions d'instructions CPU dans un processeur 100 $MIPS/W$. Avec ces contraintes, il s'avère que les techniques de communication classiques ne s'adaptent pas aux réseaux de capteurs. Ainsi, vu la nature sensible des applications qu'offrent les réseaux de capteurs sans fil, d'une part, et les faibles capacités énergétiques des capteurs d'autres part. Il est nécessaire de minimiser les communications dans le but d'optimiser la

consommation énergétique afin de prolonger au maximum l'autonomie d'un nœud dans la vie du réseau. Pour réduire les communications, une solution efficace consiste à structurer le réseau [EWB87][AKY90][Bas99][BCV03][DLV08][BPR11].

1.5.4.2 Contrainte sur la vulnérabilité aux pannes

Au-delà de l'optimisation sur la consommation énergétique sur un capteur dans un RCSF, il reste un autre élément important : la vulnérabilité aux pannes. En effet, les nœuds capteurs d'un RCSF sont susceptibles de tomber en panne du fait de l'épuisement total de leur énergie ou du dysfonctionnement d'un de leurs composants. La mobilité des nœuds peut entrainer des ruptures de liens de communication conduisant ainsi à des changements de topologie du réseau. Comme montré dans plusieurs études [LWJ05] [ZXM07] la tolérance aux pannes transitoires est une problématique dans les réseaux de capteurs sans fil. Dans [LNS09], Liu et al. et al définissent la tolérance aux pannes dans les réseaux de capteurs sans fil comme étant la capacité à assurer le service requis même en présence de pannes transitoires dans le réseau. Dans un système distribué comme les RCSF, les auteurs dans [JM11] et Johnen et al. montrent que la dynamicité est une solution efficace de tolérance aux pannes transitoires. De plus, dans [DFG06] Drabkin et al. présentent un modèle de faute qui peut occasionner des pannes transitoires dans les réseaux de capteurs sans fil et que l'auto-stabilisation peut gérer efficacement. Les fautes décrites sont celles-ci :

1. La topologie du réseau peut changer à tout instant : mobilité des nœuds, ajout ou rupture de liens de communication.
2. L'état d'un nœud ou la configuration du système peut subir des corruptions quelconques. L'algorithme est supposé sûr.
3. Les canaux de communications peuvent subir des corruptions quelconques et contenir des données erronées. Comme par exemple lors de l'inter-arrivée de plusieurs messages provoquant une collision.
4. Tout nœud du réseau peut tomber à l'arrêt total pour une durée finie mais non bornée.

L'apparition de ces fautes est finie mais non bornée. Elle peut suivre n'importe quel ordre, n'importe quelle fréquence et peut survenir à n'importe quel moment.

Pour toutes les raisons évoquées dans cette partie, la réduction de communications et la tolérance aux pannes transitoires dans les réseaux ad hoc et donc des RCSF ont inspiré des chercheurs ces dernières années. Nos travaux de recherche s'inscrivent dans ce contexte.

1.6 Conclusion

Dans ce chapitre les bases de nos travaux de recherche que sont les systèmes distribués ont été présenté avec la représentation par les graphes, les modèles d'exécution par un système de transitions et les différents modèles de communications, le modèle à états, à registres, passage de messages. Ensuite quelques algorithmes classiques furent présentés. Suite à cela nous avons présenté les différentes topologies classiques : étoile, linéaire, anneau, arbre, grille, clique. Nous avons ensuite présenté les réseaux avec et sans infrastructure en se concentrant ensuite sur les réseaux ad hoc qui nous intéressent plus particulièrement. Ainsi que leurs caractéristiques et les techniques de communications associées. Après avoir présenté les réseaux ad hoc nous avons introduit les réseaux de capteurs sans fil. Ces derniers ont deux contraintes principales qu'il faut prendre compte pour des exigences applicatives. Premièrement, la contrainte énergétique est limité aux capteurs et deuxièmement, les pannes transitoires. Pour gérer ces deux contraintes dans le but d'améliorer les performances du réseau et de prolonger sa durée de vie, les chercheurs

se sont penchés sur plusieurs solutions de structuration, tolérance aux fautes et à base de dynamicité dans les réseaux ad hoc. Mais aussi sur des protocoles radio plus économes en énergie. Notamment dans le cadre de l'Internet des Objets où il existe plusieurs protocoles avec différentes consommations et spécificités.

Dans le prochain chapitre, nous étudierons les différents approches, protocoles mais aussi les problématiques de l'Internet des Objets.

Chapitre 2
Internet des objets

Résumé : *Nous présentons dans ce chapitre le concept d'Internet des Objets en commençant par son historique ainsi que ses différentes définitions données à travers le temps. Nous classons en 2 grandes familles les protocoles utilisés au sein de l'IoT, ceux gérés par un opérateur de réseaux de téléphonie mobile et ceux qui ne le sont pas. L'existence de différents protocoles radio s'explique par l'existence des différents aspects d'utilisations et d'applications. Pour comprendre les différents domaines où intervient l'IoT nous avons fait une liste non exhaustive des différents objets ainsi que des différents papiers scientifiques autour de certaines études au sein de chaque domaine d'application.*

Sommaire

2.1 Introduction

Le concept type de l'Internet des Objets (Internet of Things) devient de plus en plus présent, ce paradigme est né de l'idée de connecter différents objets à travers le réseau Internet afin d'améliorer certains aspects de la qualité de vie ou des usages des humains. Ainsi plusieurs organismes se sont intéressés à l'Internet des Objets afin de définir ses différents composants selon diverses architectures en couches mais aussi les concepts et technologies qui permettent de le déployer.
Ce chapitre va présenter une liste non exhaustive des définitions autour de l'Internet des Objets ainsi que certaines généralités. Sont ici présentés : les architectures des applications, les protocoles de communication gérés ou non par une couche MAC. L'Internet des Objets concerne plusieurs domaines d'applications tels que la domotique, l'agriculture intelligente, les villes intelligentes, les compteurs intelligents ainsi que la cybersanté. Dans le cadre de notre travail, nous nous sommes concentrés sur le protocole LoRa et LoRaWAN. Nous présentons les objectifs et motivations de notre étude ainsi que les différentes plateformes utilisées pour résoudre notre problématique.

2.2 Généralités sur l'Internet des Objets

2.2.1 Historique

Le terme Internet des Objets (IoT : Internet of Things) remonte à la fin des années 1990 lorsque Kevin Ashton l'a utilisé pour définir le lien entre la technologie RFID et l'Internet. Selon Ahston l'Internet des Objets n'est pas une vision mais plutôt un nouvel environnement capable de collecter des informations sans besoin d'être humain afin de comprendre l'environnement [Ash+09]. En 2008 l'alliance IPSO (Internet Protocol for Smart Objects), cette alliance regroupe les grands industriels technologiques. L'IPSO promeut l'utilisation du protocole IP (Internet Protocol) pour l'interconnexion des objets [VD08]. Le CISCO Internet Business Solutions Group a indiqué que l'IoT est né entre 2008 et 2009. Au moment où le nombre d'objets connectés à Internet est plus grand que le nombre d'humains tout en citant le nombre d'appareils connectés qui ne cesse de croitre [Eva11].

Plusieurs autres organismes ont reconnu l'Internet des Objets. Parmi ceux-ci on retrouve le Gartner Hype Cycle, un organisme de recherche et de conseil international qui suit les cycles de vie des technologies du moment. L'Internet des Objets a, d'après cet organisme, atteint le pic des attentes exagérées du cycle Gartnet Hype [FL11].

Depuis son lancement, l'Internet des Objets a fait l'objet d'une évolution et d'une croissance forte. Par la suite nous présentons divers organismes de standardisation mais aussi différentes commissions européennes et mondiales. L'Union Internationale des Télécommunications (UIT-T) a proposé en 2005 un rapport portant sur l'Internet des Objets [Wu+10]. Il y a également eu un ensemble de conférences comme à Zurich sur l'Internet des Objets en 2008 [Flo+08]. La National Intelligence Council (NIC) a présenté l'Internet des Objets comme une des six technologies civiles qui ont des répercussions potentielles sur les intérêts américains jusqu'en 2025 [Mad+15] [NGK16]. L'UIT-T a proposé le Y.2060 un document qui présente une vue d'ensemble de l'Internet des Objets en 2012 [htt12] ainsi qu'une liste de recommandations pour l'environnement Internet des Objets avec un autre document Y.2066 en 2014 [LUI14].

Il y a d'autres organismes de standardisation qui se sont intéressés à l'IoT ainsi que les différentes technologies associées (que nous présentons dans un autre section), comme l'Internet Engineering Task Force qui a crée le groupe de travail 6Lo en 2014 précédé par le groupe de travail 6LoWPAN [16a] [16b], pour adapter les technologies radio à l'IoT. Ce même organisme propose plusieurs

documents de références décrivant les challenges à considérer dans l'Internet des Objets[GKS18a] [GKS18b]. Actuellement, il n'y a pas de dénombrement mais seulement des estimations projetées sur l'année 2020 durant laquelle seraient à dénombrer 30 milliards d'objets connectés [Sta12].

2.2.2 Définitions

Parmi les différents organismes présentés dans la section précédente certains ont spécifié plusieurs définitions portant sur le paradigme de l'Internet des Objets. Nous présentons dans cette section les définitions proposées par ces organismes de standardisation internationaux.

Le groupe de travail SWG10 précédé par SWG6 de International Organisation for Standardization / International Electrotechnical Commission (ISO/IEC) a présenté une définition de l'Internet des Objets avec une spécification du champ lexical utilisé au sein de ce type d'environnement. L'Internet des Objets d'après cet organisme est :

Définition 2.2.1 (IoT selon ISO/IEC [115][Com17])
Un réseau d'objets physiques qui collectent et transmettent des données. Il s'agit d'une infrastructure d'objets inter-connectés d'humains et de ressources d'informations qui permettent de traiter les informations remontées par les objets et d'agir en conséquence.

L'ISO/IEC a défini des exigences pour l'Internet des Objets comme l'auto-configuration, l'identification unique, la connectivité, la fiabilité, etc.

L'Internet Engineering Task Force a défini ainsi l'Internet des Objets :

Définition 2.2.2 (IoT selon IETF [Kon+12])
Une idée générale étant celle de connecter des objets pour assurer des services, cela à travers différentes technologies pour aboutir à un service accessible depuis n'importe où et à n'importe quel moment [MBR15]. L'IETF considère l'Internet des Objets comme étant un réseau d'objets inter-connectés, adressables d'une manière unique et utilisant des protocoles standardisés pour la communication entre ces objets. L'IETF insiste sur les exigences spécifiées par l'ISO/IEC pour l'environnement IoT.

L'UIT-T, qui étudie l'environnement IoT et ses différents domaines d'application à travers le groupe SG20, a pris la relève suite à la clôture des travaux de groupe SG13. Le document Y.2060 fut défini à l'issue de ce groupe de travail, ce document définit l'Internet des Objet ainsi :

Définition 2.2.3 (IoT selon l'UIT-T Y.2060)
L'IoT est un réseau ubiquitaire disponible n'importe où, à n'importe quel moment et à n'importe qui [htt12]. L'Internet des Objets est une infrastructure globale de la société d'informations, permettant d'offrir des services avancés en inter-connectant des objets à travers des technologies de communications variées [MBR15].

L'UIT-T a défini des objets dans l'environnement IoT en tant qu'objets virtuels, et objets physiques. Un objet physique peut être représenté par un ou plusieurs objets virtuels et des objets virtuels peuvent exister sans la nécessité d'une correspondance physique. Ces objets peuvent communiquer entre eux par des passerelles, à travers un réseau de communication, ou directement. L'UIT-T a spécifié également dans[htt12] [LUI14] un certain nombre d'exigence comme la protection de la vie privée, la sécurité et l'identification.

Le NIST pour National Institue of Standards and Technology considère l'Internet des Objets

comme :

Définition 2.2.4 (IoT selon le NIST)
Étant un système cyber-physique [STc]. Ce système connecte des objets intelligents à travers de nouvelles techniques pour améliorer la qualité de vie des utilisateurs [Mat17].

L'OASIS pour Organization for the Advancement of Structured Information Standards s'est concentrée sur l'adaptation des technologies existantes pour l'IoT [htt]. Il définit l'IoT ainsi :

Définition 2.2.5 (IoT selon OASIS)
Un système à travers lequel Internet est connecté au monde physique via des capteurs [Com].

D'après le groupe de travail T2TRG pour Thing-to-Thing Research Group de l'Internet Research Task force crée en 2015. L'Internet des Objets assure la communication entre les objets en utilisant Internet pour offrir de meilleurs services. Ce groupe de travail a définit l'IoT comme :

Définition 2.2.6 (IoT selon T2TRG)
Un réseau où les appareils à faibles ressources communiquent entre eux et avec Internet pour assurer des innovations [19a].

L'Institue of Electrical and Electronics Engineers (IEEE), à travers le projet et le groupe de travail P2413, a étudié l'Internet des Objets [19b]. Le projet P2413 comporte des membres industriels comme CISCO Systems et des organismes comme l'Industrial Internet Consortium (IIC). L'IIC est un groupe formé de plusieurs industriels connus dans le monde des technologies de l'information comme Huawei, Intel, etc. L'IEEE décrit l'Internet des Objets ainsi :

Définition 2.2.7 (IoT selon IEEE)
L'Internet des Objets est l'inter-connexion de différents systèmes possédant des standards bien adaptés comme par exemple différentes technologies de communication permettant de respecter les besoins de chacun de ces systèmes.

Il existe depuis l'émergence de l'Internet des Objets plusieurs projets et consortiums européens et internationaux traitant cette thématique. Nous allons maintenant présenter d'autre définitions de l'Internet des Objets émanant de plusieurs projets de recherche internationaux.

Le projet Internet of Things Initiative (IoT-I) est un projet européen financé par les programme FP7-ICT [Woe10]. Le projet a pour but la création d'un environnement économiquement durable et socialement acceptable en Europe, pour les technologies de l'Internet des Objets. L'IoT-I a défini l'Internet des Objets ainsi :

Définition 2.2.8 (IoT selon IoT-I)
L'IoT est une part intégrée de l'Internet du futur formée par une infrastructure de réseau globale et dynamique possédant des capacités d'auto-configuration basées sur des standards et des protocoles de communications interopérables.

Comme les autres organismes précédemment cités, l'IoT-I a spécifié que les objets doivent posséder des identités, des attributs et doivent utiliser des interfaces intelligentes afin d'être intégrés de manière transparente. Cette définition de l'Internet des Objets a été utilisé aussi par le consortium des projets européens sur l'Internet des Objets par exemple le Cluster of

European Research Projects on the Internet of Things (CERTP-IoT) [CAS09].

La CASAGRAS (Coordination and Support Action for Global RFID) est un projet européen financé par le programme FP7-ICT (2008 - 2009) [I S09]. Ce projet s'est concentré sur des études fondamentales pour les technologies sous-jacentes de l'IoT et leurs adaptations. Selon ce projet l'IoT est défini ainsi :

Définition 2.2.9 (IoT selon CASAGRAS [Cra09])
L'IoT est une infrastructure de réseau globale, reliant les objets physiques et virtuels pour la capture de données à travers des capacités de communication. Cette infrastructure intègre les développements existants de l'Internet et du réseau.

De plus, elle offre une identification unique aux objets et une capacité de communication qui sont la base pour le développement de services et d'applications coopératives indépendants. Ceux-ci seront caractérisés par la capture de données d'une manière autonome, le transfert des données, la connectivité réseau et l'interopérabilité.

2.3 Protocoles de communication

Il y a deux grandes familles qui existent dans les les protocoles utilisés dans l'Internet des Objets ceux qui sont gérés par les opérateurs de réseaux téléphoniques qualifiés de *cellulaires* et les autres *non cellulaires*. Nous présentons d'abord les non cellulaires puis les cellulaires.

2.3.1 Protocole non cellulaire

Des technologies sans fil non cellulaires peuvent être utilisées pour la communication entre les objets dans l'IoT. Ces technologies opèrent à différents niveaux du modèle OSI (Open Systems Interconnection). Nous présentons dans cette section une liste non exhaustive des différentes technologies sans fil non cellulaires utilisées dans l'environnement IoT. Ici nous considérons que les protocoles non cellulaires sont ceux qui n'utilisent pas une architecture d'un opérateur, mais qui ont une logique de communication point à point où il n'y a pas de hiérarchisation de nœuds.

- LoRa (Long Range) est une technologie sans fil offrant une longue durée de vie des batteries et une bonne couverture pouvant atteindre les 15 Km. Cette technologie est utilisée dans l'IoT pour des applications à débit réduit entre 0.3 Kbit/s et 50 Kbit/s. La technologie LoRa se limite à la spécification de la couche 1 du modèle OSI tandis que LoRaWAN étend la spécification à la couche 2 [All15].
- IEEE 802.15.4 est un standard de communication adapté aux réseaux à faible débit et à faible portée. Ce standard définit les caractéristiques des couches 1 et 2 du modèle OSI et peut être utilisé comme base par d'autres technologies de l'IoT [LN16] comme par exemple [Cra17] :
  - ZigBee
  - 6LowPAN

  Les fréquences, puissances, modulations et techniques de contrôle d'accès au medium partagé avec un débit pouvant atteindre au maximum 250 Kbit/s sont définis par la norme IEEE.
- Z-Wave [ITU15] qui est un standard de communication permettant une faible consommation énergétique. Cette technologie assure des débits entre 9.6 Kbit/s et 40 Kbit/s pour une couverture de l'ordre de 30 mètres.

— Bluetooth Low Energy [And14] variante du standard Bluetooth adaptée pour les capteurs ayant peu de ressources. Cette technologie assure une couverture allant jusqu'à 60 m avec un débit pouvant atteindre les 305 Kbit/s.

2.3.2 Protocole cellulaire

Il existe des technologies et variantes de technologies sans fil cellulaires qui ont été proposées pour une utilisation dans l'Internet des Objets. Ainsi, la technologie LTE (Long Term Evolution) est une évolution des normes de téléphonie mobile qui peut être adaptée et utilisée dans l'IoT en assurant une bonne couverture étendue arrivant à 5 Km pour un grand nombre d'objets et avec un débit théorique entre 50 et 100 Mbit/s [Ame15] [101]. Il existe différentes variantes de LTE tel que le LTE-M qui permet d'économiser de l'énergie, d'optimiser les interférences et d'être adapté au caractéristiques des appareils au sein de l'Internet des Objets [Ame15] . Il existe également le LTE-Advanced, connu par le grand public comme la 4G, celle-ci permet d'avoir des réseaux plus rapides et plus efficaces avec de meilleures bandes passantes et une meilleure intensité du signal dans un environnement urbain. La 4G assure une fiabilité accrue en termes de perte de paquets et une meilleure sécurité pour une utilisation dans un environnement IoT. Il existe également la Narrowband IoT (NB-IoT) qui est une technologie radio avec une consommation énergétique faible pour assurer une longue durée de vie afin de s'adapter aux différentes problématique des objets IoT la contrainte est celle d'un faible débit, de l'ordre de 200 Kbit/s [Ass].

La technologie LoRaWAN qui définit le protocole de communication et l'architecture réseau correspondante avec le LoRa organise un réseaux via l'utilisation du LoRa avec une couche MAC supplémentaire. Celle-ci s'organise comme les réseaux mobiles autour d'une antenne d'un opérateur.

La 5G est la 5ème génération de réseaux mobiles, une évolution significative des réseaux 4G LTE Advanced actuels. La 5G est conçue pour répondre à la forte croissance des flux de données échangés sur les réseaux. Il existe trois grandes catégories de cas d'utilisation pour la 5G.

1. Les communications appareil à appareil (Device to Device, D2D) qui consistent à connecter des milliards de périphériques sans intervention humaine.
2. Communications ultra-fiables à faible temps de latence.
3. Des débits de transmission de données considérablement plus importants.

Avec la 5G le débit de liaison descendante peut atteindre 20 Gbit/s et celui de la liaison montante 10 Gbit/s avec une latence entre 10 et 20 ms [219]. La 5G est au moment de l'écriture de ce manuscrit en plein déploiement à l'usage des particuliers.

2.4 Domaine d'applications

Dans cette section les domaines comme la domotique, l'environnement intelligent, le transport, la logistique, les villes intelligentes et la E-santé vont être présentés.

2.4.1 Domotique

La catégorie domotique regroupe les appareils de contrôle à distance, comme par exemple allumer et éteindre des appareils à distance pour éviter des accidents et économiser de l'énergie. Les compteurs intelligents pour contrôler l'énergie électrique ou l'eau afin de surveiller la

consommation d'énergie pour obtenir des conseils sur la façon d'économiser les ressources. La domotique au sens large c'est aussi de la surveillance d'un lieu en intérieur et donc par exemple d'un musée ou sont exposés des œuvres sensibles à l'humidité et donc des capteurs d'humidité et de température peuvent être déployés dans ce cadre d'utilisation. Il y a également des systèmes de détection d'intrusion : ouverture de portes, de fenêtres et des violations dans le but d'empêcher les intrusions.

De plus, des assistants intelligents tels que Google Home [Goo], Alexa [Lam18] [Ama], HomePod avec Siri [Appb] sont munis de haut parleurs et de microphones pour gérer des commandes vocales avec les utilisateurs environnants.

D'autres appareils comme des éclairages intelligents qui sont dans la domotique ; le domaine le plus prisé par les particuliers et dans la communauté de recherche [Mar11][YH11]. Particulièrement, Apple a conçu un framework appelé HomeKit [Appa] qui permet aux utilisateurs de configurer, communiquer avec et contrôler des appareils domestiques intelligents. L'une des marques la plus connue est celle de Philipps Hue [Phi18] qui sont des ampoules pouvant être contrôlées à partir d'un appareil mobile. L'ampoule réagit au contexte et peut changer de couleur et de luminosité en fonction des préférences de l'utilisateur, de la saison, du jour, de l'heure et de l'activité de l'utilisateur.

Par ailleurs dans le domaine de la domotique il existe des thermostats qui mémorisent les habitudes des utilisateurs et leurs températures préférées. Ils baissent ou augmentent le chauffage en fonction de leur présence à leur domicile avec un calcul du temps nécessaire pour chauffer le logement. Le tout est contrôlable à distance [Nes] ou, par exemple, basé sur une solution comme Tado [Tad18]. Des serrures électroniques comme les Lockitron [Loc] ou les Nuki SmartLock [Nuk] peuvent être ouvertes et fermées à distance pour gérer un domicile à distance ou à proximité.

2.4.2 Environnement intelligent

Cette catégorie regroupe la détection d'événements ou d'anticipations d'événements lié à des catastrophes naturelles comme les tremblements de terre, glissement de terrains, avalanches, détections d'incendies dans un milieu forestier. Comme par exemple [Rob18] Insightrobotics détecte les incendies de forêt grâce aux informations collectées par des caméras et différents types de capteurs de température, vent, etc. Ou dans des cas plus complexes comme la détection de sources radioactives ou nucléaires [Bov+18].

Les mesures de la pollution marine sont comprises également dans cette catégorie via par exemple : la surveillance en temps réel des fuites et des déchets en mer, la détection des fuites chimiques en rivière, la surveillance des fluctuations du niveau d'eau des rivières, les barrages et retenues, la télémesure des piscines, la détection de la présence de liquide à l'extérieur des réservoirs, la fluctuation de pression le long des canalisations et la surveillance de l'eau potable : surveillance de la qualité de l'eau du robinet.

Le réseau de capteurs flottants recueille des données en temps réel comme le débit, la vitesse, la température, niveau de pollution sur les voies navigables au moyen d'une série de bouées connectées [TRB12].

Intelligentriver est un système de surveillance qui permet l'analyse, le suivi et la gestion des ressources en eau en temps réel. Une solution similaire a été développée par l'ESRP (European Shoal Research Project) [Whi+10] [Sho18].

2.4.3 Transport et logistique

Cette catégorie regroupe tout ce qui encadre le contrôle des marchandises sensibles comme les bijoux, les médicaments ou les marchandises dangereuses, la qualité des conditions d'expédition : surveillance, à des fins d'assurance, des vibrations, des coups, des ouvertures de conteneurs ou de leur entretien.

La société HiKoB promeut une gestion d'informations en temps réel sur les conditions de circulation des différents services de transport de marchandises et de logistique. La collecte des mesures se fait en temps réel comme par exemple les températures, l'humidité ou les gradients de température sur des capteurs déployés [Hik18].

Une solution par Alltrafficsolutions consiste à collecter des données sur le trafic routier au moyen de capteurs et fournit aux conducteurs des informations actualisées. Il prend en compte les modifications des panneaux de signalisation numériques, à message variables et de limitation de vitesse [Sol18b].

Cantaloupe Systems permet à l'utilisateur de suivre à distance les stocks dans les distributeurs automatiques [Sys17].

Senseaware est une solution développée pour prendre en charge le suivi des expéditions en temps réel. Les informations comme la température, la localisation, l'humidité relative, la lumière et la pression sont collectées [Sen18].

Des travaux liés à la gestion de chaînes d'approvisionnement [DT08] intégrant l'architecture orientée service (Service Oriented Architecture : SOA) ont été réalisés. Des applications basées sur des capteurs positionnés sur la chaîne d'approvisionnement permettent un contrôle plus efficace de la qualité des articles périssables.

Il y a aussi divers travaux sur les voitures dans le futur dans l'industrie automobile [Kir15] ainsi que des services de Smart Transport basé sur l'IoT [ZYZ11].

Tous les types de systèmes de transport peuvent bénéficier des avantages qu'offre l'IoT comme être plus performants en améliorant la sécurité, l'efficacité, la maintenance des véhicules [Ent18].

Il existe un projet américain qui utilise un système de surveillance pour informer les conducteurs des places de stationnement disponibles et appliquer des tarifs appropriés en fonction de la demande en temps réel [KA].

Au royaume-uni il y a un programme de Smart Motorways de Highways England qui permet de réduire les embouteillages en surveillant en temps réel la charge de trafic selon les caractéristiques des routes. UKCITE, un autre projet britannique financé par le Centre des véhicules connectés et autonomes a pour objectif de fournir un environnement de test pour les véhicules de conduite connectés et autonomes [gov12] [gov].

Il existe des travaux concernant le tracking de véhicule appliqué aux bus pour prédire l'heure d'arrivée en temps réel[JJK17].

2.4.4 Ville intelligente

En ce qui concerne les villes intelligentes, les données surveillées sont par exemple : le niveau des champs électromagnétiques, la santé des infrastructures, la gestion des déchets, la détection des smartphones, les routes intelligentes, le stationnement intelligent, l'éclairage intelligent, la gestion des embouteillages, la surveillance de la nuisance sonore urbaine.

BigBelly Solar est une solution de gestion intelligente des déchets qui propose un panier de capteurs intégrés pouvant analyser le contexte en temps réel et alerter les services dédiés lorsqu'il est plein et doit être vidé afin de planifier une récupération efficace [Sol18a]. L'application LiveHood analyse comment les habitants d'une ville utilisent le paysage urbain et les espaces de vie communes [Liv18]. L'application StreetLine est une technologie de gestion du stationnement qui a été développée pour les villes. Les informations sont appelées par des capteurs (magnétomètres) intégrés dans les places de parking. L'application propose des services de localisation et de cartographie pour guider les conducteurs vers les emplacements appropriés en temps réel [Gil+13]. Il existe des solutions d'edge computing intelligentes pour des solutions IoT pour de la gestion d'énergie dans les smart cities [Liu+19].

Selon IHS Markit, la demande d'appareils intelligents pour la maison a considérablement augmenté avec plus de 161 millions d'unités vendues entre 2010 et 2016. Plus de la moitié de ces appareils ont été vendus en 2016. Cette augmentation comprenait l'achat de systèmes intelligents de gestion de l'énergie tels que des thermostats, des solutions de sécurité telles que des serrures intelligentes et des assistants personnels comme précédemment cité [IHS16] [Map17].

Divers comités de normalisation se concentrent sur ce domaine d'application de l'Internet des Objets à travers des groupes de travail ayant un intérêt pour la normalisation des technologies utilisées, par exemple l'utilisation commerciale et résidentielle [ISO] [ISO17] [Nok19].

Dans la communauté scientifique, il y a également des travaux autour des villes intelligentes comme dans [Gho+12] les auteurs décrivent une application d'alerte et de surveillance des conditions routières utilisant les capteurs d'un smartphone embarqué et connecté à une plateforme IoT. Mais également dans [Fos+11] où les auteurs décrivent et présentent la conception, la mise en œuvre d'une application machine-to-machine dans le domaine de la gestion du trafic routier, qui intègre une large infrastructure de services basés sur le système multimédia IP (IMS) pour des raisons d'efficacité.

Il existe des solutions autour de la finance au sein d'une ville concernant l'IoT comme une méthode de paiement rapide et sécurisée pour les plateformes edge-IoT utilisant le blockchain [HJL18].

Dans la ville intelligente le contrôle des débits d'eau : mesure de la pression de l'eau dans les systèmes de transport d'eau, les niveaux des réservoirs, de gaz dans les réservoirs, le smart grid avec le suivi et contrôle des consommations d'énergie et de ressources. Il y a des travaux autour des smartgrids avec des algorithmes pour la gestion des réseaux électriques intelligents [ME18] et des travaux autour d'un réseau de voisinage défini par logiciel pour les applications de réseaux intelligents [Naf+18].

L'industriel Echelon qui a développé une solution d'éclairage public intelligent qui transforme l'éclairage public en un réseau intelligent, économe en énergie et télécommandé qui programme l'allumage et l'extinction des lumières et la gradation des lumières individuelles ou des groupes de lumières afin qu'une ville puisse fournir intelligemment le bon niveau d'éclairage en analysant le contexte telles que la météo, la saison ou les conditions climatiques [Ech18].

Le projet SmartSantander [1] propose une installation de recherche expérimentale unique au monde à l'échelle de la ville, à l'appui d'applications et de services typiques d'une ville intelligente. Le projet prévoit le déploiement de 20 000 capteurs à Belgrade, Guildford, Lübeck et Santander (12 000), en exploitant une grande variété de technologies.

1. https ://www.smartsantander.eu/

2.4.5 E-santé

Cette catégorie comprend :

— le contrôle sur des expositions fortes aux rayons Ultra Violet (UV)
— la prise en charge des sportifs par le suivi des paramètres vitaux dans les centres et terrains de haute performance
— le suivi des personnes seules ou isolées : aide aux personnes âgées ou handicapées autonomes, surveillance de l'état des patients dans les hôpitaux
— le contrôle de l'état des congélateurs qui stockent les vaccins, les médicaments et les produits organiques

En général, les systèmes collectent les données vitales des patients via un réseau de capteurs connectés aux dispositifs médicaux et garantissent le partage de données médicales comme l'Electronic Healthcare Records (EHR) [Gac+12] [Kuo11] [Swa+19]. Il y a des propositions pour améliorer les systèmes d'informations des hôpitaux avec des solutions basées sur l'Internet des Objets comme dans [DM12] une plate-forme basée sur l'IoT pour la gestion des capteurs de santé [Fre+18].

L'assistant sportif personnel BioHarness est doté d'une ceinture pectorale qui enregistre la fréquence cardiaque, le niveau de stress, la vitesse, la distance, les calories brulées et le niveau d'activité. Il aide à recommander des exercices dans des fréquences cardiaque spécifiques [BIO18].

L'assistant médical ElectricFoxy est un anneau qui surveille et suit la fréquence cardiaque de l'utilisateur.

Amélioration cardiovasculaire LeChal fournit une assistance aux personnes handicapées sur la base d'une paire de chaussures qui fournissent intuitivement un retour de vibration pour suggérer la bonne direction et identifier les obstacles [Lec18].

Le babyphone Mimo surveille la température de la peau, la position du corps, la respiration, les bruits et le niveau d'activité au moyen de capteurs de bruit, de pression et de température et informe les parents de toute situation anormale [Mim18].

Il existe des applications qui comprennent la surveillance de la pression artérielle et du diabète, la surveillance de la température corporelle et la surveillance de la saturation en oxygène[R+13] [Map17] comme par exemple avec l'industriel Ericsson et ses partenaires proposent des prototypes portables permettant de surveiller les patients atteints de diabètes avec des durées de vie de batterie importantes[Eri18]

Il y a un domaine dans la E-santé et cybersanté tel que les Body Area Networt (BAN). Les BAN, également connus sous le nom de BSD (Body Sensor Network) ou WBAN (Wireless Body Area Network), sont de nouveaux types de réseaux de capteurs extrêmement faibles, petits et légers à usage humain définit BAN comme une norme de communication optimisée pour les appareils à faible consommation d'énergie fonctionnant dans ou autour du corps humain pour une variété d'applications, l'électronique grand public, le divertissement et plus encore [BA12] [RB12] [Cha+17].

2.5 RGPD : Règlement général sur la protection des données

Comme vu dans la partie précédente la E-santé est soumise à une réglementation, la RGPD, entrée en application le 25 mai 2018, les données de santé concernées sont « les données relatives

à la santé physique ou mentale, passée, présente ou future, d'une personne physique (y compris la prestation de services de soins de santé) qui révèlent des informations sur l'état de santé de cette personne [CNI18].

Le RGPD définit le terme « Privacy by Design » : sans le consentement clair et expresse de la personne concernée, qui est bien informée de la finalité, l'utilisation ou la commercialisation des données de santé d'une personne identifiée ou identifiable est interdite.

L'environnement IoT dans le domaine de la santé est très régulé en Europe, ainsi les applications agissant sur des domaines sensibles ou personnels doivent garantir le respect des règles imposées par le RGPD.

Plus généralement, le RGPD encadre le traitement des données personnelles, au niveau de la transparence et la confiance qu'un usager peut accorder à un service. La notion de données personnelles est à comprendre de façon très large. Une donnée personnelle est toute information se rapportant à une personne physique identifiée ou identifiable.

2.6 Conclusion

L'Internet des Objets est un concept qui intervient dans tout les domaines, avec des technologies variées. Ce paradigme a diverses contraintes, problématiques qui sont liées à la fois à la législation mais surtout aux protocoles utilisés qu'ils soient gérés par des opérateurs ou pas. Il y a plusieurs domaines pour lesquels l'Internet des Objets permet un meilleur contrôle et une meilleure surveillance. Tout ceci a été présenté dans ce chapitre afin de comprendre les enjeux autour de l'Internet des Objets. Dans le prochain chapitre nous allons aborder le cadre de travail de cette thèse, en expliquant nos choix sur les protocoles utilisés, et l'architecture associée.

Chapitre 3
Cadre de travail

Résumé : *Nous présentons le cadre de travail autour de nos travaux de recherche, la technologie LoRa et l'architecture LoRaWAN. Nous présentons également les différentes manières que nous avons envisagé pour rapatrier les données d'un nœud qui ne peut joindre le réseau LoRaWAN mais dont nous avons tout de même besoin. Ainsi pour les différentes expérimentations et implémentations nous avons utilisé différents capteurs que nous présentons dans ce chapitre.*

Sommaire

3.1 Introduction

Durant nos travaux nous nous sommes concentrés sur le choix du LoRa et du LoRaWAN qui sont des protocoles non cellulaires comme présenté dans le chapitre précédent. Dans ce chapitre nous allons donc entrer dans le détail au niveau du LoRaWAN par rapport à son architecture, les différentes classes d'objets considérées et la sécurité.

3.2 LoRa/LoRaWAN

Le protocole de communication LoRa, développé par la société Semtech, vise à mettre en place un réseau étendu de faible puissance (Low-Power Wide-Area Network (LPWAN)) basé sur une technologie sans fil à longue portée et à faible débit. Elle s'apparente à un protocole cellulaire (systèmes mobiles 2G/3G/4G) mais optimisée pour IoT/M2M. La technologie LoRa ne nécessite pas de licence d'utilisation du spectre car elle utilise des bandes de fréquences libres par exemple, 863-870 MHz en Europe, 902-928 MHz aux États-Unis, 779-787 MHz en Chine. Un end-device LoRa, doté d'une alimentation électrique autonome, est censé pouvoir communiquer sur plusieurs kilomètres dans une zone urbaine et avoir une durée de vie de huit à dix ans.

LoRaWAN est un protocole qui vise à sécuriser la couche de contrôle d'accès au support d'un réseau LoRa. Il est conçu par la LoRa Alliance, une association qui regroupe plus de 500 membres (opérateurs de télécommunications, fabricants de semi-conducteurs, sociétés de sécurité numérique, fabricants de matériel, fournisseurs de réseaux, etc.). Des réseaux LoRaWAN publics et privés sont déployés dans plus de 100 pays dans le monde par des opérateurs de télécommunications (SK Telecom, FastNet, ZTE, KPN, Bouygues Télécom, Proximus, etc.), des fournisseurs privés et des initiatives privées (par exemple, The Things Network). Plusieurs réseaux nationaux sont déjà déployés en Europe (France, Pays-Bas), en Asie (Corée du Sud), en Afrique (Afrique du Sud) et en Océanie (Nouvelle-Zélande). Des essais sont lancés au Japon, aux États-Unis (en commençant par une centaine de villes), en Chine (la couverture prévue s'étend à 100 millions de foyers et 300 millions de personnes), en Inde (la première phase du réseau vise à couvrir 400 millions de personnes dans tout le pays). Les cas d'utilisation auxquels LoRaWAN vise à répondre comprennent les compteurs intelligents (électricité, eau), le suivi (conteneurs maritimes, biens de valeur), l'agriculture (irrigation), les réseaux intelligents (gestion des pannes), l'industrie (capteurs de tremblements de terre, avalanches), les villes intelligentes, les vêtements, la santé (vêtements médicaux), la maison connectée (systèmes de sécurité) et la télématique des véhicules (informations de suivi, état des véhicules) [com] [IoT19] [Fea16] [mar16] [Bri16] [Biz15][Sma16] [OA16] [KH16] [Sor17] [com18].

Dans le cadre de nos recherches nous nous intéressons uniquement aux versions 1.0.[1-4] de la spécification LoRaWAN.

3.2.1 Architecture

La Figure 3.1 représente une architecture classique d'un réseau LoRaWAN. Typiquement un réseau LoRaWAN est en topologie "star of star" étoile en étoile, le centre de cette topologie est le serveur de réseau qui assure la gestion du débit adaptatif, de la sécurité des données ou encore de la redondance des données. Celui-ci est entouré d'un côté par les gateways connectées en ethernet/4G au serveur de réseau, auquel les end-devices seront connectés, eux en LoRa ce qui assure une sécurité de bout en bout. D'autre part nous avons les serveurs d'applications liés par ethernet, ces mêmes serveurs d'applications liés elles-mêmes à une interface web (échange HTTP, MQTT). Une des particularités d'un réseau LoRaWAN, est qu'un équipement ne

communique pas exclusivement à travers un concentrateur. Tous les concentrateurs couvrant l'équipement peuvent recevoir les données transmises par ce dernier. Cela facilite grandement la communication avec les équipements en mobilité en dispensant le réseau de mécanismes de hand-over (passage d'un concentrateur à un autre) qui auraient pour effet de complexifier sa gestion et très probablement de réduire ses performances. Par contre, lorsque le serveur envoie un message à destination d'un équipement, c'est par le biais d'un seul concentrateur. C'est le cas en particulier des messages requérant un acquittement par le serveur. Ce qui assure l'échange entre le serveur de réseau et les serveurs d'applications est l'élément appelé AppSKey. La sécurité après ces serveurs n'est plus gérée par la spécification LoRaWAN [Dwo05][LI06][Sor17].

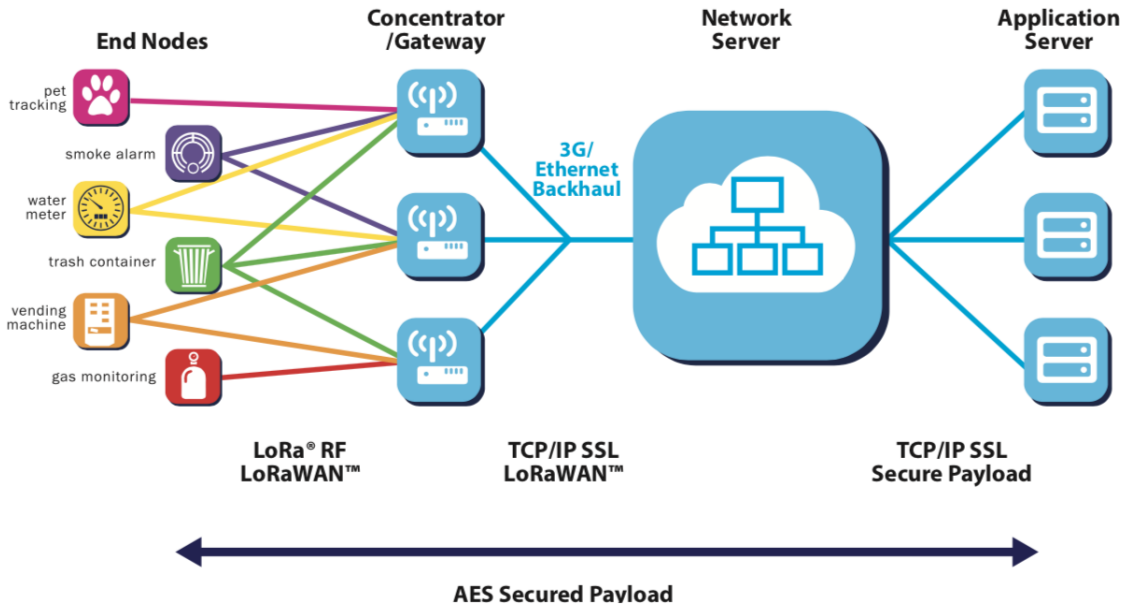


FIGURE 3.1 – Architecture d'un réseaux LoRaWANSource : [All]

3.2.2 Les différentes couches d'un réseaux LoRaWAN

En définissant le protocole réseau (LoRa MAC), pour une communication d'équipements LoRa à travers un réseau, le protocole LoRaWAN assure une communication bidirectionnelle et définit trois classes d'équipements différents comme le montre la Figure 3.2 :

— Classe A : All devices
— Classe B : Beacon
— Classe C : Continuously listening

Nous definissons celles-ci en détail dans la partie consacrée à l'explication des différences entre celles-ci.

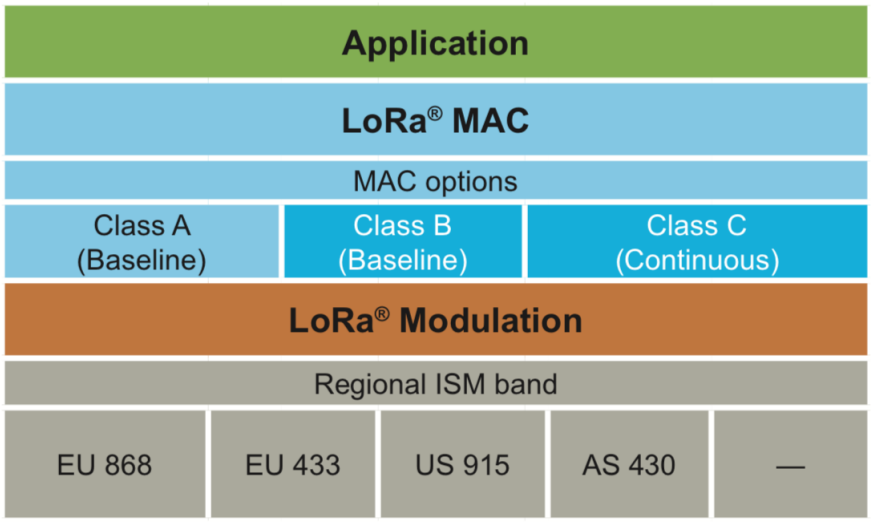


FIGURE 3.2 – Les différentes couches d'un réseau LoRaWANSource : [All]

3.2.3 Sécurité du LoRaWAN

Une question importante est aussi celle de la sécurité à travers le réseau, on parle de capteurs, voire d'un ensemble d'informations qui vont transiter. La sécurité est assurée de bout en bout par un chiffrement AES-128 bits [ST02]. Les clefs de chiffrement sont aux nombre de deux :

- Network Session Key (NwkSKey), assure l'authenticité des équipements sur le réseau.
- Application Session Key (AppSKey), la sécurité et la confidentialité des données transmises à travers le réseau.

En d'autre termes, la clef réseau permet à l'opérateur de sécuriser son réseau alors que la clef applicative permet au fournisseur de l'application de sécuriser les données qui transitent à travers le réseau.

Les données utiles qui sont transmises sont tout d'abord chiffrées via la AppSKey. Un en-tête, contenant entre autres l'adresse de l'équipement, est ensuite ajouté aux données chiffrées. À partir de cela, le MIC (Message Integrity Code) est calculé via NwkSKey. Le MIC permet au réseau de vérifier l'intégrité des données et de l'équipement sur le réseau. Enfin, le MIC est ajouté au message contenant l'en-tête et les données chiffrées avant transmission.

À réception du message par le serveur de gestion du réseau, ce dernier pourra vérifier l'intégrité des données grâce au MIC tout en préservant la confidentialité des données (chiffrées par l'AppSKey).

3.2.4 Classe A : Initiative complète du device

Cette classe A présente la consommation énergétique la plus faible. Lorsque l'équipement a des données à envoyer, il le fait sans contrôle puis il ouvre 2 fenêtres d'écoute successives pour des éventuels messages provenant du serveur, les durées recommandées sont de 1 puis 2 secondes. Ces 2 fenêtres sont les seules durant lesquelles le serveur peut envoyer à l'équipement les données qu'il a précédemment stockées à son attention.

Il n'est pas mis en évidence dans la Figure 3.3 les sauts de fréquences qui s'opèrent lors du placement des fenêtres de réception. Chaque fenêtre est sur une fréquence différente. Celles-ci sont décrites et doivent respecter la spécification.

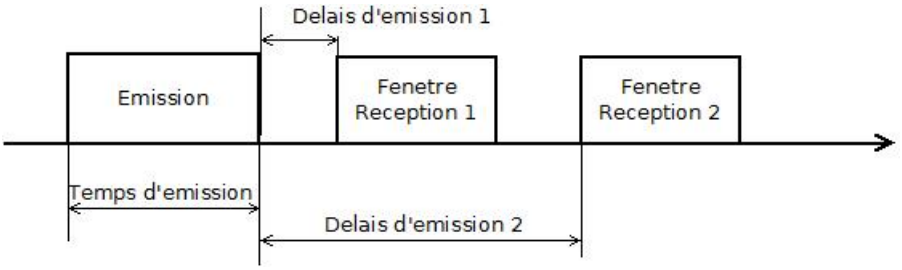


FIGURE 3.3 – Fenêtre de réception et d'émission d'un objet en classe A

3.2.5 Classe B : Rendez-vous

Cette partie va décrire les principes d'un objet respectant la classe B comme spécifié par la spécification LoRaWAN. De plus nous mettons en avant les différences pour passer de la classe A à la classe B comme il est prévu dans la spécification.

3.2.5.1 Introduction

Cette section décrit la couche LoRaWAN Classe B optimisée pour les appareils alimentés par batterie qui peuvent être mobiles ou montés à un emplacement fixe. Les terminaux doivent mettre en œuvre une opération de classe B lorsqu'il est nécessaire d'ouvrir des fenêtres de réception à des intervalles de temps fixes afin d'activer les messages downlink. Un objet en classe B ajoute une fenêtre de réception synchronisée sur le device. L'une des limitations d'un objet en classe A est la méthode ALOHA d'envoi de données à partir du périphérique. Il ne permet pas de connaître le moment de la réaction lorsque l'application client ou le serveur souhaite s'adresser aux périphériques. L'objectif de la classe B est d'avoir un device disponible pour la réception à une heure prévisible, en plus des fenêtres de réception qui suivent la transmission aléatoire d'un message uplink du périphérique de classe A. La classe B est obtenue en envoyant à la passerelle un beacon sur une base régulière pour synchroniser tous les appareils dans le réseau de sorte que l'appareil puisse ouvrir une courte fenêtre de réception supplémentaire (appelée "ping slot") à un moment prévisible pendant un intervalle de temps périodique.

3.2.5.2 Principe de la synchronisation du réseau initialisé en downlink

Pour qu'un réseau puisse prendre en charge les terminaux de classe B, toutes les passerelles doivent diffuser de manière synchrone un beacon fournissant une référence de synchronisation aux terminaux. Sur la base de cette référence de temporisation, les devices peuvent périodiquement ouvrir des fenêtres de réception, appelées après des intervalles, qui peuvent être utilisées par l'infrastructure de réseau pour initier une communication downlink. Un message downlink

utilisant l'un de ces emplacements est appelée un slot-ping. La passerelle choisie pour initier cette communication downlink est sélectionnée par le serveur de réseau sur la base des indicateurs de qualité de signal de la dernière liaison uplink de l'appareil. Pour cette raison, si un périphérique se déplace et détecte une modification de l'identité annoncée dans le beacon reçu, il doit envoyer en uplink afin que le serveur puisse mettre à jour la base de données du chemin de routage. Tous les périphériques démarrent et rejoignent le réseau en tant que périphériques de la classe A. L'application peut alors décider de passer à la classe B.

3.2.5.3 Acquisition du beacon et tracking

Avant de passer de la classe A à la classe B, le device doit d'abord recevoir un beacon pour aligner sa référence d'horloge interne avec le réseau. Une fois en classe B, le périphérique final doit périodiquement rechercher et recevoir un beacon pour annuler toute dérive de son d'horloge interne, par rapport à la synchronisation du réseau. Un périphérique de classe B peut être temporairement incapable de recevoir des beacons (hors de portée des passerelles réseau, présence d'interférences...). Par conséquent, le terminal doit élargir progressivement ses fenêtres de réception de beacon et de ping pour prendre en compte une éventuelle dérive de son horloge interne.

3.2.5.4 Synchronisation d'un emplacement en downlink

Définitions Pour fonctionner correctement dans la classe B, l'appareil doit ouvrir des créneaux de réception à des instants précis par rapport aux beacon de l'infrastructure. L'intervalle entre le début de deux beacons successifs est appelée "beacon period". La transmission de la fenêtre du beacon est alignée avec le début de l'intervalle BEACON_RESERVED. Chaque beacon est précédé d'un intervalle de temps de garde où aucun emplacement de ping ne peut être placé. Ceci afin d'assurer qu'un downlink initié pendant une fenêtre de réception juste avant l'heure de garde aura toujours le temps de se terminer sans entrer en collision avec la transmission du beacon. L'intervalle de temps utilisable pour l'intervalle de ping s'étend donc de la fin de l'intervalle de temps réservé au beacon jusqu'au début de l'intervalle du beacon suivant.

Emplacement aléatoire Pour éviter les collisions systématiques ou les problèmes de congestion, l'index des créneaux horaires est aléatoire et modifié à chaque période de beacon. Les paramètres suivants sont utilisés :

DevAddr	Device 32 bit network unicast or multicast address
pingNb	Number of ping slots per beacon period. This must be a power of 2 integer: *pingNb* = 2^k where 1 <= k <=7
pingPeriod	Period of the device receiver wake-up expressed in number of slots: *pingPeriod* = 2^{12} / *pingNb*
pingOffset	Randomized offset computed at each beacon period start. Values can range from 0 to (pingPeriod-1)
beaconTime	The time carried in the field **BCNPayload**.Time of the immediately preceding beacon frame
slotLen	Length of a unit ping slot = 30 ms

FIGURE 3.4 – Option Random class B

À chaque période du beacon, le terminal et le serveur calculent un nouveau décalage pseudo-aléatoire pour aligner les créneaux de réception. Ce qui donne le comportement suivant en terme de fenêtre de réception :

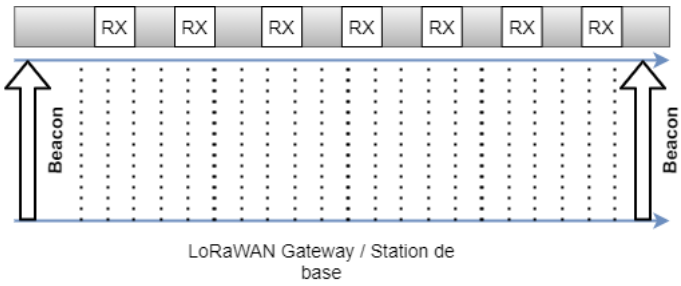


FIGURE 3.5 – Fenêtre de réception et d'émission d'un objet en classe B

3.2.6 Classe C : Continiously listening

Les devices implémentant l'option de classe C sont utilisés pour des applications qui ont une énergie plus que suffisante et n'ont donc pas besoin de minimiser le temps de réception. Les terminaux de classe C ne peuvent pas implémenter l'option de classe B. Le périphérique de classe C écoutera les paramètres des fenêtres RX2 aussi souvent que possible. Pour ce faire, il ouvrira une petite fenêtre sur les paramètres RX2 entre la fin de la transmission d'un uplink et le début de la fenêtre de réception RX1 et basculera sur les paramètres de réception RX2 dès que la fenêtre de réception RX1 sera fermée ; la fenêtre de réception RX2 reste ouverte jusqu'à ce que l'appareil envoie un autre message.

Temps de la deuxième fenêtre de réception pour la classe C Les périphériques de classe C implémentent les mêmes deux fenêtres de réception que les périphériques de classe A, mais ils ne ferment pas la fenêtre RX2 tant qu'ils n'ont pas besoin d'envoyer à nouveau. Par conséquent, ils peuvent recevoir un downlink dans la fenêtre RX2 à tout moment.

Multicast downlink De même que pour la classe B, les périphériques de classe C peuvent recevoir des trames downlink multicast. L'adresse de multidiffusion et la clef de session de réseau associée et la clef de session d'application doivent provenir de la couche d'application. Les mêmes limitations s'appliquent aux trames downlink multicast de classe C :

3.2.7 Activation d'un end-device

En LoRaWAN, les trois éléments indispensables pour la communication sont le DevAddr pour l'indentification du Device, ainsi que deux clés : le NwkSKey pour l'authentification et l'AppSKey pour le chiffrement. Voici la liste des éléments qui concerne l'activation d'un device au sein d'un réseau LoRaWAN.

- Le DevAddr est un mot de 32 bits qui identifie le end-devices dans le réseau courant.
- Le DevEUI est l'identifiant de l'appareil au format IEEE EUI64.
- La clef de l'application (AppKey) est une clef AES-128 bits.
- L'identification de l'application (AppEUI) identifie le provider de l'application.
- La clef de session réseau (NwkSkey) est spécifique au device utilisé par le serveur réseau et le device pour calculer l'intégrité de tous les messages.
- La clef de session d'application (AppSKey) est spécifique au device utilisé par le serveur réseaux et le device pour chiffrer/déchiffrer le champ Payload de tous les messages. Mais aussi pour calculer l'intégrité de tous les messages.

Deux méthodes sont possibles pour fournir ces informations à la fois au nœud LoRa et au serveur :

— Activation avec Over the Air (OTA)
— Activation par Personnalisation (ABP)

3.2.7.1 Activation avec Over the Air (OTA)

Les devices utilisant l'OTA doivent forcément utiliser la procédure "join" pour participer aux échanges de données avec le serveur de réseaux. Un device doit forcement utiliser une nouvelle fois la fonction "join" à chaque fois qu'il a perdu les informations contextuelles liées à la session. À l'issue de cette procédure le DevAddr, l'AppSKey et le NwkSKey vont être générées

Au préalable, le device LoRa doit connaitre : le DevEUI, l'AppEUI, et l'Appkey. Le Network Server doit connaitre le DevEUI, l'AppEUI, et l'Appkey.

Procédure Join La procédure "join" consiste en 2 messages MAC entre un device et un serveur de réseau. Ceux-ci sont appelés join request et join accept.

Le déroulement se passe ainsi comme montré dans la Figure 3.6 :

1. Le device LoRa émet un join request à l'aide des informations DevEUI, AppEUI et AppKey qu'il possède.
2. Le Network Server authentifie le join request et le valide. Il génère alors une NwkSKey, une AppSKey, et un DevAddr.
3. Le Network Server retourne le DevAddr, ainsi qu'une série de paramètres.
4. Les paramètres fournis lors du join accept, associés à l'AppKey, permettent au device LoRa de générer le même NwkSKey et le même AppSKey qui avaient été initialement générés sur le Network Server.

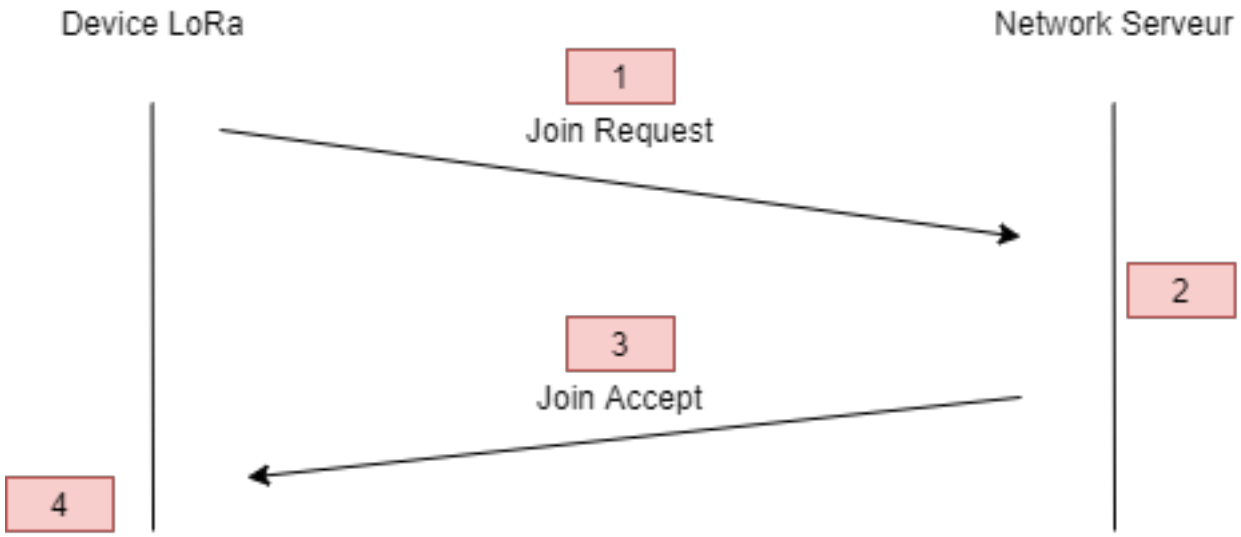


FIGURE 3.6 – Join-Request – Join-Accept en OTAA

3.2.7.2 Activation par Personnalisation

Le device contient directement le DevAddr, NwkSKey et AppSKey à la place de DevEUI, APPEUI et AppKey. Le Device LoRa possède déjà le DevAddr, l'AppSKey et le NwkSKey. Le Network server et le serveur d'application possède déjà le DevAddr, NwkSKey et AppSKey. Il n'y a aucune procédure supplémentaire ou message à utiliser pour rejoindre le réseau.

3.2.8 Différentes versions du LoRaWAN de la 1.0 à 1.1

La principale différence entre LoRaWAN v1.0 et v1.1 est qu'avec la v1.1, l'itinérance (le roaming) des appareils est autorisée et rendue possible par l'introduction d'un serveur supplémentaire appelé "Join Server". La norme v1.1 a également introduit plusieurs améliorations en matière de sécurité. Comme dans le cas de tout système informatique, la sécurité est l'une des plus grandes préoccupations dans l'architecture LoRaWAN. La prolifération des dispositifs IoT (en particulier les dispositifs LPWAN/LoRaWAN) dépend de l'existence d'un réseau public. Il est très important de renforcer le niveau de sécurité des dispositifs LoRaWAN pour obtenir le soutien et l'acceptation du public. C'est pourquoi cette norme s'oriente plus sur la sécurité. Concrètement, le changement opère au niveau des clefs d'identification. Le DevEUI ne suffit plus, il faut ajouter l'élement JoinEUI qui identifie le Join Server [Ism18].

3.3 Opérateur

Comme dit dans les précédentes sections, il est nécessaire d'avoir une architecture d'antennes pour avoir un réseaux LoRaWAN. Lors de nos travaux de recherche nous avons utilisé le réseau proposé par Objenious par Bouygues Télécom.

Bouygues Telecom a lancé son premier réseau LoRaWAN en juin 2015, lors d'une expérimentation à Grenoble. Pour renforcer sa position dans ce secteur de l'IoT en plein essor. Il a créé dès l'année suivante Objenious, devenue sa marque dédiée à l'Internet des Objets. "Chaque jour, plus de 6 millions de messages LoRA circulent sur notre plateforme", a chiffré Stéphane Allaire, son président, lors de la Matinale Connectée d'Objenious en juin 2018.

Membre fondateur de l'alliance LoRa, qui rassemble plus de 500 entreprises, Objenious s'appuie sur des partenaires spécialisés pour perfectionner ses solutions. L'opérateur s'est notamment allié aux sociétés JRI et Oceasoft pour la surveillance de la chaîne du froid ou avec SilentSoft pour effectuer le contrôle des taux de remplissage de cuves de fioul notamment. Dans la gestion de l'énergie, Objenious collabore avec une demi-douzaine d'entreprises, dont Abeeway, Invoxia ou ffly4u.

Objenious propose depuis janvier 2017, un réseau national qui couvre 95% de la population et 86% de la surface en extérieur soit plus de 30 000 communes comme le montre la Figure 3.7.

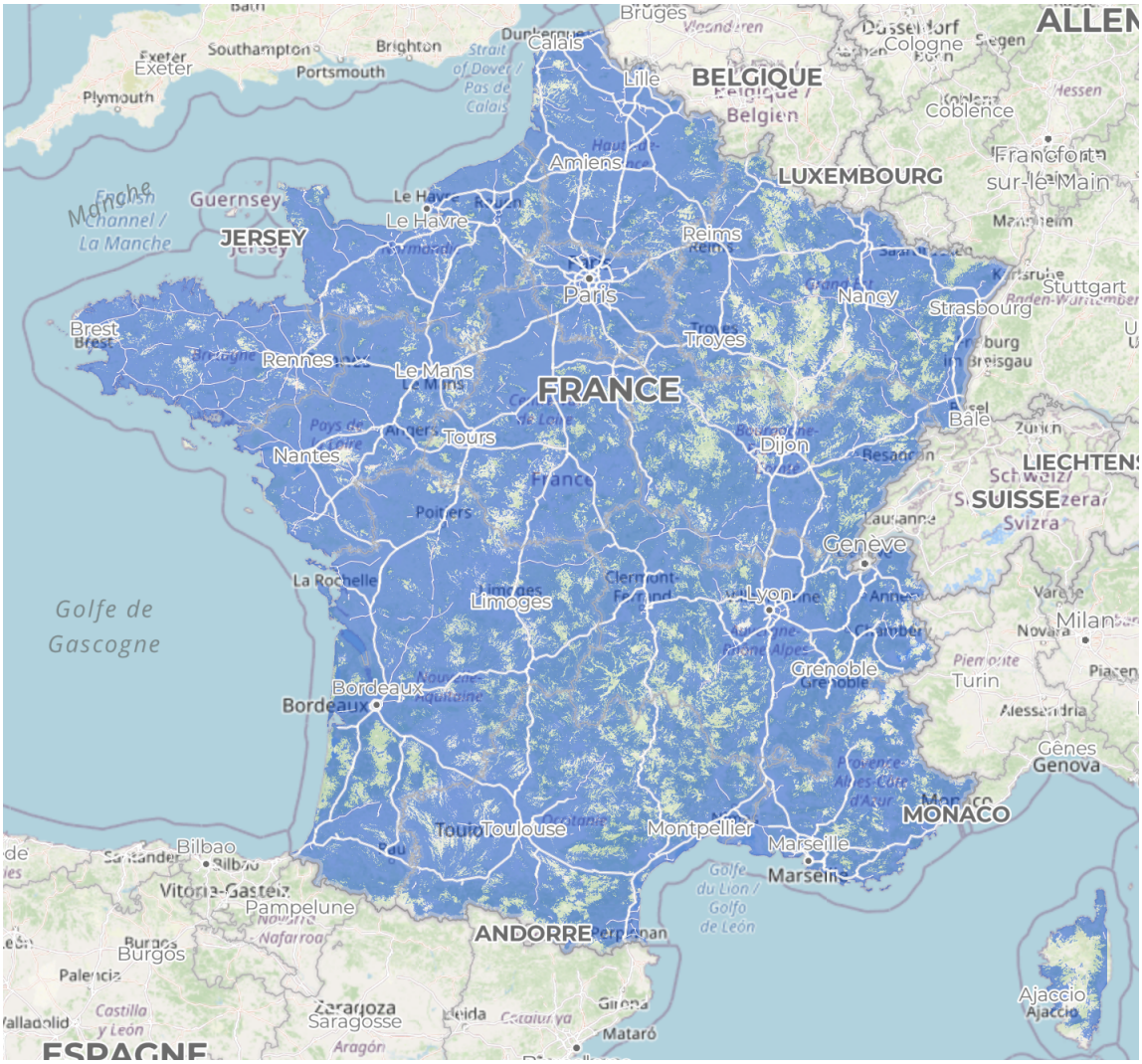


Figure 3.7 – Couverture nationale LoRaWAN de Objenious

Objenious le 19 février 2019 a déclaré accompagner la société Airbus dans l'étude, le déploiement et l'exploitation d'un réseau pour ses objets connectés au service de l'ensemble des métiers Airbus en France et en Europe, en proposant une approche hybride des réseaux LPWan grâce au "réseau dédié". Ci-joint une partie de la déclaration associée à cet événement. *Objenious a été choisi par Airbus pour l'accompagner dans la mise en œuvre de ses solutions connectées. Objenious apporte son expertise réseau, ses services d'audit, de déploiement d'infrastructures, ses outils de monitoring (plateforme SPOT) et un accompagnement au quotidien pour assurer une qualité de service optimale. La formule de "réseau dédiée" LoRaWAN, proposée par Objenious, utilisée par Airbus est un modèle de réseau hybride offrant le meilleur du public et privé* [Obj]

Objenious a deux plateformes pour l'IoT :

— La plateforme GetWay qui offre une autonomie complète dans la gestion de flotte M2M. GetWay permet d'accéder en temps réel à la consommation, à l'état des lignes M2M et aux informations météorologiques du réseau. Le service permet de gérer les cartes SIM, activer, résilier ou suspendre des lignes voire changer d'offres, selon les besoins.

— La deuxième plateforme qui est Smart Portal of Things (SPOT) offre toutes les fonctions de gestion de cycle de vie des capteurs depuis leur déclaration sur le réseau jusqu'à leur fin de vie, en passant par les outils de supervision. Elle intègre également une base de données permettant de décoder les données collectées. Disponible en mode SaaS, elle peut être interfacée à tous types de plateforme « métier » grâce aux API.

3.4 Motivations et objectifs

Comme présenté dans les sections précédentes dans l'architecture LoRaWAN, chaque nœud est connecté à un concentrateur, une antenne dans une topologie en étoile. La question posée est donc : *Que se passe-t-il si un nœud n'arrive pas à joindre le réseau LoRaWAN pour des raisons d'environnement, accessibilité, etc.* C'est ainsi que nous definissons le concept de nœuds isolés.

Définition 3.4.1 (Nœuds isolés)
Le LoRaWAN étant basé sur un modèle d'infrastructure, il existe un cas de figure lorsqu'un nœud n'est pas visible par une LoRaWAN Gateway mais est visible par un end-device, lui même à portée d'une LoRaWAN Gateway. Cette condition est illustrée par la Figure 3.8 .

Il existe un cas particulier qui concerne celui où le nœud n'est visible d'aucun device et d'aucune gateway *LoRaWAN*. Le nœud y étant isolé de manière permanente nous ne pouvons rien faire à ce stade de notre étude.

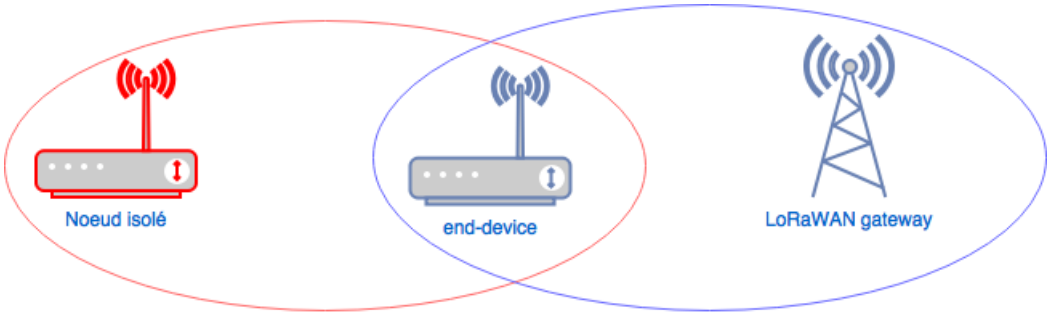


FIGURE 3.8 – Problème des nœuds isolés

La problématique est donc : **Comment gérer un groupe de nœuds LoRaWAN en classe A qui peuvent être des nœuds isolés ou des nœuds type end devices à travers le temps afin de récupérer leurs données tout en étant visible par la plateforme SPOT ?** L'objectif est de gérer les nœuds isolés. Le tout en étant capable d'exécuter et de respecter un ensemble d'élément :

1. Récupérer les informations détenues par les capteurs / nœuds isolés.
2. Gérer les nœuds isolés vers une meilleure gestion de leur énergie.
3. Faire en sorte que du point de vue du serveur réseau et de SPOT les nœuds isolés soit actifs.
4. Développer des classes/librairies pour utiliser l'algorithme de manière transparente.

Pour ce faire nous avons proposé plusieurs pistes qui sont les objets des chapitres suivants : La première solution est la mise en place d'un relais intermédiaire qui commute sur du LoRa RAW pour communiquer avec les nœuds isolés et en LoRaWAN pour communiquer avec le réseau d'Objenious. Cette solution a subi plusieurs déclinaisons et évolutions à travers nos travaux qui sont : La première version de l'algorithme de relais uniforme avec 1 relais et 1 nœud isolé, appelée également LLRP qui est présentée dans le **Chapitre 4**. Nous avons réalisé des simulations pour évaluer l'impact de notre solution simple saut sur un système. Nous avons mis en place des expérimentations et implémenté cette solution dans des objets LoRa.

La deuxième version de notre algorithme de relais uniforme est celle avec 1 relais et K nœuds isolés, appelée également S-LLWURP qui est présentée dans le **Chapitre 5**. Cette version ajoute la sécurité des échanges entre les nœuds isolés et relais mais aussi la prise en charge par un relais de plusieurs nœuds isolés. Nous avons réalisé une campagne de simulations sur les mêmes paramètres.

La troisième étape est la mise en place d'une version de l'algorithme de relais uniforme avec 1 relais et K nœuds isolés avec la prise en compte de la dynamicité et de la sûreté de fonctionnement, appelée également FT-LLWURP qui est présentée dans le **Chapitre 6**. Des simulations ont été réalisées afin d'évaluer l'impact sur les bonnes réceptions, le nombre de messages et la consommation.

La version finale présentée est celle de notre algorithme de relais uniforme en maillage, appelée également MESH-FT-LLWURP qui est présentée dans le **Chapitre 7**. Cette version a subi une campagne de simulation identique à l'étape précédente pour pouvoir évaluer l'impact d'un algorithme maillé.

Nous avons aussi un autre concept sur la remontée des informations des nœuds isolés par le relais avec pour variant une usurpation de l'identité d'un nœud isolé par un nœud relais intermédiaire afin de ne pas surcharger la structure de l'opérateur des concepts de relais et isolé.

3.5 Plateformes

Durant nos travaux, nous avons utilisé plusieurs appareils/plateformes. Dans cette section nous allons les présenter en détails capacité mémoire, processeurs et toutes les autres informations fournis par les constructeurs. Nous avons séparé cette section en 2 sous-sections, la première est celle concernant au sens strict les appareils utilisant directement les librairies STM32 qui gèrent les modules LoRa. La deuxième est celle concernant les appareils de type PyCom, SAMR qui utilisent la bibliothèque STM32 encapsulée dans une autre couche. Ces differents capteurs seront la base en terme de comparaison dans des compagnes de simulation Omnet++ et FLoRa. Ceux-ci seront détaillés dans les futurs chapitres. Chaque feuille de spécifications est mise à disposition dans l'annexe.

3.5.1 STM32

3.5.1.1 STM32 Discovery KIT L072Z

Le kit de découverte LoRa B-L072Z-LRWAN1 (Figure 3.9) est un board de développement pour apprendre et développer des solutions basées sur les technologies LoRa et FSK/OOK. Ce kit de découverte est doté du module ouvert tout-en-un CMWX1ZZABZ-091 de Murata. Le module est alimenté par un micro-contrôleur STM32L072CZ et un émetteur-récepteur SX1276. L'émetteur-récepteur est doté du modem longue portée LoRa, qui assure une communication à spectre étalé à très longue portée et une grande immunité aux interférences, tout en minimisant la consommation de courant. CMWX1ZZABZ-091 étant un module ouvert, l'utilisateur a accès à tous les périphériques du STM32L072CZ tels que l'ADC, le timer 16 bits, le LP-UART, l'I2C SPI et l'USB 2.0 FS (supportant BCD et LPM). Le kit de découverte B-L072Z-LRWAN1 comprend une interface d'outil de débogage embarqué ST-LINK/V2-1, des LED, des boutons-poussoirs, une antenne, des connecteurs Arduino Uno V3 et un connecteur USB OTG au format Micro-B. La stack LoRaWAN supporte la classe A, la classe B et la classe C. Elle est disponible dans le package firmware I-CUBE-LRWAN. Ce board est entre autre équipé d'un Arm® Cortex®-M0+, avec 192 Koctets de mémoire Flash, 20 Koctets de RAM, 20 Koctets d'EEPROM [STb].

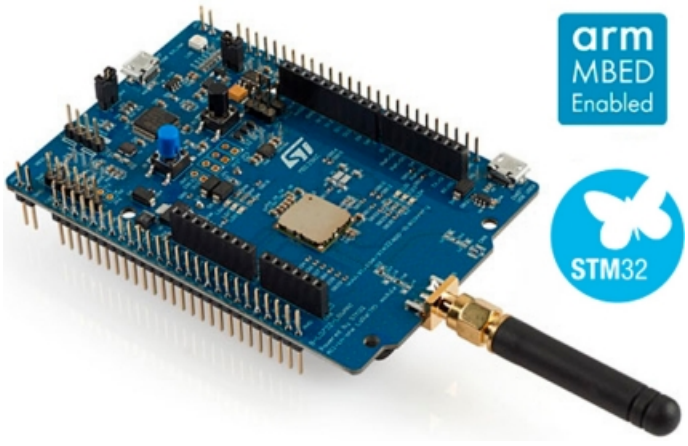


Figure 3.9 – Capteur STM32 Discovery kit

3.5.1.2 STEVAL STRTK01

Le tracker LoRa IoT STEVAL-STRKT01 (Figure 3.10) est conçu et optimisé pour mettre en œuvre les dernières technologies dans les applications de tracker IoT tel que le suivi des biens, des personnes et des animaux ainsi que la gestion de flottes. La carte d'évaluation simplifie

le prototypage, l'évaluation et le développement de solutions innovantes de tracker. Elle est livrée avec un logiciel complet, des bibliothèques de micrologiciels, des outils, une batterie, des câbles et un boîtier en plastique. Grâce au STM32L072CZ embarqué dans le module LoRa CMWX1ZZABZ-091 par Murata, la STEVAL-STRKT01V1 permet d'acquérir la position, de gérer la géolocalisation et l'enregistrement des données du module GNSS Teseo-LIV3F et de contrôler les capteurs de mouvement (LIS2DW12) et environnementaux (HTS221 et LPS22HB). La carte transmet et reçoit également des données, des configurations et des événements vers et depuis le cloud via un réseau LoRaWAN, où sont stockées localement les données dans l'EEPROM M95M02-DR. Le STEVAL-STRKT01 est une solution fonctionnant sur batterie LiPo et met en œuvre des stratégies de faible consommation grâce à un design amélioré de gestion de l'alimentation/de la batterie, basé sur le chargeur de batterie STBC02 et le convertisseur abaisseur ST1PS01, afin de garantir une longue autonomie de la batterie. Le STUSB1600A s'occupe de la gestion du port USB Type-C 5 V et offre une protection haute tension contre les courts-circuits [STa].

FIGURE 3.10 – Capteur STEVAL STRTK01

Interface de développement L'interface de développement utilisée pour programmer le STEVAL STRTK01 et STM32 Discovery KIT L072Z est le STM32CubeIDE en version 1.6.0. Le système d'exploitation hôte a changé à travers le temps en passant de macOs à Windows 10. Auparavant l'IDE utilisé était Eclipse avec le plugin OpenSTM32.

3.5.1.3 Feather

Voici le Adafruit Feather 32u4 LoRa Radio (RFM9x) présenté dans la Figure 3.11. C'est un micro-contrôleur avec un émetteur-récepteur radio LoRa avec USB et chargeur de batterie intégrés. C'est un Adafruit Feather 32u4 avec un module radio 868/915 MHz intégré.

Le Feather est la nouvelle carte de développement d'Adafruit, elle est fine, légère. Le Feather a été conçu pour être un nouveau standard pour les micro-contrôleurs portables.

Il s'agit de la version radio 900 MHz, qui peut être utilisée soit pour la transmission/réception 868MHz ou 915MHz - la fréquence radio exacte est déterminée lorsque le logiciel est chargé car elle peut être réglée dynamiquement. Ce qui est un avantage pour permettre des applications sur les 2 bandes, soit USA, soit Européenne.
Au cœur du Feather 32u4 se trouve un ATmega32u4 cadencé à 8 MHz à 3.3V. Cette puce dispose de 32K de flash et de 2K de RAM, avec USB intégré, ce qui lui permet non seulement de programmer et de déboguer en USB vers série sans avoir besoin d'une puce de type FTDI, mais aussi d'agir comme une souris, un clavier, un dispositif MIDI USB, etc. [Ada]

Voici une liste des spécificités techniques du Feather :

— Module à base de SX1276 LoRa® avec interface SPI

— Packet radio avec bibliothèques Arduino prêtes à l'emploi

— Utilise les bandes ISM sans licence (ITU "Europe" @ 433MHz et ITU "Amériques" @ 900MHz)

— Capacité de sortie de puissance de +5 à +20 dBm jusqu'à 100 mW

— 300µA en veille complète, 120mA en crête pendant l'émission +20dBm, 40mA pendant l'écoute radio active

FIGURE 3.11 – Capteur Feather

Interface de développement L'interface de développement utilisée est EMBitz sur Windows 10.

3.5.2 Pycom

3.5.2.1 Pycom LoPy

La LoPy4 [PyC] est une carte de développement compact à quadruple réseau compatible en MicroPython (LoRa, Sigfox, WiFi, Bluetooth)3.12. Avec le dernier chipset ESP32 d'Espressif, la LoPy4 offre une combinaison parfaite de puissance et de flexibilité.

Le LoPy4 peut servir de nano-passerelle LoRa et de plateforme de développement adaptée à tous les réseaux LoRa du monde. Elle est programmable avec MicroPython et les plugins Pymakr pour un développement rapide. Le LoPy4 est configurable aussi en LoRa RAW pour envoyer des paquets directement entre les LoPy4 via communication point à point.

Dans le cadre des spécifications LoRaWAN, le LoPy est équipé d'un Semtech LoRa transceiver SX1276, de la pile LoRaWAN et de la possibilité d'utiliser en classes A ou C.

Interface de développement L'interface de développement utilisée a été Visual Studio Code avec le plugin PyMakr et Atom avec le même plugin.

FIGURE 3.12 – Capteur LoPy4

3.5.3 Autre boards

3.5.3.1 Ti CC1350

Le kit de développement LaunchPad™ du micro-contrôleur (MCU) sans fil SimpleLink™ CC1350 (représenté dans la Figure 3.13) combine une radio Sub-1 GHz avec une radio Bluetooth® Low Energy pour une combinaison d'intégration facile des téléphones mobiles avec une connectivité longue portée comprenant un processeur ARM® Cortex®-M3 32 bits.

Le dispositif CC1350 est un MCU sans fil ciblant les applications sans fil à faible consommation et à longue portée avec des implémentations Bluetooth à faible énergie. L'unité MCU sans fil CC1350 contient un processeur ARM® Cortex®-M3 de 32 bits fonctionnant à 48 MHz comme processeur principal et un ensemble de fonctions périphériques comprenant un contrôleur de capteurs unique à très faible consommation. Ce contrôleur de capteurs est idéal pour l'interfaçage de capteurs externes et pour la collecte de données analogiques et numériques de manière autonome lorsque le reste du système est en mode veille.

Le LaunchPad CC1350 fait partie de la plateforme MCU SimpleLink de TI, qui offre un environnement de développement unique proposant des options matérielles, logicielles et d'outils flexibles pour les clients développant des applications câblées et sans fil.

Voici quelques caractèristiques techniques du Ti CC1350 [Ins] :

- ARM® Cortex®-M3 de 32 bits fonctionnant à 48 MHz
- Mémoire flash de 128 Ko autoprogrammable dans le système
- RAM de 20 Ko (SRAM)
- Mémoire SRAM à faible consommation de 8 Ko
- Prise en charge multi-régionale : bandes 315 MHz, 433 MHz, 470 MHz, 500 MHz, 779 MHz, 868 MHz, 915 MHz, 920 MHz, 2.4 GHz ISM prises en charge

Interface de développement L'interface de développement utilisée a été Code Composer Studio, un IDE basé sur Eclipse et développé par Texas Instruments.

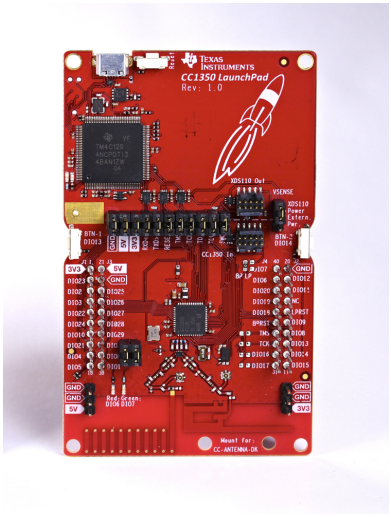


FIGURE 3.13 – Capteur Ti LAUNCHXL CC 1350

3.5.3.2 SAMR34

Le SAM R34/R35 (représenté comme dans la Figure 3.14) est une famille hautement intégrée de dispositifs de technologie LoRa qui comprend un micro-contrôleur (MCU) 32 bits ultra-basse consommation et haute performance, un émetteur-récepteur LoRa et une pile logicielle. Grâce à leurs conceptions de référence certifiées et à leur interopérabilité éprouvée avec les principaux fournisseurs de passerelles et de réseaux LoRaWAN.

Voici quelques caractéristiques techniques du SAMR34 [Mic] :

- Support par Microchip du SAMR34
- ARM Cortex -M0+ CPU jusqu'a 48 MHz
- Mémoire flash de 256 Ko autoprogrammable dans le système
- RAM de 32 Ko (SRAM)
- Mémoire SRAM à faible consommation de 8 Ko
- Paramètres régionaux LoRaWAN 1.0.4
- Support des modes de fonctionnement de classe A et de classe C
- Semtech SX1276/77/78/79
- Prise en charge multirégionale : bandes US915, EU868, AS923, AU915, KR920, IN865 prises en charge

Interface de développement L'interface utilisée est Microship Studio uniquement disponible sur Windows 10.

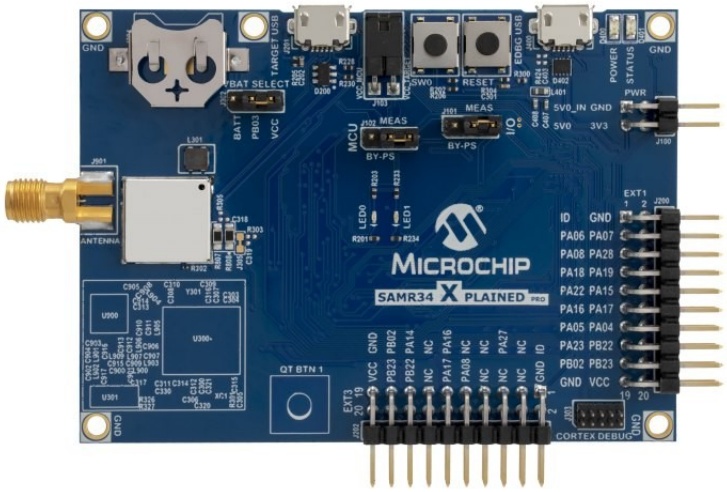


FIGURE 3.14 – Capteur SAMR34

3.5.3.3 nRF9160

Le nRF9160 est un highly-integrated System-in-Package (SiP) (représenté comme dans la Figure 3.15) compact et hautement intégré qui rend accessible et facile à utiliser la dernière technologie LTE à faible consommation ainsi que le traitement et la sécurité avancés, pour une large gamme de conceptions IoT cellulaires (cIoT) à faible consommation et à dispositif unique.

Intégrant un processeur d'application Arm Cortex-M33 uniquement pour les applications, un modem LTE complet, un RF Front End (RFFE) et un système de gestion d'énergie.

Le modem intégré prend en charge à la fois LTE-M et NB-IoT et peut fonctionner à l'échelle mondiale en supprimant tout besoin de variantes régionales. Toutes les fonctions d'économie d'énergie, y compris eDRX et PSM, sont prises en charge, ainsi que le support IPv4/IPv6 jusqu'au niveau du transport et de la sécurité (TCP/TLS). Le micrologiciel du modem peut être mis à jour par le biais de mises à jour sécurisées et cryptées de type FOTA (Firmware Over The Air).

Le processeur d'applications Arm Cortex-M33 est soutenu par 1 Mo de flash et 256 Ko de RAM.

Un récepteur GPS est intégré à la radio offrant différents modes de fonctionnement pour répondre à une large sélection d'applications qui utilisent la fonctionnalité de suivi de la localisation.

Un large choix d'interfaces générales et de périphériques est inclu dans le nRF9160, notamment un CAN 12 bits, un RTC, un SPI, un I^2C, un I^2S, un UARTE, un PDM et un PWM.

La sécurité est la meilleure de sa catégorie grâce à la technologie Arm TrustZone qui permet d'isoler et de protéger les zones normales et sécurisées pour les microprogrammes et les éléments matériels, notamment la mémoire et les périphériques. Arm TrustZone permet de créer des applications IoT solides et sécurisées, avec un démarrage sécurisé, des mises-à-jour de firmware fiables et des implémentations de racines de confiance, sans compromettre les performances.

Arm CryptoCell renforce encore la sécurité en offrant des ressources cryptographiques et de sécurité pour aider à protéger les applications IoT contre diverses menaces d'attaque. CryptoCell est conçu pour les solutions de cryptographie haute performance optimisées pour les appareils à contrainte énergétique.

Le nRF9160 prend en charge les cartes SIM et eSIM pour la connexion et l'authentification avec

les opérateurs de réseaux mobiles [Nor].

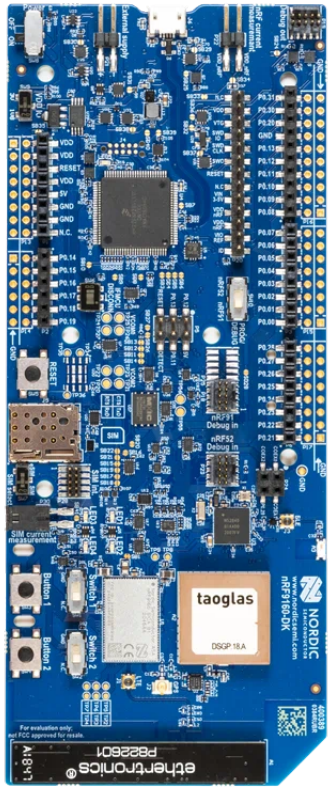


FIGURE 3.15 – Capteur nrf9160

Interface de développement La nRF9160 a été programmée via le SDK nRF Connect.

3.5.3.4 Silicon EFR32

Le kit de démarrage pour SoC sans fil Bluetooth® Low Energy EFR32BG [Lab] (Figure 3.16) comprend une pile logicielle Bluetooth Low Energy et un adaptateur de débogage intégré. Cela permet aux développeurs d'établir rapidement une connexion Bluetooth et d'évaluer les SoC Blue Gecko grâce à la suite d'outils Simplicity Studio. Les développeurs peuvent profiter du développement graphique d'applications sans fil ainsi que du profilage et de l'optimisation visuel de l'énergie. Ce kit prend en charge les spécifications Bluetooth 5, Bluetooth beacon et Bluetooth mesh.

L'avantage de ce board est la modularité et l'intégration de différents modules radio. Voici d'autres caractéristiques :

— Conception modulaire prenant en charge différentes cartes radio
— L'en-tête d'extension permettant une expansion facile
— Débogage et traçage de paquets intégrés
— Prises en charge des connexions Ethernet et USB
— Support AEM (Advanced Energy Monitor) pour la mesure de la batterie

FIGURE 3.16 – Capteur EFR32BG SLWSTK6020B

Interface de développement L'interface de développement utilisée est Simplicity Studio 4.0 et 5.0.

3.5.4 Tableau comparatif et récapitulatif

Pour conclure cette partie sur la présentation des différents boards utilisés tout au long de nos travaux de recherche, voici un tableau récapitulatif 3.1 qui met en avant la pluralité des protocoles, processeurs, caractéristiques différentes :

Plateforme	Connectivité	Processeur	RAM	ROM
Silicon EFR2	Bluetooth	Arm cortex M3	32 kB	256 kB Flash
nRF9160	LTE-M	64 MHz Arm Cortex-M33	256 kB	1 MB
SAMR34	LoRa	Cortex M0+	40 kB	256 kB
CC1350	802.15.4	Arm Cortex-M3	28 kB	128 kB
LoPy	LoRa, Bluetooth, Wifi, Sigfox	Xtensa dual core 32 bits LX6	520 kB	8 MB
Feather	LoRa	Atmega32u4 a 8 MHz	2 kB	32 kB
Stevalk STRTK01	LoRa GPS	Arm Cortex M0+	20 kB	20 kB
STM Discovery Kit L072Z	LoRa	Arm Cortex M0+	20 kB	20 kB

TABLE 3.1 – Récapitulatif des différents boards

3.6 Conclusion

Dans cette section la norme LoRaWAN a été présentée au niveau de l'architecture, des couches d'un réseau LoRawan ainsi que de la sécurité au sein de celles-ci. Les différentes classes d'objet présentes dans la norme ont été présentées. Ainsi nous avons mis en avant la problèmatique des nœuds isolés ainsi que la manière de récupérer leurs informations en utilisant 2 méthodes principales. Pour ce faire nous avons utilisé différents capteurs présentés dans ce chapitre. Dans les prochains chapitres, nous présentons les solutions annoncées dans ce chapitre. Nous commencons donc avec la solution 1 relais 1 nœud isolé appelé Algorithme de relais (LLRP).

Chapitre 4
Algorithme de relais (LLRP)

Résumé : *Dans ce chapitre nous présentons les deux aspects sur lesquels nous avons travaillé : l'usurpation et le relais fonctionnel. Nous présentons les acteurs, l'algorithme formel, les différents messages dans le système ainsi que les différents chronogrammes associés. Pour déterminer l'impact de notre solution, nous avons réalisé une campagne de simulations dont nous présentons les résultats. Par la suite nous mettons en avant les expérimentations et les difficultés associées pour permettre l'implémentation de notre protocole au sein d'un objet LoRa.*

Publications :

1. Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. "A low power LoRa-LoRaWan relay function with a single input, single output device". In : *International Conference on Embedded Wireless Systems and Networks (EWSN)*. Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks. Lyon, France, 2020, p. 283-288. URL : https://hal.univ-reims.fr/hal-02852171
2. Olivier Flauzac, Joffrey Herard et Florent Nolot. "Synchronization Solution to Optimize Power Consumption in Linear Sensor Network". In : *International Conference on Fog and Mobile Edge Computing (FMEC)*. Paris, France, 2020, p. 295-300. DOI : 10.1109/FMEC49853.2020.9144887. URL : https://hal.archives-ouvertes.fr/hal-02909570

Sommaire

4.1 Définitions

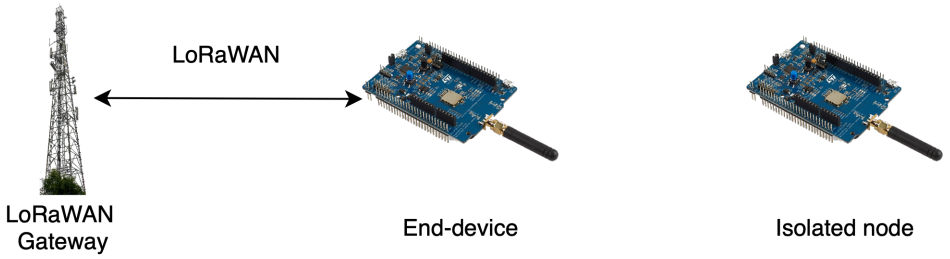


FIGURE 4.1 – Rappel du problème

Nous avons vu la définition de nœud isolé qui est au centre de notre problématique. Nous allons maintenant définir le concept de nœud relais, de nœud end-device et de nœud complètement isolé. Le nœud isolé est abrégé IN. L'antenne LoRaWAN est abrégée LWGW (pour LoRaWAN GateWay).

Définition 4.1.1 (Nœud relais)
Un nœud relais est un end-device qui relaye au minimum un nœud isolé. Abrégé en LoRa Gateway, RN, NR ou LGW pour LoRaGateway.

Définition 4.1.2 (Nœud end device)
Un nœud end-device est toujours défini comme vu précédemment par la spécification LoRaWAN. Abrégé en ED.

Définition 4.1.3 (Nœud complètement isolé)
Un nœud complètement isolé est un nœud qui n'est pas à porté d'un nœud relais ou d'une antenne LoRaWAN.

Dans ce chapitre nous présentons donc 2 aspects qui ont été développés au tout début de cette thèse. Le premier aspect est celui du **relais** qui a été renforcé à travers le temps et le deuxième est celui de l'usurpation.

Nous commençons donc par présenter celui de l'usurpation. Mais cela après avoir vu le schéma global fonctionnel qui englobe les deux aspects, les différents échanges, suivi de l'algorithme sur chaque acteur.

4.2 Schéma global fonctionnel

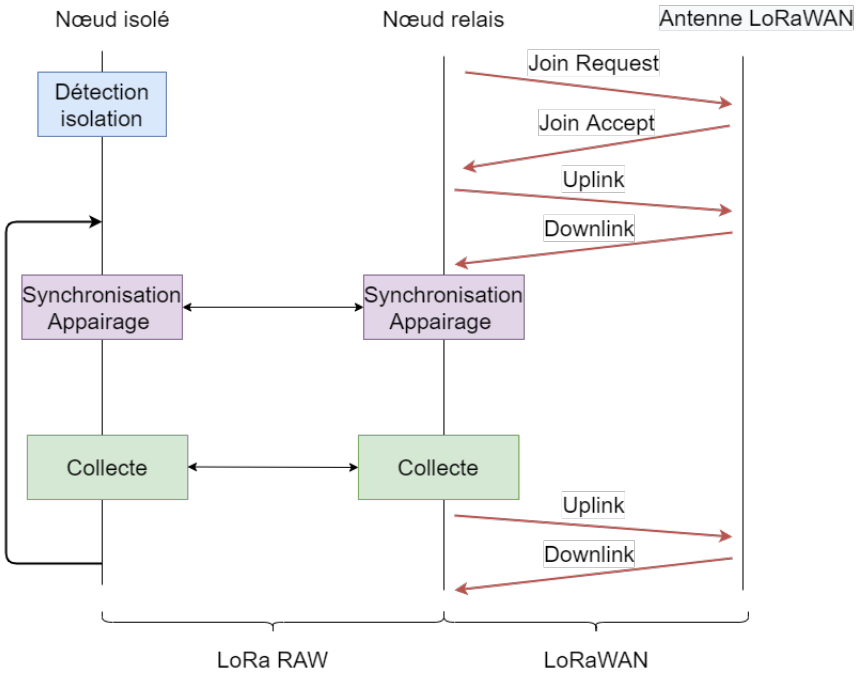


FIGURE 4.2 – Schéma Introductif

La Figure 4.2 représente les étapes de notre solution. Chaque étape va être présentée puis détaillée en terme de message et d'algorithme.

— Déterminer l'isolation : nous nous sommes basé sur un procédé à base de plusieurs join request à la suite espacés d'intervalles de temps relativement courts et aléatoires, sans réponse. Si au bout d'un temps X arbitraire aucun join accept n'est reçu par l'appareil. Le nœud devient isolé. S'il reçoit un join accept il est considéré comme un end-device qui, potentiellement, va devenir un relais.

— Synchronisation : La synchronisation se repose sur un procédé de double poignées de mains. La synchronisation se fait à l'initiative de l'isolé. Ceci de manière arbitraire, le nœud isolé cherchant à s'appairer doit choisir un relais disponible.

— Collecte : La collecte se repose sur le paradigme maitre-esclave. La collecte se fait à l'initiative du relais. La différence entre les deux aspects cités plus tôt se fait au passage en LoRaWAN entre le relais et l'antenne LoRaWAN. Après la récolte, soit l'identité du nœud isolé est repris par le nœud relais qui est procédé d'usurpation, soit le nœud relais identifie chaque payload envoyé à l'antenne LoRaWAN.

4.3 Schéma d'échanges

Pour mieux visualiser l'exécution des algorithmes présentés par la suite, nous avons représenté chaque étapes par les échanges associés dans le système sous forme de chronogramme. Il y a trois phases dans l'algorithme, deux concernent l'appairage et la synchronisation et la troisième concerne la récolte.

— Découverte/Discover

— Enregistrement/Register

— Collecte de donnée/Data Request

4.3.1 Phase de Découverte et d'enregistrement

La Figure 4.3 montre les messages concernés par la phase de découverte à savoir les messages **Discover**, **Accept** et le message **Register**.

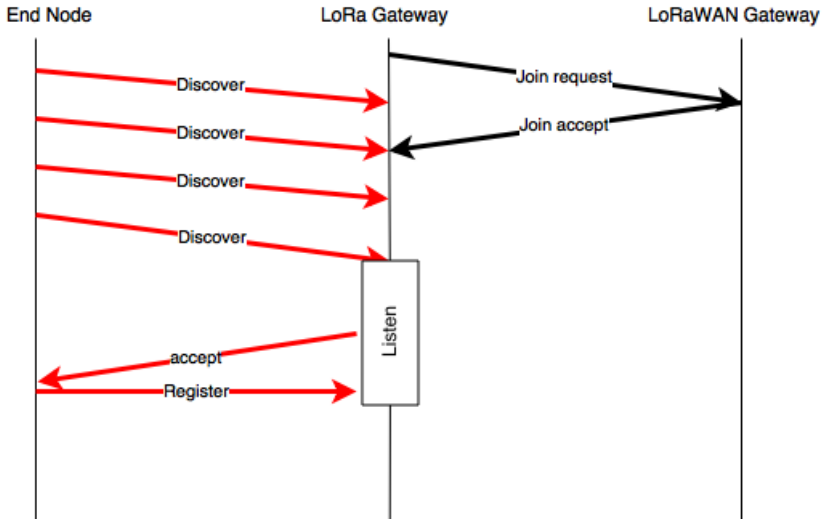


FIGURE 4.3 – Chronogramme d'exécution de l'algorithme

Côté Isolated node Lorsque le nœud isolé se réveille il envoie des messages **Discover**. Si une *LoRa* Gateway répond par un message **Accept** le nœud répond alors avec un message **Register** pour confirmer son appairage avec la *LoRa* Gateway. Ce qui correspond sur la Figure 4.3 aux différentes flèches rouges.

Côté LoRa Gateway Avant toutes opérations avec un élément de l'environnement la *LoRa* Gateway doit s'appairer avec une gateway *LoRaWAN* avec la requête **join request**. Ce qui correspond sur la Figure 4.3 à la flèche noire.

Côté LoRaWAN Gateway La gateway *LoRaWAN* exécute un déroulement normal : elle répond aux requêtes join avec un message **join accept**. Ce qui correspond sur la Figure 4.3 à la flèche noire.

4.3.2 Phase de Collecte

La Figure 4.4 montre les messages concernés par la phase de collecte, à savoir les messages **Request Data** et **Data Response**.

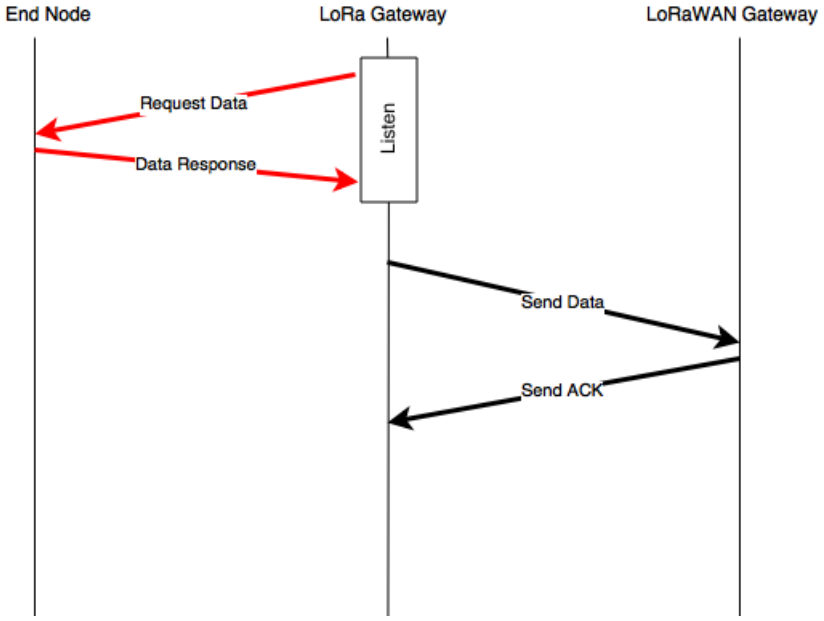


FIGURE 4.4 – Chronogramme d'exécution de l'algorithme

Côté Isolated node Le nœud se réveille à un slot prévu entre la *LoRa* Gateway et le nœud. Tout ceci afin de recevoir la demande de données de la part de la gateway *LoRa*. Le nœud répond avec les données dans un message nommé **Data Response**.

Côté LoRa Gateway La *LoRa* Gateway va se réveiller pour collecter les données des nœuds qui lui sont accrochés. Elle envoie ensuite un message **Request Data**.
Mode Concentrateur : Une fois que toutes les données sont reçues, la Gateway assemble les données et envoie à la *LoRaWAN* gateway.
Mode Proxy : Une fois que toutes les données sont reçues, la Gateway usurpe chaque isolated node et envoie les données à la *LoRaWAN* gateway pour finir par envoyer ses propres données.

Côté LoRaWAN Gateway La *LoRaWAN* Gateway accuse la réception des données envoyées par la gateway *LoRa*. **Send ACK**

4.4 Algorithme

L'ensemble des messages du système sont de la forme :

```
< message_type , source , destination , data >
```

En considérant que chaque message est associé à la fonction du même nom, on peut alors distinguer plusieurs messages dans le système. La fonction correspondante est alors en mesure d'initialiser le type et la source du message tout en prenant en paramètres sa destination et ses données.

4.4.1 les messages nœud isolé → Nœud relais

— les messages `discover`
  — message mis en place pour la découverte d'une `Nœud relais` par un `nœud isolé`
  — `destination = udef` : en broadcast
  — `data = udef` : aucune info
— les messages `pair`
  — message d'appairage d'un nœud isolé sur un Nœud relais
  — `data = udef` : aucune info
— les messages `data_response`
  — réponse à une demande de données de la part d'un nœud relais

4.4.2 les messages nœud relais → nœud isolé

Pour l'ensemble des messages LoRa issus du `nœud relais` la partie data est structurée ainsi :

— `answer_frequency` : fréquence sur laquelle l'`nœud isolé` doit répondre
— `next_slot` : délai d'ici la prochaine fenêtre d'écoute
— `next_duration` : temps fixé de la prochaine fenêtre d'écoute
— `next_frequency` : fréquence de la prochaine fenêtre d'écoute
— `data` : espace de données spécifiques à l'échange

Le message devient donc :

```
< message_type , source , destination , answer_frequency , next_slot ,
                next_duration , next_frequency , data>
```

Les différents messages `nœud relais` → `nœud isolé` sont donc :

— les messages `candidate`
  — message de réponse d'une `nœud relais` après réception d'un `discover` d'un `nœud isolé`
  — `data = udef` : aucune info
— les messages `data_request`
  — message de demande de données
  — `data = udef` si une seule donnée disponible ou `data = requested_data` dans le cas de données multiples

4.4.3 Liste des fonctions utilisées

4.4.3.1 Fonction d'émission

La fonction `sendLora` envoi le message `message` sur la fréquence `frequency`. Le prototype de cette fonction est le suivant : `void sendLora(frequency, message)`

4.4.3.2 Fonction de réception

La fonction `listen` écoute sur la fréquence `frequency` un temps défini par `time`. Le prototype de cette fonction est :

```
(message,time) listen(frequency , source , message_type , time_listen)
```

les valeurs des paramètres de cette fonction sont :

- `frequency` : fréquence d'écoute
- `source` : id de l'émetteur du message
  - `source = udef` : écoute de tous les nœuds sur la fréquence définie
- `message_type` : type de message attendu
  - `message_type = udef` : écoute de tous les types de messages
- `time_listen` : durée de la fenêtre de réception
  - `time_listen = udef` : fenêtre infinie

Valeurs de retour :

- `message` message reçu
  - passage du message dans sa totalité
  - `message == udef` : pas de réception respectant les contraintes
- `time` temps restant basé sur `time_listen`
  - `time == udef` : dans le cas de `time_listen = udef`

4.4.4 Nœud isolé

L'algorithme employé par les nœuds isolés est séparé en deux parties, l'initialisation (1) et l'algorithme en tant que tel (2) :

Algorithm 1 Initialisation des variables de communication

```
1: procedure init_var(msg)
2:     lgw ← msg.source
3:     freq_send → msg.answer_frequency
4:     next_time ← msg.next_slot
5:     timer ← msg.next_duration
6:     freq_listen ← msg.next_frequency
7: end procedure
8:
9: procedure flush_var( )
10:    lgw ← udef
11:    msg ← udef
12:    next_time ← udef
13:    timer ← timer_disco
14:    freq_listen ← freq_disco
15:    freq_send ← freq_disco
16: end procedure
```

Algorithm 2 Algorithme nœud isolé 1-1

```
1: while (true) do
2:     flush_var()
3:                                  ▷ ——————————————————— phase d'appairage
4:     while (msg = udef) do
5:         sendLora(freq_listen, discover(udef, udef))
6:         (msg, t) = listen(freq_send, udef, candidate, timer + rnd())
7:     end while
8:     initVar(msg)
9:     sendLora(freq_send, pair(lgw, udef))
10:
11:                                 ▷ ——————————————————— Phase d'échanges
12:    while (lgw! = udef) do
13:        sleep(next_time)
14:        (msg, t) = listen(freq_send, lgw, data_request, timer)
15:        if msg! = udef then
16:            initVar(msg)
17:            sendLora(freq_send, date_response(lgw, local_data)
18:        elseflush_var()
19:        end if
20:    end while
21: end while
```

4.4.5 Nœud relais

L'algorithme employé par les nœuds relais est séparé en deux parties, l'initialisation (3) et l'algorithme en tant que tel (4) :

Algorithm 3 Initialisation des variables de communication

```
1: procedure init_var( )
2:     freq_send → chose()
3:     timer ← chose()
4:     freq_listen ← chose()
5:     freq_next ← chose()
6: end procedure
7:
8: procedure flush_var( )
9:     timer ← timer_disco
10:    freq_listen ← freq_disco
11:    freq_send ← freq_disco
12:    in ← udef
13: end procedure
```

Algorithm 4 Algorithme lgw 1-1

```
1: LoRaWAN_join()
2: flush_var()
3: while (true) do
4:     if (in == udef) then
5:         (msg, t) = listen(freq_listen, udef, discover, timer)
6:     end if
7:     if (msg! = udef) then
8:         in ← msg.source
9:         init_var()
10:        sendLora(freq_send, candidate(in, freq_listen, slot, duration, freq_next, udef)
11:        (msg, t) = listen(freq_listen, in, pair, timer)
12:        if (msg == udef) then
13:            flush_var()
14:        end if
15:    end if
16:    if (in! = udef) then
17:        init_var()
18:        sendLora(freq_send, data_request(in, freq_listen, slot, duration, freq_next, udef)
19:        (msg, t) = listen(freq_listen, in, data_response, timer)
20:        if (msg! = udef) then
21:            send_lora_data(id + " : " + local_data + "; " + in + " : " + msg.data)
22:        else
23:            send_lora_data(id + " : " + local_data + "; " + in + " : " + udef)
24:            flush_var()
25:        end if
26:    end if
27: end while
```

4.5 Premier aspect : émission par usurpation de capteur

Bien que cette méthode soit plus compliquée à être développée et mise en place, elle possède l'avantage que du point de vue de la plateforme (ici SPOT) les nœuds sont visibles et actifs directement. Contrairement à notre deuxième solution. Toutefois les définitions des acteurs restent les mêmes, à savoir :

— LoRaWAN Gateway (LWGW)

— LoRa Gateway / nœud relais

— Isolated Node (IN)/ nœud isolé

Les messages restent structurés par les mêmes champs et l'algorithme reste globalement le même. La différence se fait sur le traitement de la réponse à la demande de données lors d'une collecte initiée par le nœud relais. On conserve donc la forme de message suivant :

```
< message_type , source , destination , answer_frequency , next_slot ,
                next_duration , next_frequency , data>
```

Dans le champ *Data* du message retour il sera constitué ainsi :

```
"DATA:Appkey:AppNonce;DevNonce:NwkSKey:AppSKey:DevAddr:MIC"
```

Ces éléments sont par rapport à ce que nous avons vu dans la définition de chaque éléments LoRaWAN dans le chapitre 3, des éléments d'identification uniques et propres aux capteurs. Cela suppose que les éléments de sessions, à savoir : AppNonce, DevNonce, NwkSKey, AppSKey, DevAddr, MIC, se créent avec un procédé OTA au moment du premier join accept reçu. Pour ce faire un join doit être procédé et sauvegardé dans le capteur avant le déploiement. En supposant l'existence de ces éléments, la partie "data" du message donnerait par exemple :

```
DATA:304C99263EA5E643B5A08CB3254A61FA:204B36820700002F:9512:
1F7DF0B603278471EE0A177871DCFE62:2BFAB54B2309FB4F096E406F1B27F1A6:E3AC02F:613D250A
```

Une fois cette chaîne reçue, le relais va entamer une procédure d'usurpation en remplaçant ses données d'identifications par toutes les données d'identifications reçues, hormis l'AppKey. L'AppKey faisant ici office de trace de passage. Une fois l'envoi des données effectué le relais termine sa collecte et envoie ses propres données en ayant préalablement remplacé les champs, précédemment modifiés, par ses propres valeurs.
Les chronogrammes restent donc les mêmes sauf que cette fois-ci du point de vue de la Gateway LoRaWAN on a un schéma comme ci-après :

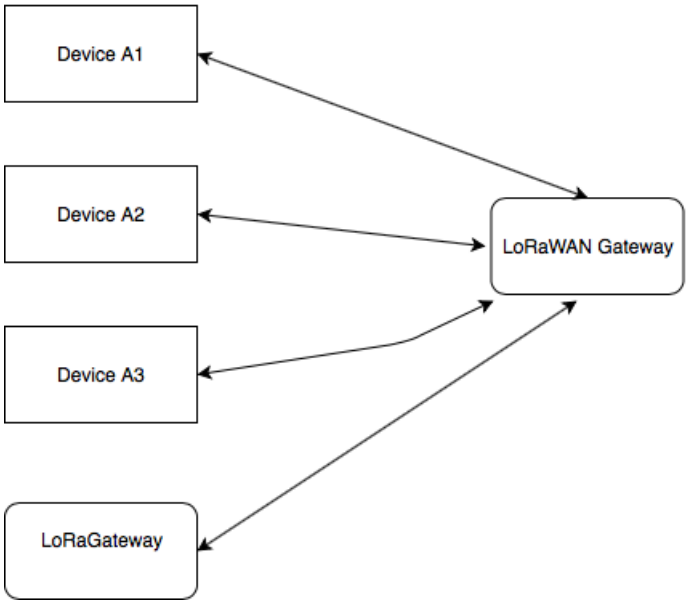


FIGURE 4.5 – Point de vue de la LoRaWAN Gateway

Alors que dans la réalité, quand on observe ce qu'il se passe, on aurait le schéma suivant :

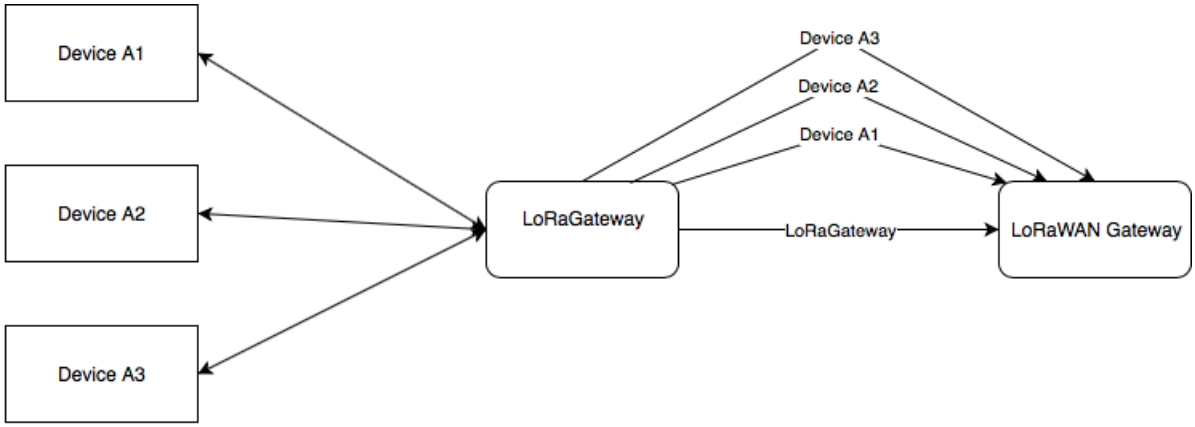


FIGURE 4.6 – Point de vue global

Pour des raisons de sécurité et d'impact mémoire cet aspect n'a pas été retenu. En effet la sauvegarde de chaque élément de sessions serait trop coûteux en mémoire au moment du passage à l'échelle. De plus, faire transiter ces éléments reste dangereux dans le cas d'une attaque Man in the middle. En terme de code cela suppose de faire une activation ABP à chaque fois qu'une remontée est nécessaire, l'opération est également coûteuse en terme de ressources.

4.6 Deuxième aspect : Relais fonctionnel

La différence avec l'autre aspect se fait au niveau de l'interprétation du serveur de réseaux du LoRaWAN. Ici il faut identifier la donnée remontée pour savoir si c'est une donnée d'un nœud isolé ou du nœud relais lui-même. Le payload avec le protocole LoRaWAN n'assure l'identité que de celui qui envoie. Cet aspect sera détaillé lors de la partie "Expérimentation". De plus, cet aspect impacte le nombre de messages, si la donnée d'un nœud est la taille maximale du champ data d'un payload LoRaWAN, alors procéder à une agrégation de message entre le nœud isolé et relais est impossible. Par conséquent, il faut prendre en compte que plus de messages sont nécessaires lors de la remontée de la donnée. L'aspect relais est celui qui fut retenu à travers cette thèse pour garder l'authentification offerte par la couche LoRaWAN et ne pas créer une faille potentielle. Par exemple, voici un payload envoyé par un relais d'un nœud isolé représenté dans le Tableau 4.1. Le payload est le champ PHYPayload, la donnée en tant que telle est le champ FRMPayloadDecrypted qui est **70B3D54999AEEF65FF15** avec **70B3D54999AEEF65** qui est l'identifiant du nœud relayé et ensuite **FF15** étant la donnée.

Champ du message	Paquet
PHYPayload	4032180F0E8000000231D17 93997B7AA376FE3632A4B0D
MHDR	40
MACPayload	32180F0E8000000231 D1793997B7AA376FE3
MIC du paquet	632A4B0D
FHDR	32180F0E800000
FPort	02
FRMPayload decrypted	70B3D54999AEEF65FF15
DevAddr	0E0F1832
FCtrl	80
FCnt	0000 (Big Endian)
FOpts	null
Message Type	Unconfirmed Data Up
Direction	up
FCnt	0
FCtrl.ACK	false
FCtrl.ADR	true

Table 4.1 – Paquet LoRaWAN

4.7 Simulations

Dans cette sous-section nous présentons l'environnement des simulations, la construction des graphes, puis les résultats. Ces simulations visent à vérifier le bon fonctionnement de l'algorithme, à estimer l'impact sur l'énergie et le nombre de messages.

4.7.1 Le simulateur : Omnet++

OMNeT++ est une bibliothèque et une structure de simulation C++ extensible, modulaire et basée sur des composants utilisés principalement pour la construction de simulateurs de réseau. Le terme "réseau" est entendu dans un sens plus large qui inclut les réseaux de communication câblés et sans fil, les réseaux sur puce, les réseaux de mise en file d'attente, et ainsi de suite. Les fonctionnalités spécifiques au domaine telles que la prise en charge de réseaux de capteurs, de réseaux ad-hoc sans fil, de protocoles Internet, de modélisation de performances, de réseaux photoniques, etc. sont fournies par des frameworks de modèles développés en tant que projets indépendants. OMNeT++ offre un IDE basé sur Eclipse, un environnement d'exécution graphique et une foule d'autres outils. Il existe des extensions pour la simulation en temps réel, l'émulation de réseau, l'intégration de base de données, l'intégration SystemC et plusieurs autres fonctions. Même si OMNeT++ n'est pas un simulateur de réseau en soi, il a acquis une grande popularité en tant que plateforme de simulation de réseau dans la communauté scientifique ainsi que dans les milieux industriels, et a constitué une importante communauté d'utilisateurs. Afin de réaliser un ensemble de tests/simulations nous avons utilisé OMNeT++.

4.7.2 Génération des graphes

Nous avons choisi d'utiliser iGraph, afin de générer des graphes aléatoires. iGraph est une collection de bibliothèque servant à créer et manipuler des graphes et analyser des réseaux. Elle

est écrite en C et existe également en tant que paquet Python et R. Il existe également une interface pour Mathematica. Le logiciel est largement utilisé dans la recherche universitaire en sciences des réseaux et dans des domaines connexes.
iGraph a été développé par Gábor Csárdi et Tamás Nepusz. iGraph est disponible gratuitement sous GNU General Public License Version 2.

La génération se fait en plusieurs étapes :

1. Récupération des arguments : nombre de *LoRaWAN* Gateway, *LoRa* Gateway et de Isolated node
2. Création du graphe et ajout du nombre de sommets dont on a besoin
3. Connexion des *LoRaWAN* à une *LoRaWAN* gateway, enregistrés, par définition, à au moins une gateway *LoRaWAN*
4. Connexion des Isolated Node de manière aléatoire à au moins une LoRa Gateway
5. Application d'un deuxième tour d'association de nœuds à des gateways basée sur une méthode Erdös Renyi (on définit la probabilité d'existence entre deux nœuds, pour chaque couple de nœuds, on tire au sort un nombre, si le nombre est inférieur on met le lien)
6. Écriture du fichier .ned qui décrit la configuration du graphe créé

Par exemple, voici le résultat graphique réalisé avec GraphViz dans la Figure 4.7 : Les paramètres

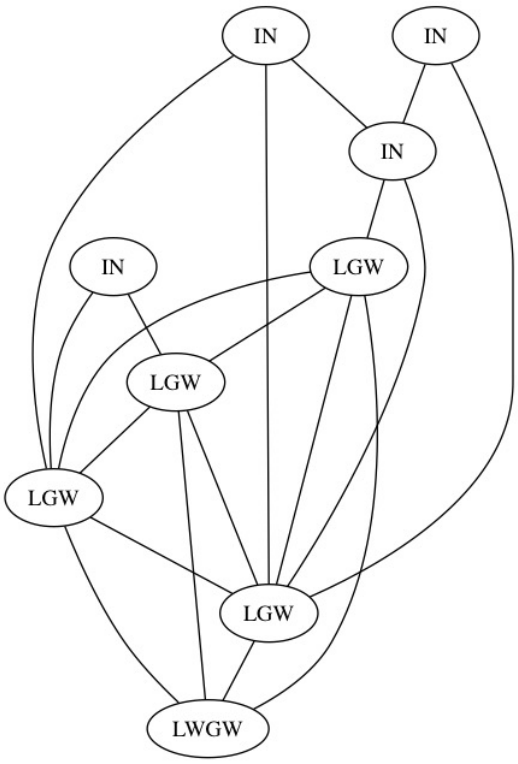


FIGURE 4.7 – Exemple de Graphe de sortie du script

choisis pour la construction ici sont arbitraires en terme de nombre de type de nœud.

4.7.3 Exemples de simulations graphiques

Dans une optique de simplification, les messages contiennent le nom du message et possèdent un code couleur propre et unique. Une *LoRa* Gateway possède une couleur propre à son groupe (elle-même et les IN associés), les INs ont aussi cette couleur. Dans cette partie nous allons mettre en place un système de notation. Par exemple 1/1 : le premier 1 est celui du nombre de noeud isolé et le deuxième celui du relais.

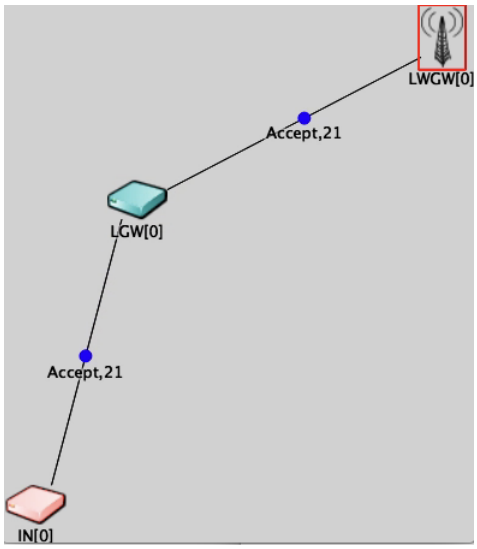


FIGURE 4.8 – Exemple avec un seul IN en 1/1

Simulation 1/1 Cette première Figure(4.8), sortie de la simulation graphique après notre premier test, constitue un exemple typique d'une chaîne avec seulement un élément de chaque composant. Ce fût aussi un terrain de développement pour affiner le comportement individuel et vérifier le fonctionnement correct de chaque composant.

Simulation 3/1 La Figure 4.9 est le deuxième exemple de simulation pour gérer l'enregistrement multiple. Ici nous avons donc toujours une *LoRaWAN* Gateway qui est connectée à une *LoRa* Gateway qui doit gérer trois Isolated Node.

Simulation avec un cas réel La Figure 4.10 est le troisième exemple de simulation pour gérer l'enregistrement multiple ainsi que l'unicité d'un enregistrement sur une Gateway *LoRa* d'un device. Ici nous avons donc toujours une *LoRaWAN* Gateway, connectée à quatre *LoRa* Gateway, qui doit gérer X Isolated Node. Cet exemple est surtout présent pour pousser la simulation à être plus proche de la réalité tout en restant algorithmique. L'exemple présenté ci-contre fait aussi transparaître que les communications sont difficiles. Un nœud peut, pendant un certain temps, ne faire qu'envoyer des messages **Discover** indéfiniment. Pour plusieurs raisons, notamment celles-ci :

— Problème de latence sur le réseau de communication

— Problème de bruit radio (plusieurs émissions sont réalisées en même temps de la part de plusieurs devices, que ce soit sur le device de réception ou d'émission provoquant la corruption des messages). Dans la simulation, la gestion de colision est fait à partir d'un intervalle de temps déclenché à la première réception. Si un autre message arrive sur le temps donné, à savoir le temps "On air" d'un message en LoRa, alors le message est perdu.

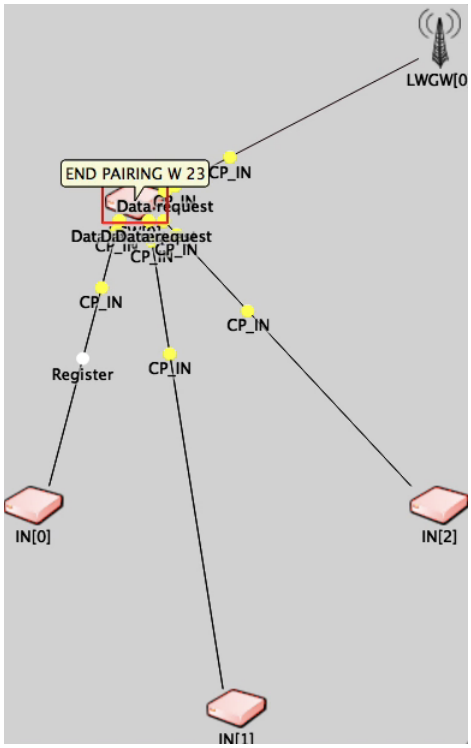


FIGURE 4.9 – Exemple avec 3 IN

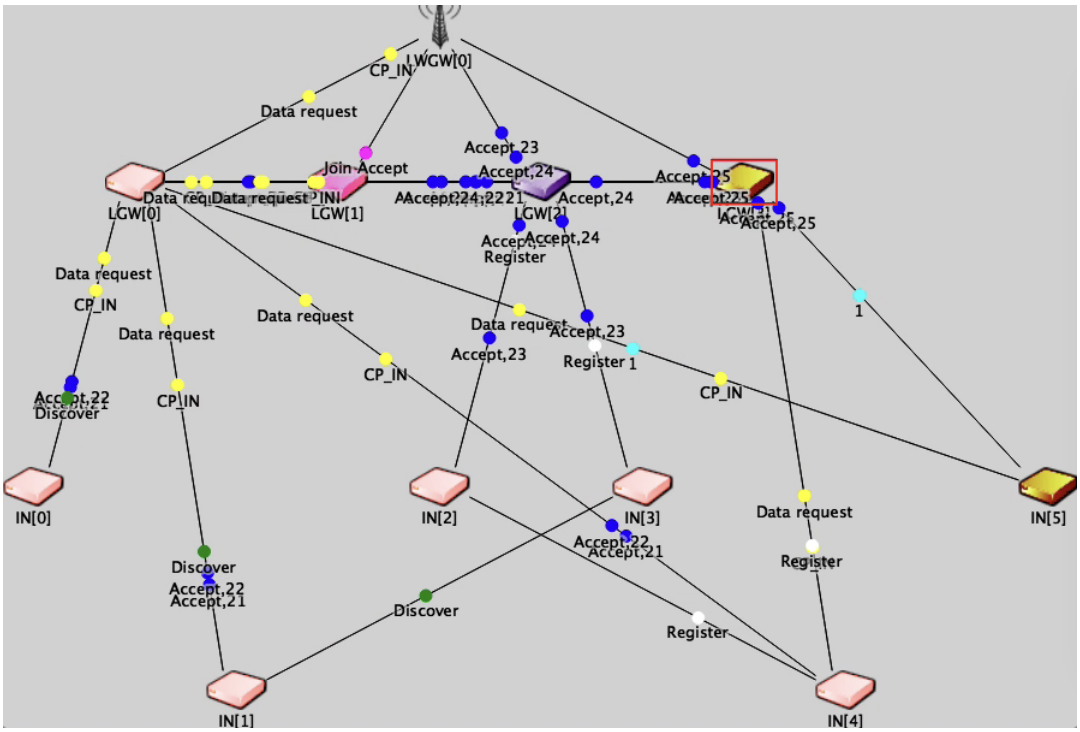


FIGURE 4.10 – Exemple en cas réel à petite échelle 6/4

— Problème de phase réception, en effet des slots sont prévus ainsi que deux fenêtres de réception dans la spécification *LoRa*. Il se peut que deux devices qui ne sont pas encore synchronisés aient du mal à l'être.

4.7.4 Générations des graphes

Un ensemble de 100 graphes contenant arbitrairement 1000 nœuds ont été simulés en respectant cette configuration.

- Variation du nombre de nœuds isolés par passerelle entre 1 et 4
- Variation du nombre de passerelles maximum par puits entre 1 et 4
- Variation du nombre de données à récupérer par jour entre 1 et 24 messages
- Variation sur la méthode de ré-assemblage des données collectées par les nœuds isolés (i.e. envoyés au puit à chaque réception (méthode NA) ou re-assemblés après fusion de tous les messages (méthode A)
- Variation de la vitesse de chaque lien entre deux nœuds

Nous avons donc écrit un algorithme basé sur la probabilité d'existence d'un lien entre deux nœuds. C'est-à-dire qu'au début nous créons le premier puit (LWGW) en faisant attention au degré (celui-ci est de zéro au début) de la passerelle accrochée à lui. On tire une probabilité et si elle est supérieure à celle établie en entrée de l'algorithme on ajoute une passerelle (LGW) et le lien entre le puit et celui-ci. Nous faisons de même avec les nœuds isolés.
Pour résumer, nous avons : Cent graphes avec un maximum d'un nœud isolé par passerelle et un maximum d'une passerelle par puit. Cette configuration sera abrégée 1 (IN) -1 (GW) → 1-1.

Nous avons donc utilisé l'outil Omnet ++ pour simuler tous ces graphes et chaque cas. Dans ceux-ci nous avons noté pour le nœud isolé le nombre de messages envoyés ainsi que la batterie restante à la fin de la simulation. Pour la batterie, nous avons supposé qu'il y avait 6600 mAh ou 23760000 mAs. L'énergie électrique consommée provient de la documentation LoPy, section WiFi. Nos calculs de prévisions sont réalisés comme suit :

L'appareil est paramétré pour envoyer X msg (#msg) par jour. Le courant d'une émission est Ce= 107,3 mA. Chaque émission dure Te = 2s. Le courant en réception est Cr = 37mA. L'heure à la réception est Tr.
Le courant de veille est Cv = 0,531 mA. L'appareil est inactif pour Tv = (24 * 3600- #msg * Te-Tr * Cr).
Le courant consommé sur une journée est donc :
CDay (mA.s) = (#msg * Ceci * Te + Cr * Tr + Cv * Tv)

Nous avons choisi ces paramètres de manière arbitraire. Ces choix visent à être le plus large possible, car nous n'avons pas de chiffre sur le cas d'usage précis et sur l'environnement de celui-ci à ce moment de l'étude. Cependant les données sur la consommation venant du LoPy ont été prise car cette plateforme a été utilisée en amont de manière à effectuer un démonstrateur et donc nous avons continué avec une simulation basée sur cet appareil.

4.7.5 Résultats

Voici donc un graphique montrant la différence entre le mode d'agrégation et de non-agrégation au niveau des messages :

On peut remarquer que dans les figures suivantes un aspect linéaire ressort nettement sur la variation du nombre de messages de la part d'une passerelle quel que soit le mode (avec agrégation ou sans agrégation). Ce que nous pouvons également remarquer c'est que ce nombre, comme prévu, est plus grand et croît beaucoup plus vite sans l'agrégation. Ce qui peut donc être considéré comme suffisamment significatif. En effet, l'ajout des données de la passerelle à celles des nœuds isolés fait que le nombre de messages en mode "relayé" est le même que si les IN étaient des GW et donc des nœuds connectés.

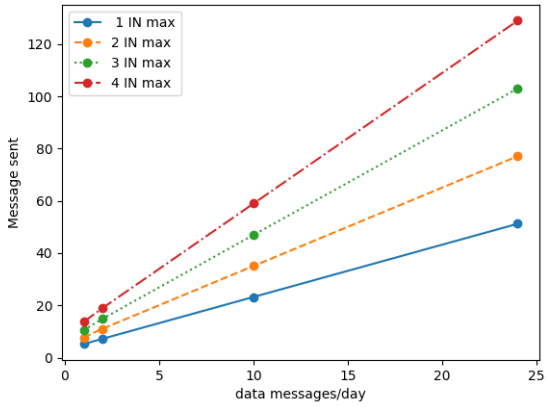


FIGURE 4.11 – Mode : Agrégation, Nombre de message envoyé en fonction du nombre demandé de message par nœud journalier

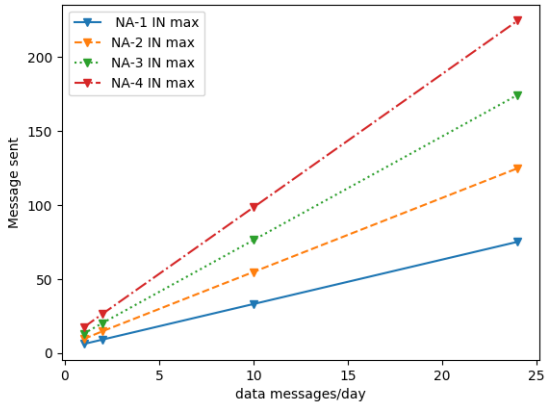


FIGURE 4.12 – Mode : Sans Agrégation Nombre de message envoyé en fonction du nombre demandé de message par nœud journalier

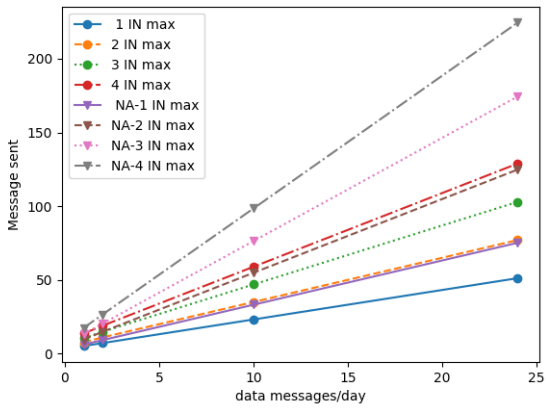


FIGURE 4.13 – Mode : Concentrateur vs Proxy

Concernant l'aspect de l'énergie, et donc de la batterie, la Figure 4.14 met en avant l'autonomie d'un relais avec différents paramètres : le nombre de nœuds isolés attachés au relais, le nombre de messages récoltés par jour et la méthode d'agrégation.

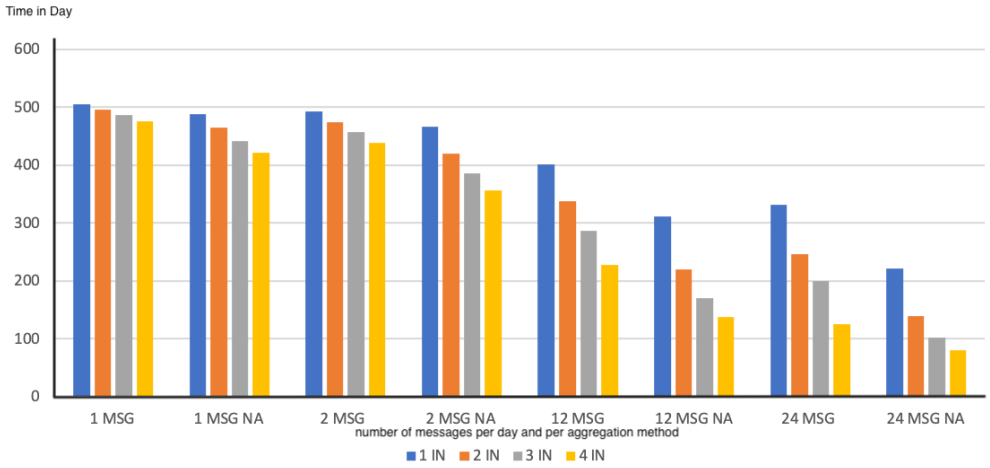


FIGURE 4.14 – Batterie d'un nœud relais en fonction du nombre de message par jour, du nombre de nœuds isolé, et de la méthode d'agrégation utilisé ou non.)

Nous en déduisons que l'augmentation est linéaire en terme de message et de consommation énergétique. En fonction de l'agrégation, celle-ci augmente proportionnellement aux nombre de nœuds isolés.

				Total
Payload Relais	DevEUI	Batterie	Température	
Taille en bits	64	8	8	80

Table 4.2 – Structure du codec

4.8 Expérimentation et problème d'implémentation dans la librairie

Dans cette partie nous voyons comment intégrer notre solution à Spot ainsi que les problèmes rencontrés liés à l'implémentation.

4.8.1 Passage au monde réel avec SPOT

L'intégration de Spot se fait via un codec affilié à un profil de capteur que nous avons déjà mis en place. Un codec SPOT est une configuration qui permet l'interprétation des donnees transmises depuis un capteur. Le CODEC associe a un capteur est soit :

— Disponible sur SPOT s'il a passe le processus de validation de capteur d'Objenious

— Fourni par le fabriquant du capteur

— À rediger par l'utilisateur afin d'etre en accord avec la configuration du capteur

SPOT permet egalement l'ajout d'un CODEC auxiliaire directement dans le profil du capteur. Ce CODEC sera fonctionnel en addition du CODEC existant pour ce capteur. Cela permet de creer des CODEC generiques pour un type de capteurs tout en ajoutant des parametres specifiques a chaque profil.

Le CODEC est redige en JSON et renvoie un resultat egalement en JSON a partir des donnees binaires du payload envoye par le capteur.

Actuellement, s'agissant d'un démonstrateur, les informations affichées et décodées par ce codec sont présentées dans le Tableau 4.2 :

Suite aux discussions avec Bouygues Télécom, dans le cadre de la phase 2, un payload plus fourni qui sera plus complet est proposé. Celui-ci prend en compte la taille des données, si les nœuds pour lesquels le DevEUI qui suit est isolé et d'autres informations qui seront à préciser. Le codec intégré sur Spot se présente sous la forme vue dans la Figure4.15.

```
{
  "defaults":{
    "endian":"big"
  },
  "attributes":{
    "devEUI":{
      "type":"binary",
      "length":64
    },
    "battery_level":{
      "type":"uint",
      "unit":255,
      "length":8
    },
    "temperature":{
      "type":"int",
      "length":8
    }
  },
  "format":[
    {
      "attributes":[
        "devEUI",
        "battery_level",
        "temperature"
      ]
    }
  ]
}
```


FIGURE 4.15 – Codec sur Spot

4.8.2 Problème d'implémentation dans la librairie

Nous avons réalisé des tests d'intégration fonctionnelle et de performance. Ceux-ci ont duré 34 jours sans problèmes précédemment cités (ils ont fonctionné jusqu'à validation de la phase du projet par Objenious). Les paramètres sont arbitrairement choisis pour ce scénario :

- 1 message envoyé en LoRaWAN par heure par un relais et par nœud isolés attachés à celui-ci
- 50 minutes de mise en état de faible consommation énergétique
- 8 minutes en écoute en LoRa avec récolte des nœuds isolés après cette écoute pour le relais

Soit 1 message toutes les heures par nœud isolé après appairage. Soit 2 messages en LoRa et 3 messages LoRaWAN pour le relais par heure.

Durant le développement de cette partie, nous avons rencontré plusieurs problèmes. Pour la mise en contexte, voici l'environnement de développement sur lequel fut implémenté cette solution :

- Eclipse C/C++ avec le plugin OpenSTM32 d'avril 2018
- Librairies V1.1.5 du 30 mars 2018
- Capteur ST Discovery STM32LO72CZY6TR MCU SX1276
- Mac OS X High Sierra

4.8.2.1 Problème dans la librairie STM32 fournie sur la plateforme des Discovery

Les fonctions de comparaison implémentées dans la librairie STM32 ne comparent que les 14 premiers bits des ID des nœuds au lieu des 16 bits. Comme les nœuds peuvent avoir des ID consécutifs. Il est arrivé que certains messages envoyés soient considérés dans le traitement d'un device non concerné par ceux-ci. Ce qui empêchait le bon fonctionnement de notre algorithme. L'identification de ce bug nous a pris beaucoup de temps. En effet, il fallut déterminer qu'une fonction de la librairie standard qui avait été modifiée précédemment par Stackforce était en cause. Cela a donc été corrigé.

4.8.2.2 Problème sur le compteur FCnt LoRaWAN.

Dans la première version présentée à Objenious, le capteur dit "relais" exécutait un **join request** après chaque remontée de ses données et des nœuds isolés. Cela était dû au changement de mode Radio dans la librairie STM32, dans le cas où l'on souhaite passer du mode end-device (LoRaWAN) au mode LoRa. Dans la librairie il y un changement d'utilisation des variables globales et de la mémoire dès que nous passons du mode LoRaWAN au mode LoRa. Toutes les valeurs initialement utilisées pour établir la connexion à la LoRaWAN gateway sont alors perdues. Afin d'éviter des join requests trop nombreuses et coûteuses en énergie, les éléments de session sont gardés en RAM. Nous avons donc utilisé la fonctionnalité de l'activation par personnalisation (ABP). Ce qui a pour effet de changer l'ensemble des variables globales du code et donc du point de vue de la librairie le device a déjà reçu un join accept. Or, avec cette méthode, tous les messages n'étaient pas reçus sur Spot. Pourtant les messages étaient bien envoyés. Afin de comprendre l'origine du problème nous avons étudié chaque trame envoyée par le relais grâce à l'outil LoRawan-packet-decoder [1] [2]. Il a été remarqué que le compteur FCnt n'était pas mis à jour entre chaque envoi. En effet Spot recevait bien tous les messages et effectuait un downlink Technique pour accuser réception dudit message. Ci-joint dans le Tableau 4.3 les trames et leurs décodages par l'outil cité.

1. https ://lorawan-packet-decoder-0ta6puiniaut.runkit.sh/
2. https ://github.com/anthonykirby/lora-packet

Fields	Packet 1	Packet 2 and more
Message Type	Data	Data
PHYPayload	4032180F0E8000000231D17 93997B7AA376FE3632A4B0D	4032180F0E80010002F4DA24 0493C487880F706FC5CAB3
MHDR	40	40
MACPayload	32180F0E8000000231 D1793997B7AA376FE3	32180F0E80010002F4 DA240493C487880F70
MIC from packet	632A4B0D	6FC5CAB3
MIC expected	632A4B0D	6FC5CAB3
FHDR	32180F0E800000	32180F0E800100
FPort	02	02
FRMPayload encrypted	31D1793997B7AA376FE3	F4DA240493C487880F70
FRMPayload decrypted	70B3D54999AEEF65FF15	70B3D54999AEEF65FF15
DevAddr	0E0F1832	0E0F1832
FCtrl	80	80
FCnt	**0000 (Big Endian)**	**0001 (Big Endian)**
FOpts	null	null
Message Type	Unconfirmed Data Up	Unconfirmed Data Up
Direction	up	up
FCnt	**0**	**1**
FCtrl.ACK	false	false
FCtrl.ADR	true	true

TABLE 4.3 – Different paquets LoRaWAN envoyés par un nœud relais

4.8.2.3 Problème de gestion mémoire

Le problème décrit dans cette section était d'actualité à la date de l'implémentation de cette version de notre solution. Il apparaissait que la gestion mémoire devenait problématique à partir d'un certain nombre de messages envoyés en uplink (72) mais aussi en LoRa. La réfléxion autour du problème s'est fait en deux étapes. La première concerne l'utilisation de la gestion dynamique de mémoire en C. En effet, le nombre de nœuds que chaque relais peut gérer étant dynamique, la gestion des données récoltées et les identifiants de chaque nœud se fait à chaque réception d'un nouveau message. Chacune de ces réceptions implique donc de nouvelles allocations mémoire. Il est donc nécessaire de pouvoir la libérer en conséquence. Or il apparaît que malgré l'appel à la fonction de la librairie chargée de la libération mémoire, celle-ci reste allouée. Pour s'assurer que le problème provenait bien de la mémoire, le code a été modifié en lui imposant des limites sur l'allocation. Nous avons constaté que le problème perdurait. La deuxième étape porte donc sur le comportement de la librairie. Il y a fort à parier qu'il y soit présent des allocations mémoire non libérées. Cela serait dû au fait de passer du mode OTAA à ABP ce qui n'est techniquement pas prévue par la spécification.
Pour terminer cette section, l'ensemble des problèmes est propre à la librairie mise à disposition pour un capteur en particulier. La définition même de STM32 est suffisamment vague pour que chaque capteur dispose d'une librairie personnalisée pour un capteur. Nous avons donc regardé les bibliothèques fournies pour les différents capteurs à notre disposition. Celles-ci sont parfois proches, parfois très différentes.

Ce premier aspect a fait l'objet de publications dans des conférences internationales avec comité de lecture [Fla+20b] [Fla+20c].

4.9 Conclusion

Nous avons donc présenté la première version de notre solution qui théoriquement et techniquement n'a été appliquée que pour relayer un seul nœud isolé. Nous avons présenté nos divers résultats de simulations et les problématiques liées à l'implémentation en mode relais fonctionnel. Nous avons également présenté le deuxième aspect que nous avons appelé l'usurpation. Cet aspect n'a pas été retenu pour la suite du développement. Dans le prochain chapitre nous intégrons l'aspect authentification et relais de plusieurs nœuds isolés par un seul nœud relais ce qui actuellement manque à notre solution, nous n'avons décrit que l'aspect théorique. La sécurité liée à l'authentification des nœuds n'a pas été abordée. Présentement tous les messages entre les nœuds isolés relais sont en clair.

Chapitre 5
Passerelle sécurisée uniforme (S-LLWURP)

Résumé : *Dans ce chapitre nous présentons une version sécurisée contrairement à la version du chapitre précédent. De plus cette solution est capable de prendre en charge plusieurs nœuds isolés. Objenious a choisi une méthode basée sur un échange Diffie-Hellman parmi celle proposées pour l'aspect d'authentification. Par la suite nous effectuons plusieurs comparaisons :*

- *Une première sur le modèle d'Heinzelman pour nous comparer à des modèles de références*
- *Une deuxième avec le même modèle que la précédente pour la confronter*
- *La troisième où nous comparons l'efficacité de notre algorithme sur l'autonomie en fonction de la plateforme*

Nous mettons en avant un cas d'étude sur le relais de plusieurs applications différentes entre elles.

Publications :

1. Olivier Flauzac, Joffrey Herard et Florent Nolot. "Synchronization Solution to Optimize Power Consumption in Linear Sensor Network". In : *International Conference on Fog and Mobile Edge Computing (FMEC)*. Paris, France, 2020, p. 295-300. doi : 10.1109/FMEC49853.2020.9144887. url : https://hal.archives-ouvertes.fr/hal-02909570
2. Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. "Comparison between different platform using a Uniform Relay Protocol in Linear Sensor Network". In : *International Conference on the Internet of Things Companion (IoT)*. Malmö, Sweden : ACM, 2020, p. 1-4. doi : 10.1145/3423423.3423462. url : https://hal.univ-reims.fr/hal-02978428
3. Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. "Comparison Between Different Types of Sensor Architecture Using a Uniform Relay Protocol". In : *International Conference on Wireless Networks and Mobile Communications (WINCOM).*

2020 8th International Conference on Wireless Networks and Mobile Communications (WINCOM). Reims, France : IEEE, 2020, p. 1-6. DOI : 10.1109/WINCOM50532.2020.9272428. URL : https://hal.univ-reims.fr/hal-03042336

4. Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. "LLWURP : LoRa/LoRaWAN Uniform Relay Protocol with a Single Input, Single Output Device". In : *Future Technologies Conference (FTC) 2020*. T. 2. San Francisco, United States, 2020, p. 862-876. DOI : 10.1007/978-3-030-63089-8_56. URL : https://hal.univ-reims.fr/hal-02989181

Sommaire

5.1 Introduction et Objectif

Cette seconde version normalise la solution de passerelle LoRa/LoRaWAN appelée ici S-LLWURP à partir de la version précédente présentée auparavant. Les solutions liées à la sécurité des échanges LoRa ont été étudiées et implémentées dans la maquette actuelle. Suite à ce travail de normalisation, une bibliothèque logicielle a été développée, permettant la mise en œuvre de cette nouvelle passerelle LoRa/LoRaWAN sécurisée. Le développement de cette bibliothèque s'est fait sur la même plateforme STM32 [STb], avec une solution de chiffrement validée. Afin de trouver une solution de chiffrement adaptée nous avons réalisé une étude sur les solutions de chiffrement dans le domaine de l'Internet des Objets. Un test en environnement réel clôture cette phase. La montée en charge et les différents paramètres énergétiques ont été étudiés afin de réaliser des simulations. Les résultats des différentes simulations sont présentés. Les processus de provisionning et de déploiement des différents nœuds ont également été étudiés.

5.2 Synchronisation de plusieurs nœuds

La première évolution étudiée ici est l'enregistrement de plusieurs nœuds isolés.

5.2.1 Schéma d'échanges

La solution précédente se fixait sur un relais d'un seul nœud isolé, or, dans le cas d'usage présenté, il y a plus de nœuds isolés. Pour augmenter la prise en charge du relais, nous avons mis en place un enregistrement multiple au niveau du relais à base de tableau. L'algorithme précis est présenté dans la section Algorithme après. La Figure 5.1 met en avant le multi-appairage et le déroulement de l'algorithme.

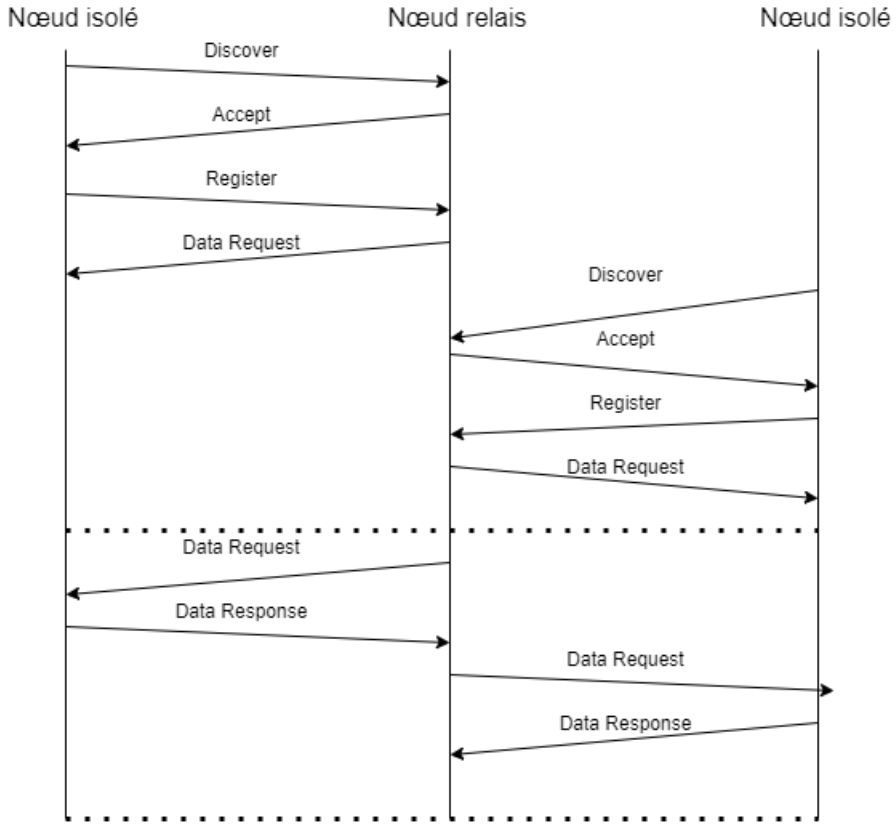


FIGURE 5.1 – Multi-enregistrement et récolte

5.2.2 Algorithme

Pour que le relais puisse gérer plusieurs nœuds isolés, nous avons ajouté des tableaux d'enregistrements de nœuds isolés ainsi que de leurs données. Il y aussi l'intégration d'un timer permettant de délimiter techniquement les moments où le nœud relais peut s'appairer à des nœuds isolés. Voici donc l'algorithme associé en 5.

Algorithm 5 Algorithme lgw 1-n

```
1: while (true) do
2:     while !(isTimeToCollect()) do
3:         if (in == udef) then
4:             (msg, t) = listen(freq_listen, udef, discover, timer)
5:         end if
6:         if (msg! = udef) then
7:             in ← msg.source
8:             init_var()
9:             sendLora(freq_send, candidate(in, freq_listen, slot, duration, freq_next, udef)
10:            (msg, t) = listen(freq_listen, in, pair, timer)
11:            if (msg == udef) then
12:                flush_var()
13:            end if
14:        end if
15:        if (in! = udef) then
16:            if (in NOT in tabRegisteredIN) then
17:                tabRegisteredIN.append(in)
18:                tabRegisteredData.append(−1)
19:            end if
20:            sendLora(freq_send, data_request(in, freq_listen, slot, duration, freq_next, udef)
21:            (msg, t) = listen(freq_listen, in, data_response, timer)
22:        end if
23:    end while
24:    if isTimeToCollect() then
25:        for i= 0 ; i < tabRegisteredIN.length  do
26:            init_var()
27:            sendLora(freq_send,data_request(tabRegisteredIN[i],
   freq_listen,slot,duration,freq_next,udef)
28:            (msg, t) = listen(freq_listen, tabRegisteredIN[i], data_response, timer)
29:            tabRegisteredData[i] = msg.data
30:            strTmpData ← strTmpData + ";" + tabRegisteredIN[i] + " : " +
   tabRegisteredData[i]
31:        end for
32:        send_lora_data(id + " : " + local_data + ";" + strTmpData)
33:    end if
34: end while
```

5.3 Authentification des nœuds isolés

Après avoir vu de quelle manière l'appairage de plusieurs nœuds isolés est géré, nous abordons le deuxieme aspect de cette partie : la sécurité ou du moins l'authentification, à savoir comment s'assurer que le nœud avec lequel le relais est appairé est bien celui qui échange avec nous sur la durée. Pour ce faire nous proposons 3 solutions.

1. Une solution "Over The Air"
2. Une solution appelée "Flash en Usine"
3. Une solution utilisant un procédé basé sur un échange Diffie-Hellmann [DH76]

5.3.1 Solution Over The Air

Cette solution propose un déploiement basé sur des échanges radio à l'aide d'un acteur supplémentaire. Cette première méthode est qualifiée d'OTA. Un nœud intermédiaire supplémentaire serait présent lors du déploiement et capable d'utiliser, d'une part, les protocoles WiFi/Bluetooth et, d'autre part, le LoRa. On pourrait alors provisionner, via le SPOT et l'objet, un élément connu via un échange. Ce qui revient à développer une passerelle 4G/LoRaWAN. Avec cette solution nous avons donc la chaine suivante : le développement d'une application smartphone, le développement d'un autre type de capteur et la mise en place d'un processus spécifique de déploiement. Ce qui represente un cout supplementaire a prendre en compte comme le montre la Figure 5.2. Afin d'éviter ce surcout, la deuxième solution présente, elle, une intervention en usine et non lors du déploiement.

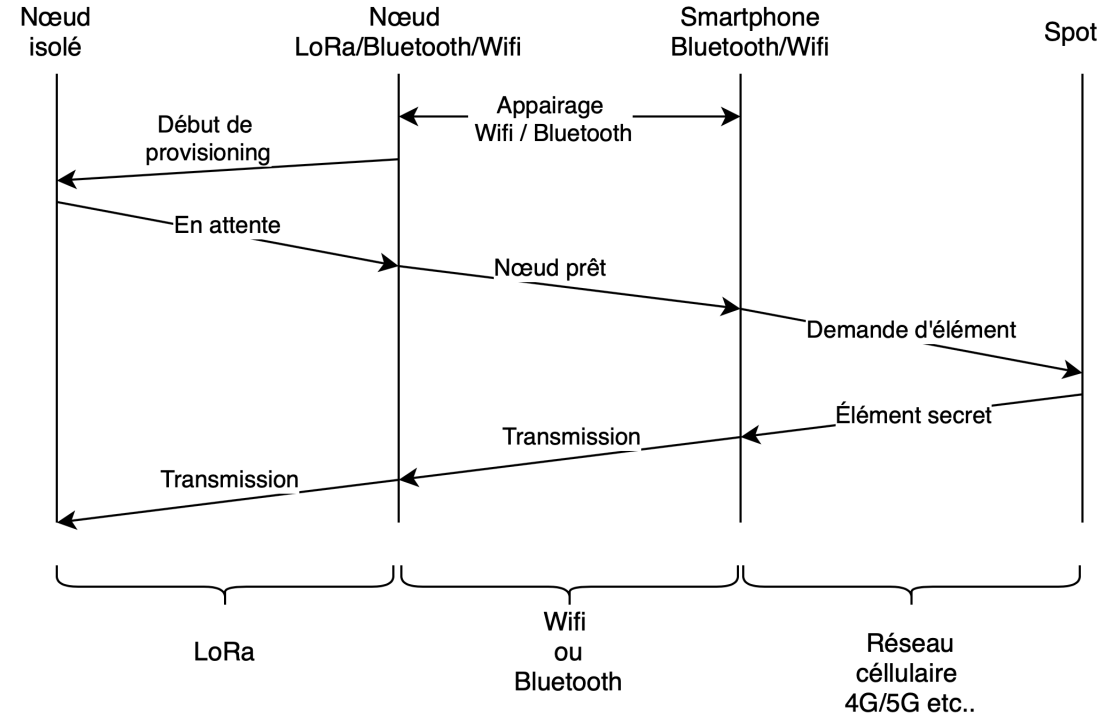


FIGURE 5.2 – Déploiement avec des acteurs supplémentaires

5.3.2 Flash en usine

Cette solution propose via un flash en usine de fournir directement un élément secret unique à chaque nœud. Ainsi le déploiement n'implique l'intervention d'aucun autre acteur comme le montre la Figure 5.3.

Contrairement à la procédure expliquée dans le point précédent. Cette méthode de flash en usine provoque une démarche supplémentaire sur la chaine de production. Cependant il ne peut techniquement pas avoir d'erreur d'accès au réseau via la 4G ou tout autre problème pouvant survenir dans la solution OTA. Ceci étant dépendant de l'environnement de déploiement. Cette

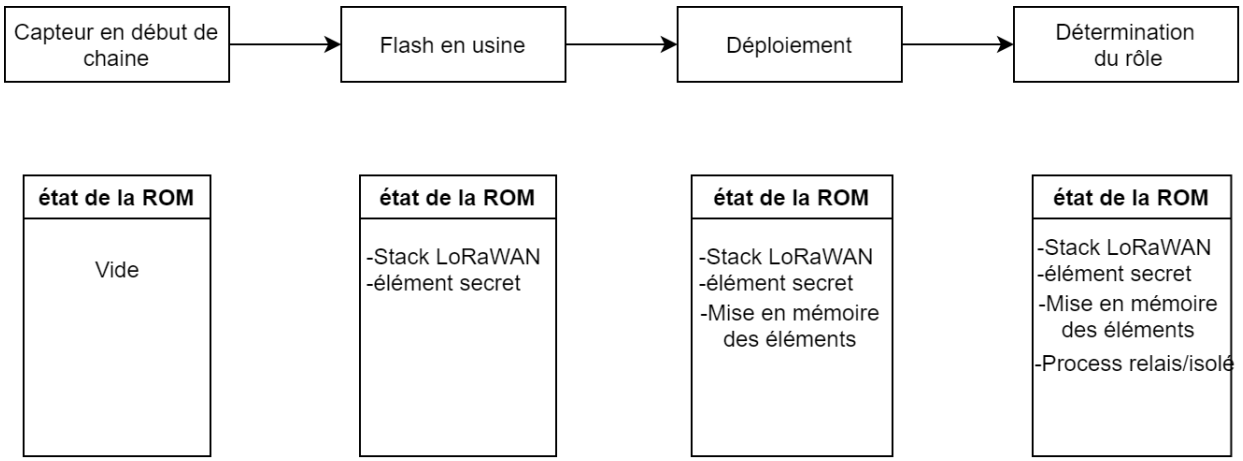


FIGURE 5.3 – Flash en usine d'un élément secret

solution réduit fortement le risque de l'attaque Man in the middle. Il n'y a pas besoin de sécuriser les échanges. Pour que l'attaque Man in the middle soit possible, il faut que l'attaque intercepte les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis. Or ici avec cette méthode cela est rendu bien plus complexe.

5.3.3 Clef commune sécurisée

Précédemment les dernières solutions se basaient sur une intervention humaine soit au déploiement soit en usine. L'avantage de cette solution est qu'elle est indépendante d'une action humaine. Afin de mieux comprendre le procédé qui est implémenté dans les capteurs, nous allons aborder la base fondamentale d'un échange chiffré de type Diffie-Hellmann. En cryptographie, l'échange de clef Diffie-Hellman [DH76], du nom de ses auteurs Whitfield Diffie et Martin Hellman, est une méthode, publiée en 1976, par laquelle deux agents, nommés par convention Alice et Bob, peuvent se mettre d'accord sur un nombre (qu'ils peuvent utiliser comme clef pour chiffrer la conversation suivante) sans qu'un troisième agent appelé Ève puisse le découvrir, même en ayant écouté tous leurs échanges.
L'échange se passe théoriquement de la manière suivante :
1- Alice et Bob ont choisi un nombre premier p et une base g. Dans notre exemple, p=23 et g=5
2- Alice choisit un nombre secret a=6
3- Elle envoie à Bob la valeur A = g^a [mod p] = 56 [23] = 8
4- Bob choisit à son tour un nombre secret b=15
5- Bob envoie à Alice la valeur B = g^b [mod p] = 515 [23] = 19
6- Alice peut maintenant calculer la clé secrète : B^a [mod p] = 196 [23] = 2
7- Bob fait de même et obtient la même clé qu'Alice : A^b [mod p] = 815 [23] = 2

La Figure 5.4 représente l'échange précédemment décrit.

Ces échanges sont intégrés dans les messages Pair/Register et Request Data (cf Figure 5.5). A, g, p sont intégrés dans le message Register. B est lui intégré dans le Data Request. Afin de trouver les nombres premiers, on utilise ici le test de primalité de Miller Rabin [Rab80]. Comme les tests de primalité de Fermat ou de Solovay-Strassen, celui de Miller-Rabin tire parti d'une

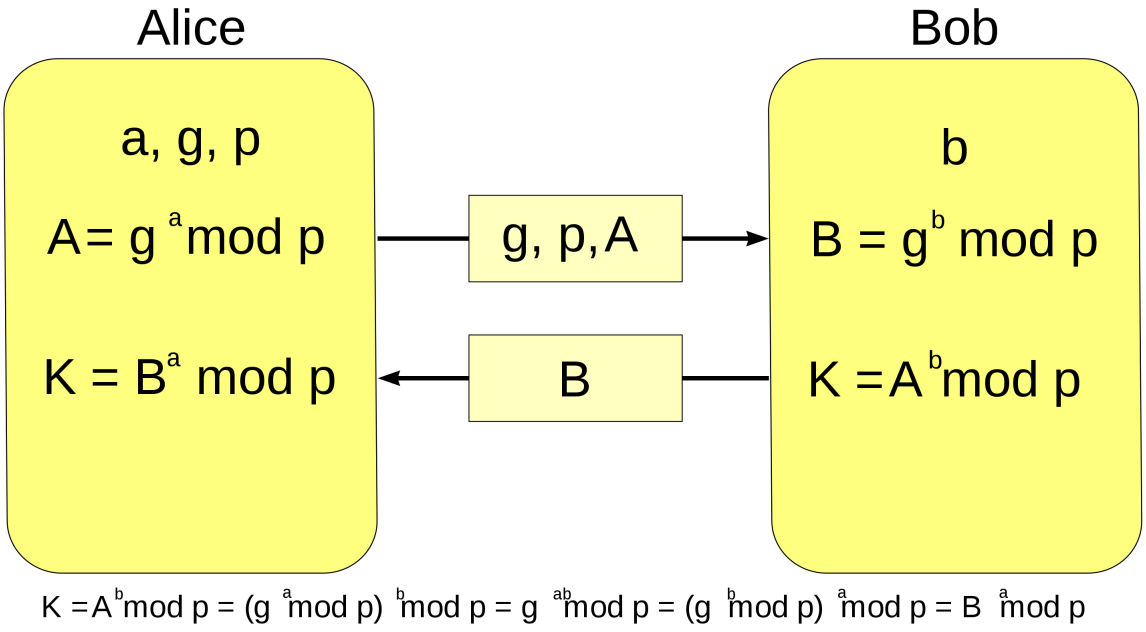


FIGURE 5.4 – Procédé Diffie Hellmann avec Alice et Bob comme acteurs

propriété de l'entier n, qui dépend d'un entier auxiliaire, le témoin, et qui est vraie dès que n est un nombre premier. Le principe du test est de vérifier cette propriété pour suffisamment de témoins. Le test de Miller-Rabin étend le test de Fermat : la propriété est un raffinement du petit théorème de Fermat.

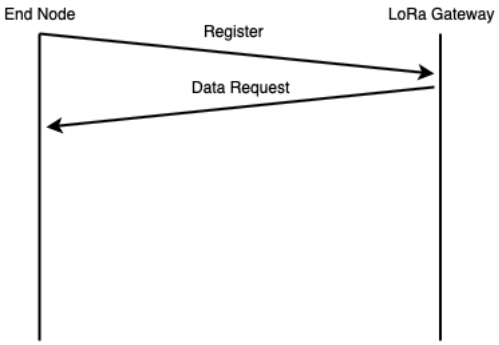


FIGURE 5.5 – Intégration de Diffie Hellmann entre les phases d'enregistrement et de collecte

Il existe d'autres solutions de chiffrement dans le même principe en matière d'échange, mais le calcul serait différent. En se basant sur les courbes elliptiques, il s'agit une fois de plus d'un échange de clefs à la manière de Diffie et Hellman. Si on reprend notre exemple sur Alice et Bob, en première étape ils se mettent d'accord ensemble et publiquement sur une courbe elliptique. Ils choisissent aussi ensemble, et toujours publiquement, un point P situé sur la courbe. Ensuite, chacun, de son côté, et secrètement choisit un entier. Alice choisit kA et Bob choisit kB. Alice envoie à Bob le point de la courbe elliptique kAP, et Bob envoie à Alice le point kBP. Chacun de leur côté, ils sont capables de calculer un élément secret commun.

Nous parlons d'un risque tel que le Man in the Middle, si dès le départ un nœud se met entre le nœud relais et le nœud isolé et ne fait que retransmettre les messages, il peut être l'usurpateur entre les deux parties. Mais comme les nœuds isolés sont dans des environnements complexes, le risque est minime bien qu'existant. Toutes ces intégrations et le travail d'implémentation de cette méthode d'authentification au sein de l'architecture STM32 ont fait l'objet d'un papier publié en conférence internationale avec comité de lecture [Fla+20c].

5.4 Comparaisons Algorithmiques

Le modèle avec lequel nous comparons est celui de Heinzelmann [HCB00].

5.4.1 Modèle énergétique

Pour calculer la dissipation d'énergie pour la communication, nous utilisons à la fois le modèle d'espace libre (perte de puissance d2) et le modèle d'évanouissement par trajets multiples (perte de puissance d4) en fonction de la distance entre l'émetteur et le récepteur. [HCB00] . L'énergie requise pour transmettre un message de k bits sur une distance d est la suivante :

$$E_{\mathrm{Tx}}(k,d) \begin{cases} (E_{\mathrm{elec}} * k) + (\epsilon_{\mathrm{mp}} * k * d^4), d \leq d_0 \\ (E_{\mathrm{elec}} * k) + (\epsilon_{\mathrm{fs}} * k * d^2), d \geq d_0 \end{cases} \tag{5.1}$$

où la distance seuil d_0 est

$$d_0 = \sqrt{\frac{\epsilon_{\mathrm{fs}}}{\epsilon_{\mathrm{mp}}}} \tag{5.2}$$

où E_{elec} est l'énergie nécessaire pour faire fonctionner la radio, ϵ_{mp} et ϵ_{fs} est l'énergie nécessaire pour faire fonctionner l'amplificateur de l'émetteur en fonction de la distance d. Pour recevoir un message de k bits, l'énergie consommée est

$$E_{\mathrm{Rx}}(k) = E_{\mathrm{elec}} * k \tag{5.3}$$

Les données recueillies auprès des nœuds voisins sont redondantes et fortement corrélées. Les données sont donc agrégées au niveau du cluster GW. L'énergie dissipée par l'agrégation de m messages de k bits chacun est

$$E_{\mathrm{DA}}(k) = E_{\mathrm{agg}} * k * m \tag{5.4}$$

5.4.2 Paramètre des simulations

Dans les réseaux de capteurs sans fil le modèle de référence pour effectuer des comparaisons est celui proposé par Heinzelman et *al* dans [HCB00] .

Table 5.1 – Paramètre de simulation

Parameter	***Value***
WSN Area	100m x 100 m
Number of Nodes n	1000
Transmission Range	100 m
Initial Energy	0.1 J
E_{elec}	50 nJ/bit
ϵ_{fs}	10 $pJ/bit/m^2$
ϵ_{mp}	0,0013 $pJ/bit/m^4$
E_{agg}	5 nJ/bit/message
Desired % of CH	5%
Data Packet Size	512 bytes

Tous les nœuds sont distribués aléatoirement sur la zone.

Les résultats concernant les variantes de LEACH sont repris de l'article [AM14]. Il s'agit ici des résultats obtenus sur de petits réseaux. Les paramètres sont expliqués dans leurs articles. Dans notre solution, la durée de vie du réseau est évaluée sur les deux rôles possibles des capteurs.

Nous avons donc les résultats apportés par leur article dans le tableau 5.2 et nos résultats. Nous pouvons remarquer l'impact sur un nœud passerelle. Nous observons que le nombre de rondes est réduit pour les passerelles par rapport aux algorithmes tel que LEACH. Le nombre de tours de passerelles pour les bornes inférieures représente la gestion de 5 nœuds isolés. La limite supérieure représente la gestion d'un seul nœud isolé. L'algorithme LEACH ne gère pas la mobilité, a une évolutivité limitée et ne considère qu'un seul saut. Notre algorithme est le plus mauvais sur le rôle de passerelle de 26,9% à 29,6%. Cependant, sur le rôle des nœuds isolés nous avons un gain de 3,5% à 35,2%. Par conséquent, dans une solution à un saut, LEACH reste le meilleur. Quant à M-LEACH, c'est une solution à un seul saut qui est très bonne en matière d'évolutivité et qui supporte la mobilité. Nous observons également que les résultats étant très proches entre M-LEACH et LEACH, nous avons les mêmes différences. Quant à EEM-LEACH [AM14], il s'agit d'une solution multi-sauts qui supporte l'évolutivité. Quant à notre solution, nous sommes dans une solution multi-sauts avec scalabilité. En effet, nous considérons que les têtes de clusters des algorithmes basés sur LEACH sont assimilés aux relais. Le point d'accès

	LEACH	M-LEACH	EEM-LEACH	Nœud isolé	Relais
Premier nœud à être à court d'énergie	145	146	205	196	[106 - 168]
Moitié des nœuds à court d'énergie	199	194	247	206	[140 - 175]

TABLE 5.2 – Comparaison en round pour des petits réseaux

central est assimilé à une LoRaWAN Gateway.

Conclusion sur la comparaison avec LEACH Notre algorithme est moins économe en énergie de 48,3% et 43,3% pour les nœuds passerelles. C'est en comparant notre algorithme avec EEM-LEACH que nous avons montré que les nœuds isolés sont moins efficaces sur le plan énergétique. EEM-LEACH est meilleur dans les solutions multi-sauts présentées. Ceci étant dû à leur choix algorithmique sur la formation des clusters et au cluster-head. Nous avons des résultats intéressants sur les nœuds isolés avec jusqu'à 35% d'autonomie supplémentaire.

Ces résultats ont fait l'objet d'un article publié en conférence internationale avec comité de lecture [FHN20].

Après avoir comparé notre solution algorithmique à d'autres solutions comme LEACH et des variantes comme EEM-LEACH et M-LEACH, nous allons comparer la deuxième version de notre solution à la première mais aussi à son application avec différents protocoles de communication.

5.5 Comparaison avec LLRP

Pour rappel, l'environnement de simulation est le même que celui décrit dans la section 4.7 afin de pouvoir comparer. Nous continuons donc de faire une campagne de simulation avec l'outil OmNET++. La génération des graphes ainsi que la modélisation des échanges radio est la même une fois de plus, comme décrit dans la section 4.7.1. Comme le choix de l'authentification s'est fait sur un procédé Diffie-Hellmann, nous avons intégré les données nécessaires à ce procédé dans les messages Register et Data Request. Par conséquent, les résultats sur le nombre de messages dans le système sont identiques et présentés dans la section 4.7.4. La différence que nous avons voulu observer est l'impact sur la durée de vie, l'autonomie. Pour ce faire nous avons pris un modèle énergétique similaire à la première campagne avec cette fois-ci des chiffres de consommation sur la plateforme STM32 choisie qui a une architecture validée par Bouygues [STb]. C'est l'objet de la prochaine section.

Les graphes générés respectent ces règles-ci :

— Variation du nombre de nœuds isolés par passerelles entre 1 et 4

— Variation du nombre de passerelles maximum par puits entre 1 et 4

— Variation du nombre de données à récupérer par jour entre 1 et 24 messages

— Variation sur la méthode de ré-assemblage des données collectées par les nœuds isolés (i.e. envoyés au puit à chaque réception (méthode NA) ou re-assemblé après fusion de tous les messages (méthode A)

— Variation de la vitesse de chaque lien entre deux nœuds

5.5.1 Modèle énergétique utilisé

Nous avons réalisé des séries de mesures en montant en série un ampèremètre sur le capteur ST Discovery (pour rappel la configuration est celle-ci : STM32LO72CZY6TR MCU SX1276 avec 32MHz ARM cortex M0 Core, 192 Kbytes de mémoire flash, 20 Kbytes de SRAM). Nous avons développé plusieurs profils afin de déterminer la consommation de chaque action, telle que l'émission d'un message, la réception et le chiffrement. Les profils ont alors été programmés ainsi :

— Profil d'émission : 1000 émissions, mesure du temps fonctionnel de chaque émission

— Profil de réception : réception à l'infini avec mesure du temps moyen de traitement d'une réception

— Profil de chiffrement : génération de la base, et des nombres premiers ainsi que de 1000 chiffrements avec mesure du temps d'exécution sur chaque chiffrement

— Profil de veille : mise en veille permanente après avoir désactivé les senseurs Ci-dessous le tableau récapitulatif des mesures effectuées et des mesures fournies par la communauté pour la même plateforme [for]

Pour relever la consommation électrique nous avons mis en série un ampèremètre qui mesurait le courant à travers un câble USB dénudé. Ce montage a permis de pouvoir flasher le capteur en même temps que l'on pouvait noter l'ampérage sur les différents profils. Comme le montre la figure 5.6.

Nos calculs de prévisions sont alors réalisés avec les configurations identiques à celles présentées dans la section 4.7.3.

Lors de la première phase, nos simulations étaient basées sur la spécification des PyCom en WiFi, car le tout premier démonstrateur était réalisé sur un PyCom. Nous avons réalisé des

	ST32 Discovery consommation annoncé par la communautée	ST32 Discovery mesurée
Cout Réception	11 mA	65 mA
Cout émission	38 mA	85 mA
Cout Veille	0.004 mA	37mA
Temps de réception	725 ms	728 ms
Temps d'émission	0.0011ms	0.0010ms
Cout chiffrement	x	55.6 mA
Temps de chiffrement	x	0.0015 ms

TABLE 5.3 – Consommation relevée et annoncée pour le STM32Discovery Kit

FIGURE 5.6 – Mesure de la consommation des capteurs.

simulations basées sur les données des ST32 Discovery annoncées pour les Ce, Cr, Cv, Te, Tr et Tv. Nous avons fait le choix des valeurs annoncées car elle sont plus réalistes que celles relevées. En effet, il semble y avoir des problèmes au niveau des librairies sur la gestion de la batterie.

Toutefois aucune information sur la consommation ne fut trouvée pour un chiffrement basé sur Diffie-Hellmann.

Nous relevons pour chaque nœud isolé le nombre de messages qu'il a envoyé et également le taux de batterie restant à la fin de la simulation. Nous avons alors supposé qu'ils ont respectivement 200 mAh ou 720 000 mAs de batterie, ce qui équivaut à une pile bouton CR2032.

5.5.2 Résultats LLRP vs S-LLWURP

Le graphique ci-contre permet la représentation de l'autonomie de nos gateways (exprimée en jour) en fonction du nombre de nœuds isolés auxquels elles sont attachées. Il permet également la représentation d'autres facteurs qui influent sur l'espérance de vie d'une gateway, tels que : le nombre de messages par jour et leur méthode de remontée. À savoir qu'il est noté A pour une méthode de utilisant l'agrégation et NA pour une méthode de remontée sans agrégation. Nous remarquons que le nombre de messages est le facteur le plus influent, suivi du nombre de nœuds isolés.

Cela n'a rien d'étonnant dans la mesure où ils sont intimement liés : - Perte de 71,7% d'autonomie entre 1 et 24 messages par jour pour 5 nœuds isolés - Entre 1 et 24 messages par jour il y a perte de 35% d'autonomie pour 1 nœud isolé

A titre d'information concernant les simulations réalisées avec les mesures que nous avons effectuées, l'autonomie de nos gateways était extrêmement dérisoire (ici de 5 à 7 jours maximum).

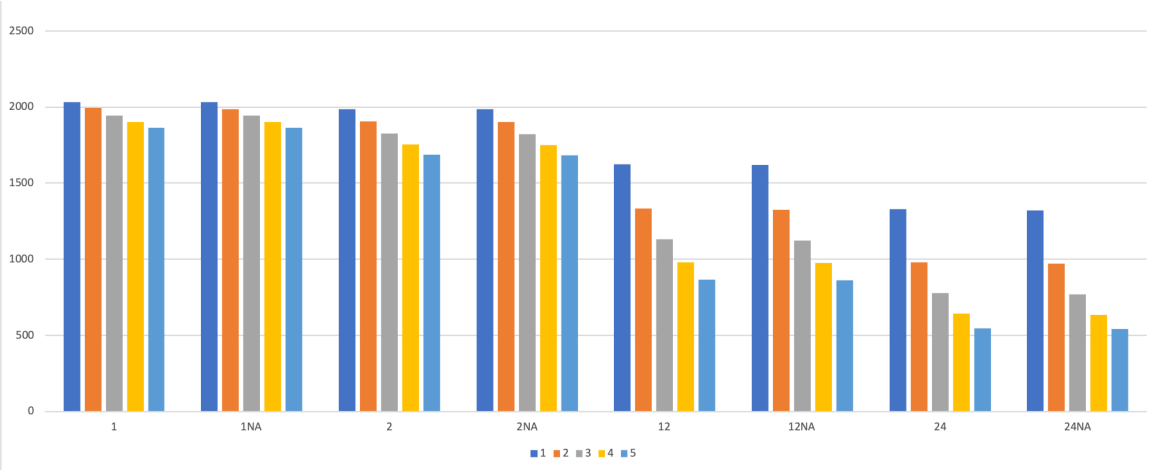


FIGURE 5.7 – Espérance de vie en jours pour un relais en fonction du nombre de nœud isolé, du nombre de message par jour, avec le cout du chiffrement

Ces derniers graphiques restent une estimation avec le coût de chiffrement (Figure 5.7) et il ne sera pas présenté de graphique équivalent avec la non prise en compte de ce paramètre car la différence ne serait pas assez visible compte tenu de l'autonomie. Une représentation de l'autonomie avec ou sans chiffrement est toutefois possible, comme le montre le graphique 5.8. Le calcul représenté est donc le suivant : (Autonomie en coût d'un capteur sans chiffrement) - (Autonomie en coût d'un capteur avec chiffrement) = nombre de jour à l'avantage du mode sans chiffrement. On peut alors remarquer "l'absence" d'un écart entre la partie avec et sans chiffrement pour les cas suivants :

— 2, 4, 5 noeuds isolés avec 1 message par jour en mode agrégation (A)
— 1, 2, 3, 4, 5 noeud(s) isolé(s) avec 1 message par jour en mode non-agrégation (NA)
— 2, 3, 4, 5 noeuds isolés avec 2 messages par jour en mode (A)
— 1, 5 noeud(s) isolé(s) avec 2 messages par jour en mode (NA)

Ce qui nous permet d'estimer que le coût du chiffrement sur l'autonomie est relativement faible quand un nombre assez faible de messages est demandé. À l'inverse, on remarque qu'il y a plus de changements dans le mode (NA) car on constate directement l'impact sur l'autonomie. Chaque message étant chiffré, chaque retransmission est déchiffrée avant d'être chiffrée avec les éléments de sessions de la gateway ce qui agrandit le surcoût du chiffrement sur l'autonomie énergétique.

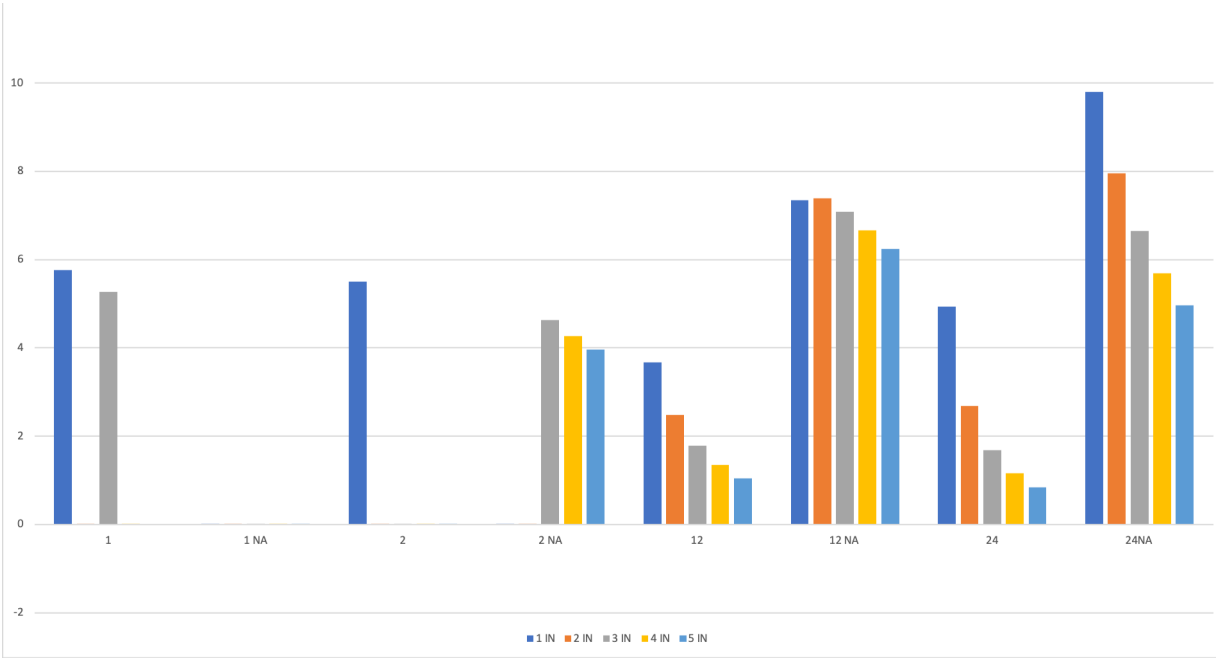


FIGURE 5.8 – Différence entre le mode sans chiffrement et avec chiffrement en fonction du nombre de messages par jour et du nombre de nœud isolés pour un relais.

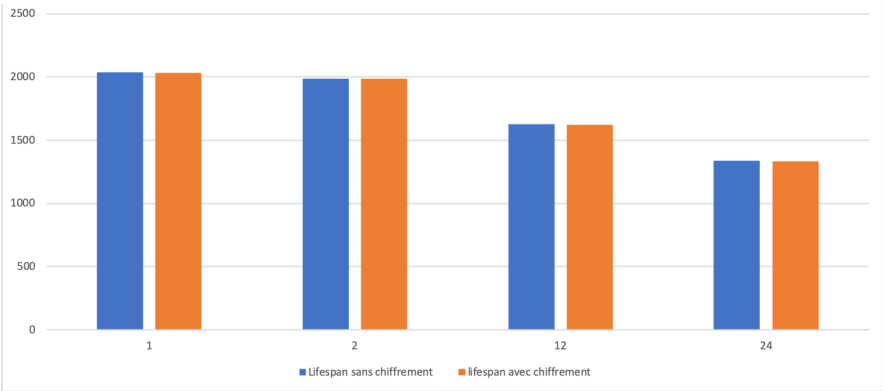


FIGURE 5.9 – Espérance de vie pour un nœud isolé en fonction du nombre de messages

La Figure 5.9 représente l'autonomie d'un nœud isolé en fonction du nombre de message et si le chiffrement est utilisé ou non. La méthode d'agrégation et le nombre de nœuds isolés attachés à la gateway n'ayant pas de rapport avec l'autonomie d'un nœud isolé, nous ne pouvons observer qu'une différence infime entre les deux graphiques présentés. Il n'y a donc aucun coût de la part de l'agrégation pour un nœud isolé. L'autonomie maximale avoisinant plus de 2 000 jours.Les quelques jours de différence ne représentent respectivement qu'une différence de : **0,282%** ; **0,249%** ; **0,304%** ; **0,370 %** .

Cette différence est représenté par la Figure 5.10.

5.6 Comparaison avec WiFi, LoRa, 828MHz : impact du protocole

Après avoir comparé de manière algorithmique notre solution, nous l'avons comparé avec la version précédente de notre solution. Maintenant nous allons comparer en expérimentant cette solution sur diverses plateformes et divers protocoles radio. Notre solution est plus économe en

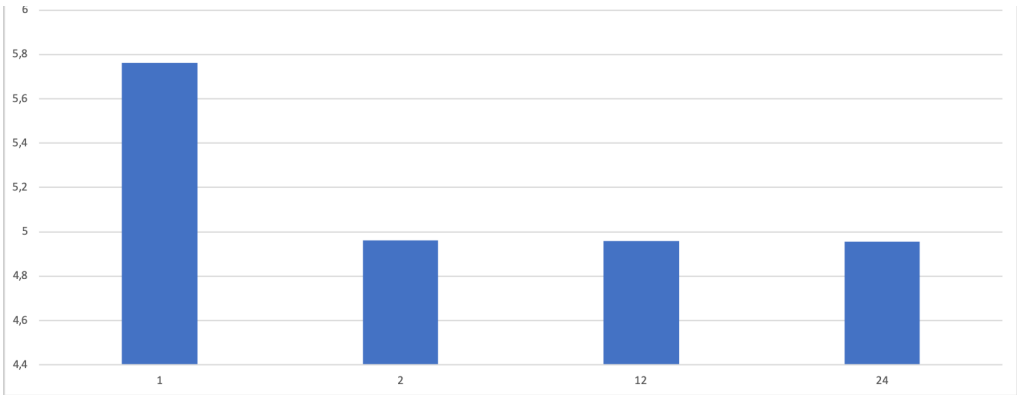


FIGURE 5.10 – Jour de différence à l'avantage de l'algorithme LLRP

fonction d'un protocole ou d'une plateforme. Pour ce faire nous avons utilisé au cours de cette comparaison quatre capteurs différents, et trois modulations radio différentes. Les différentes caractéristiques sont présentées dans le Tableau 5.4.

— ST Discovery B-L072Z-LRWAN1 avec le SX1276 [STb]
— LoPy en LoRa mode avec le SX1272 [PyC]
— LoPy en WiFi [PyC]
— Ti CC1350 en 868 RF [Ins]

Cette étude est basée sur la consommation des différentes fiches techniques. Le Tableau 5.5 résume la consommation énergétique de chaque action, ainsi que leur temps de traitement pour chaque capteurs. Lorsque les mesures ont été effectuées sur le LoPy en WiFi, la consommation

TABLE 5.4 – Les différents capteurs de notre cas d'étude

	Ti CC1350 ZigBee sensortag	LoPy WiFi	LoPy LoRa	ST32 Discovery
CPU	Cortex(R) M3	Xtensa(R) dual-core LX6	Xtensa(R) dual-core LX6	Cortex(R) M0+
RAM	28KB	520KB + 4MB	520KB + 4MB	20 Kbytes
Radio	RF 868	802.11b/g/n	LoRa 868	LoRa 868
Disk Space	128KB	8MB	8MB	20 Kbytes

n'a pas changé quelle que soit l'action d'émission ou de réception. Le temps de réception et de transmission dépend du temps d'écoute.

Les données décrites précédemment ont été prises en compte dans les simulations afin de pouvoir comparer la capacité de batteries équivalentes. Les LoPy sont capables de recevoir des batteries LiPo 5V avec diverses capacités énergétiques, alors que les CC1350 ne reçoivent que des piles CR2032 3V. Les simulations sont basées sur la capacité d'entrée minimale. La pile bouton utilisée par la sensorTag CC1350 représente 220 mAh. Le deuxième paramètre de la simulation est l'action de l'agrégation par le *nœud relais*. L'agrégation est définie ici, comme le moyen de rapporter les données des *nœuds isolés*. En mode agrégation, il existe soit un message pour le *nœud relais* **et** *nœuds isolés*, soit un seul message par nœud. Les résultats montrent quel

TABLE 5.5 – Power consumption

	ST32 Discovery annoncé	ST32 Discovery mesuré	CC1350 sensortag ZigBee	LoPy LoRa	LoPy WiFi
Receiving cost in mA	11	65	14	99	99
Transmit cost in mA	38	85	30	108	104
DeepSleep cost in mA	0,004	37	0,080	0,0019	0,0019
Receiving time in ms	725	728	164	824	X
Transmit time in ms	1,1	1	8,9	1,0	X

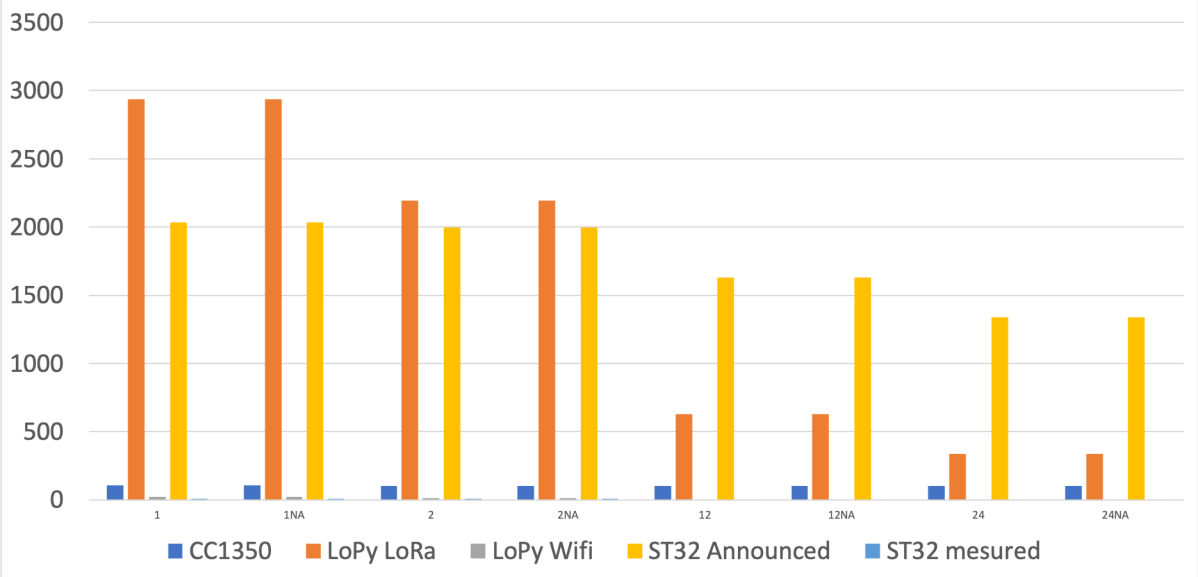


FIGURE 5.11 – Durée de vie des nœuds isolés

capteur est le plus économique en fonction du nombre de messages. Les résultats de la simulation sont représentés sur la Figure 5.11 concernant l'autonomie maximale des *nœuds isolés*. La seule influence sur leur autonomie est le nombre de messages de données par jour. En effet, il n'y a aucune influence sur le nombre de *nœuds isolés* dans leur environnement. Plusieurs choses peuvent être remarquées sur la Figure 5.11. Le capteur CC1350 est le moins sensible à la décharge de sa batterie en fonction du nombre de messages par rapport aux autres qui subissent une plus grande dégradation de leur capacité énergétique. Ceci est particulièrement vrai pour le LoPy utilisant LoRa. LoPy est passé de plus de 2900 jours à à peine 300 jours d'autonomie avec 24 messages par jour, soit autant qu'un CC1350. On peut noter que les annonces énergétiques de ST32 Discovery sont très intéressantes. Avec un message par jour, il peut atteindre près de 2100 jours d'autonomie.

Après avoir mesuré la consommation d'énergie des ST32 Discovery kit, le calcul de leur durée de vie est très affecté, il est presque invisible sur le graphique en raison de l'échelle du diagramme qui ne décrit que 7 jours d'autonomie. Il existe également deux figures distinctes, la Figure 5.12 représente la durée de vie des nœuds relais dans le cas d'une architecture de réseau de capteurs strictement linéaire avec notre solution. La Figure 5.17 est construite comme suit :

— Le nombre de messages par jour et par nœud isolé varie entre 1 et 24.

— La transmission des messages des nœuds *relais* et *isolés* se fait, elle, en agrégation (A ou

NA).

— Le nombre de *nœuds isolés* attachés au *nœuds relais* est de 1 à 5.

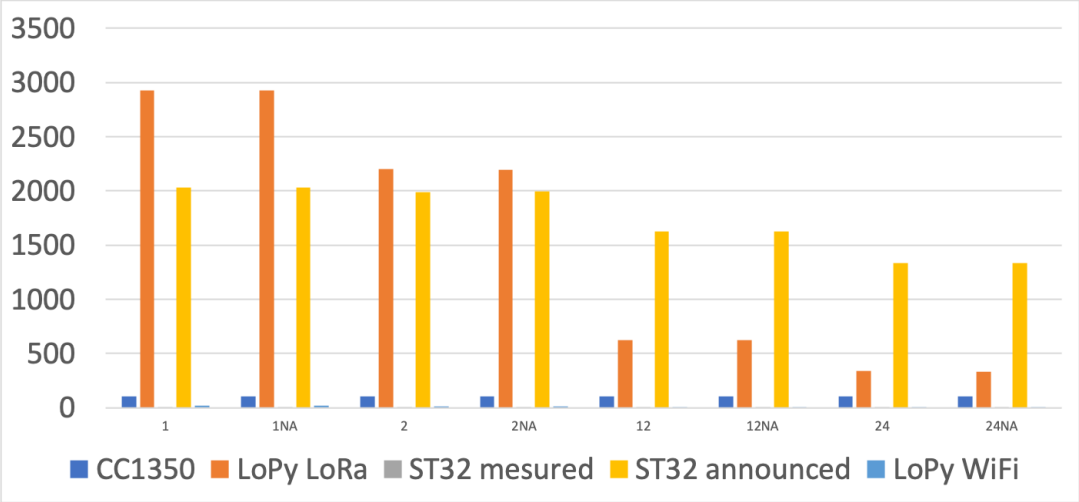


FIGURE 5.12 – Durée de vie des différents capteurs pour un nœud relais avec 1 nœud isolé

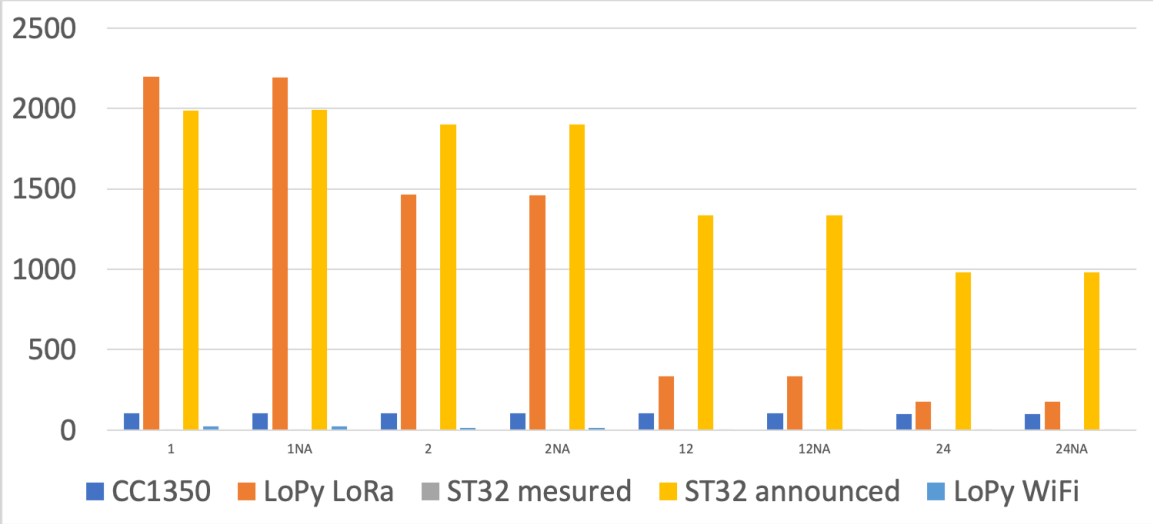


FIGURE 5.13 – Durée de vie des différents capteurs pour un nœud relais avec 5 nœuds isolés

L'information la plus intéressante est la tendance au début ; la meilleure autonomie pour un seul message par jour est obtenue par le LoPy utilisant LoRa comme le montre la Figure 5.17. Cette tendance disparaît lorsque le nombre de *nœuds isolés* augmente (sans que le nombre de messages n'augmente). La kit Discovery ST32 devient plus intéressant à ce stade, comme le montre la figure 5.17 . Cette tendance, selon laquelle la Discovery Kit de ST est la plus économe en énergie, devient très claire par la suite, même lorsque le nombre de messages augmente, mais aussi lorsque le nombre de *nœuds isolés* augmente. En termes de LoRa, selon les résultats obtenus, le ST32 Discovery reste le plus économique en matière de consommation électrique que le LoPy. La constance concernant la consommation électrique du CC1350, qui malgré l'augmentation du nombre de *nœuds isolés* appairé et du nombre de messages est à noter .

LoPy peut être choisi uniquement pour le test et le prototypage, mais sa conception ne permet pas de le mettre en production à long terme. De plus, les capteurs embarqués ont un impact significatif du fait de leur moyen radio ou de la plateforme d'implémentation. C'est ce que montre la figure 5.17.

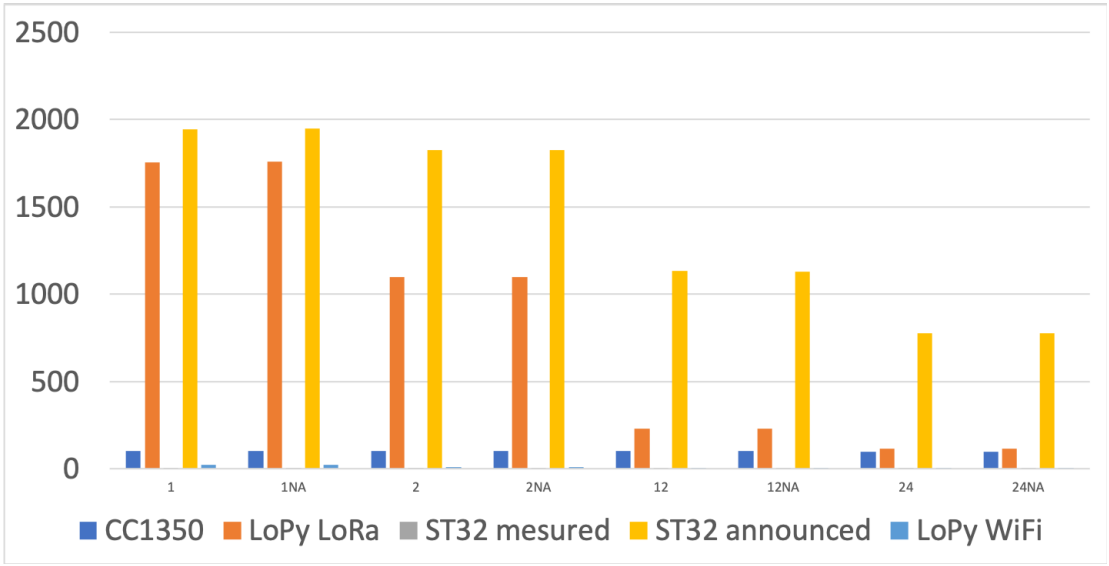


FIGURE 5.14 – Durée de vie des différents capteurs pour un nœud relais avec 3 nœuds isolés

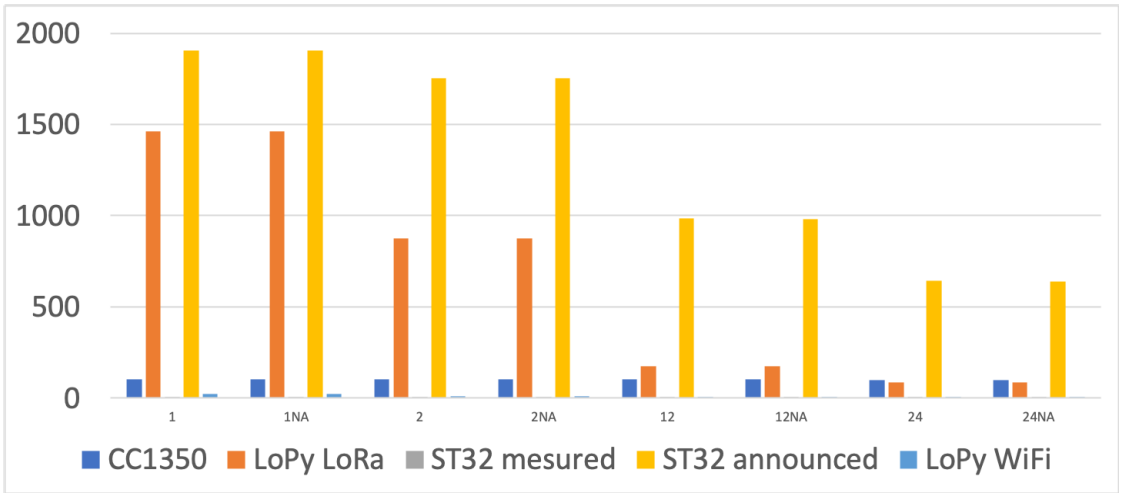


FIGURE 5.15 – Durée de vie des différents capteurs pour un nœud relais avec 4 nœuds isolés

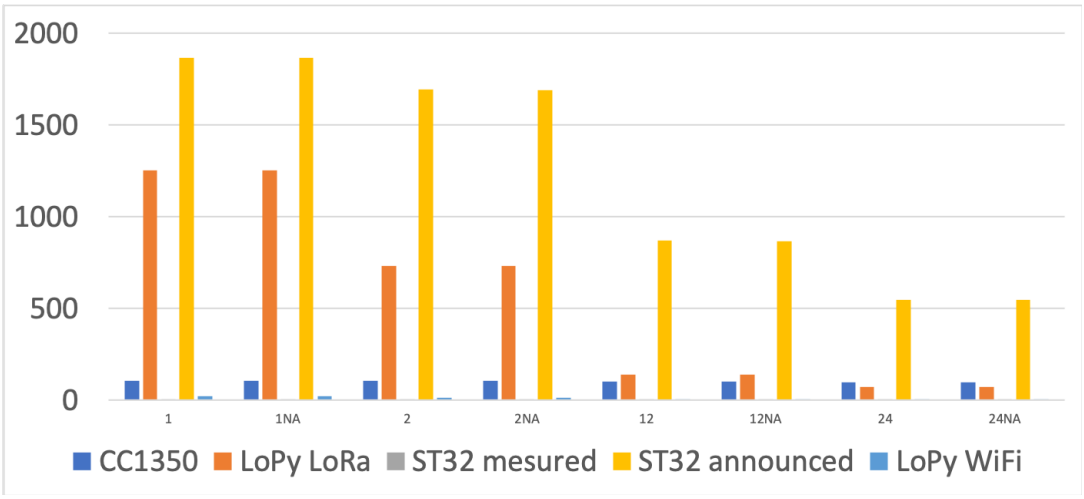


FIGURE 5.16 – Durée de vie des différents capteurs pour un nœud relais avec 5 nœuds isolés

Les cas présentés dans les différentes Figures 5.11, 5.12, 5.13, 5.14, 5.15, 5.16 et 5.17 sont ceux qui utilisent l'agrégation. Ceux qui n'utilisent pas l'agrégation ne sont pas assez pertinents pour

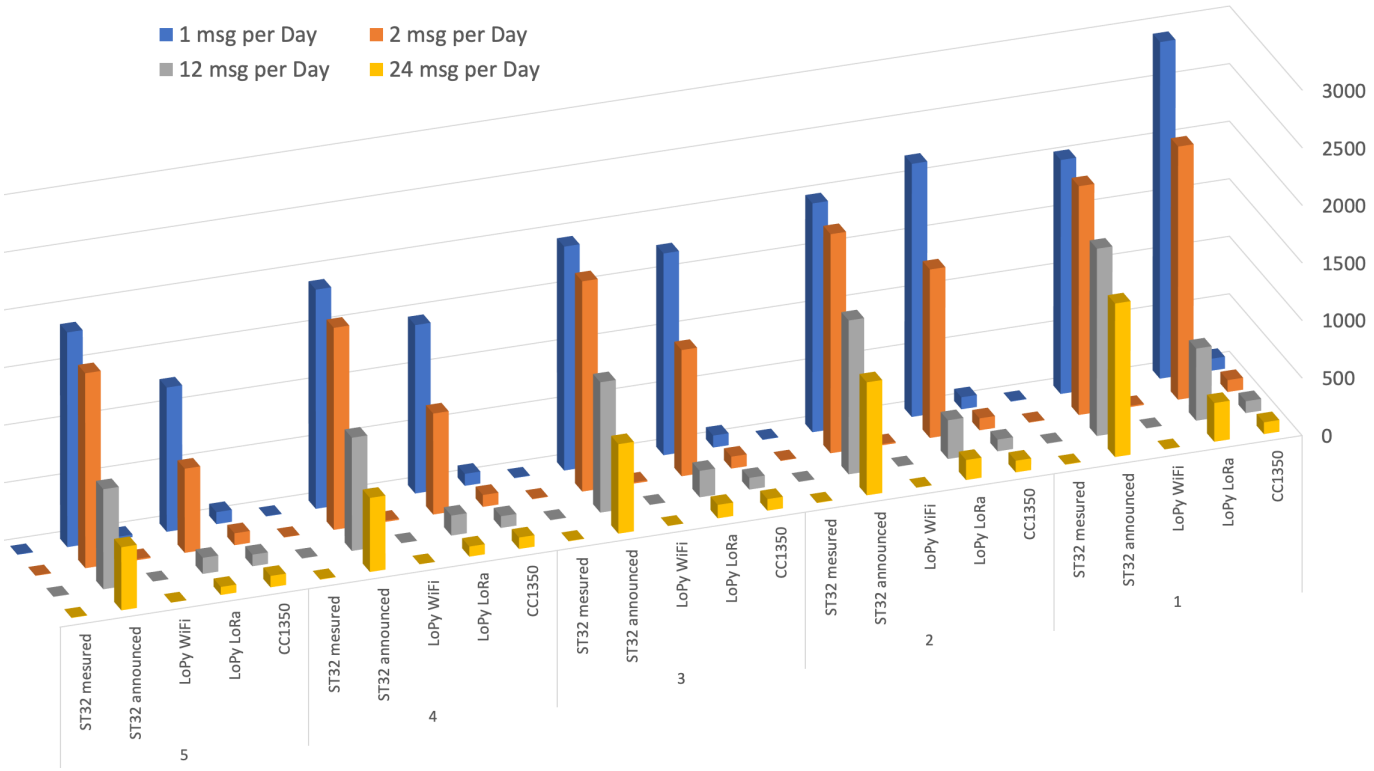


FIGURE 5.17 – Durée de vie d'un nœud relais avec 1 à 5 nœuds isolés

être inclus dans ce document. La différence entre ces deux modes est minime. La principale différence entre chaque cas peut être résumée plus particulièrement sur des périodes assez longues à la plateforme qui a la plus faible consommation dans un état de sommeil profond plus que la consommation d'émission. Ces résultats ont fait l'objet d'un papier et d'un poster publié en conférence internationale avec comité de lecture [Fla+20e] [Fla+20d].

5.7 Problèmes rencontrés

Les principaux problèmes rencontrés ont été de développer des éléments qui n'étaient pas disponibles dans la librairie de base. Seul le module AES était fourni, il a donc fallu développer les fonctions de recherche du nombre premier, de génération de la base et des autres éléments conformes au processus Diffie-Hellmann. Il est nécessaire d'avoir des nombres premiers suffisamment grands si nous voulions sécuriser correctement nos communications. Le problème vient donc essentiellement d'un principe mathématique car quand on parle d'un nombre premier suffisamment grand on parle d'un nombre avec un ordre de grandeur plus grand que ce que peut stocker un nombre entier. Un entier non signé sur notre architecture, est codé sur 32 bits il a donc une valeur maximale de **4294967295**. Il est nécessaire d'avoir une taille comprise entre 1024 et 2048 bits afin de sécuriser au mieux les échanges. Il faut alors utiliser la bibliothèque BIGNUM qui n'est pas disponible sur notre capteur. Il y a un autre problème qui est celui de la taille, le transfert de l'information à un autre capteur n'est pas possible au vu du débit disponible en LoRa.

5.8 Conclusion

Nous avons donc présenté la deuxième version de notre solution qui contrairement à la première version implémente la gestion et l'authentification de plusieurs nœuds isolés relayé par un nœud relais. Dans ce chapitre nous avons montré plusieurs pistes pour permettre l'authentification. Finalement nous avons choisi une méthode basée sur un échange Diffie-Hellman. Nous avons montré que l'impact du chiffrement est dérisoire par rapport au coût de communication en terme énergétique. La méthode Diffie-Hellman ne nous oblige pas à avoir un ou plusieurs messages supplémentaires, ainsi le coût de cette solution est infime. Nous avons présenté nos divers résultats de simulations autour de la consommation énergétique ainsi q'une étude de notre protocole sur plusieurs plateformes dotées de différents moyens de communication.

Dans le prochain chapitre nous intégrons l'aspect de dynamicité et de sûreté de fonctionnement au niveau des deux acteurs de notre solution.

Chapitre 6
Dynamicité et sûreté de fonctionnement (FT-LLWURP)

Résumé : *Auparavant si un message était perdu, un nœud disparaissait ou un nouveau nœud apparaissait. Cela entrainait des erreurs. C'est pourquoi dans ce chapitre nous présentons une version avec le support de la dynamicité et la sûreté de fonctionnement. Nous présentons les résultats de simulations qui montrent que les collisions sont réduites avec notre solution. Nous présentons également les expérimentations qui sont toujours en fonctionnement en 2021. Deux ans d'expérimentation sur la plateforme Bouygues Objenious où des pannes sur certains services comme celui des Join sont intervenues. Plusieurs nœuds ont été enlevés à la main pour simuler la disparition de nœuds isolés ou relais. Ces interventions n'ont eu aucun impact sur le fonctionnement des nœuds actifs.*

Publications :

1. Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. "A Fault Tolerant LoRa/LoRaWAN Relay Protocol Using LoRaWAN Class A Devices". In : *International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)*. Bari, Italy, 2020, p. 295-302. doi : 10.1007/978-3-030-61746-2_22. url : https://hal.univ-reims.fr/hal-02978424

Sommaire

Cette partie étudie la tolérance aux fautes et la dynamicité de la solution de passerelle LoRa/LoRaWAN présentée dans le précédent chapitre. L'étude proposée est fait selon deux axes : d'une part l'évolution du réseau de l'opérateur, et d'autre part le cycle de vie des end-devices, des passerelles LoRa/LoRaWAN et des nœuds isolés.

6.1 Fautes dans le système

Cette partie expose les différentes fautes qui peuvent arriver dans le réseau.

6.1.1 Perte d'un message

Un message peut subir un phénomène de perte dû à une collision, bruit radio ou décalage de timer.

Impact Ceci peut provoquer au premier abord une erreur quant à l'impossibilité de faire un appairage, de réaliser une récolte.

6.1.2 Evolution du réseau

Nœud isolé Si un nœud isolé vient à disparaitre :

Impact Il remplira l'espace pris par le relais pour être récolté alors qu'il n'existe potentiellement plus.

Si un nœud isolé apparait :

Impact Il est pris en charge déjà par l'algorithme grâce au multi-appairage.

Nœud relais Les pannes des nœuds peuvent aussi avoir un impact sur le réseau global. En effet si ce nœud est une passerelle :

Impact Les nœuds isolés ne seront jamais récoltés.

Si un nœud passerelle apparait :

Impact Cela ne change rien grâce à l'identification des payloads et la gestion du choix des passerelle par les nœuds isolés qui se font de manière arbitraire.

Nœud de l'infrastructure Le réseau peut subir des modifications au cours du temps comme des pannes temporaires d'une antenne relais, l'ajout d'une antenne ou l'indisponibilité temporaire d'une antenne en raison de perturbations électromagnétiques.

Impact Ces modifications de l'infrastructure peuvent avoir un impact sur le fonctionnement des différents nœuds : un *endnode* peut se retrouver isolé, ou un nœud isolé peut ne plus l'être et devenir potentiellement un *end-device*.

6.2 Solutions proposées

Nous énumérons ici les solutions proposées par rapport aux différents problèmes énoncés auparavant.

6.2.1 Perte d'un message

L'ensemble des messages est assuré d'être envoyé ou reçu par la mise en place de timer. Comme décrit par la suite :

- Si un message **Discover** est perdu, il sera renvoyé par le nœud isolé.
- Si un message **StartDiscovery** est perdu, il sera renvoyé par le nœud relais.
- Si un message **Accept** est perdu, il sera renvoyé par le nœud relais à la nouvelle réception d'un message **Discover**.
- Si un message **Register** est perdu, il sera renvoyé par le nœud isolé dans une limite de 5 avant de retourner dans l'état DISCOVER.
- Si un message **DataRequest** est perdu, il sera renvoyé par le nœud relais.
- Si un message **DataResponse** est perdu, il sera renvoyé par le nœud isolé à la demande du nœud relai. Le nœud isolé ne se mettra pas en low power mode avant un temps (dans nos simulations et expérimentations) de 300 secondes.

6.2.2 Evolution du réseau

6.2.2.1 Nouveau nœud

Nœud isolé Un message **StartDiscovery** est ajouté et est effectué par le relais après avoir reçu le join accept. Afin de signaler aux nœuds isolés qu'un relais est disponible. Ainsi un nœud isolé peut s'appairer dès qu'il détecte un relais grâce à ce message.

Nœud relais Un nouveau nœud relais déployé n'impactera pas le réseau. Il est prévu déjà depuis les versions précédentes des directives pour choisir le relais (basé actuellement sur des choix arbitraires). Un nœud isolé qui devient un nœud relais est une forme de nouveau nœud relais, ce changement est pris en charge dans le réseau par la partie "disparition d'un nœud". Celui-ci communiquera à son ancien relais à la prochaine récolte qu'il en est détaché.

Nœud de l'infrastructure Une nouvelle antenne va possiblement changer un nœud isolé en nœud relais, ceci est déjà couvert par le paragraphe précédent. Afin de permettre le changement le nœud isolé envoie un *Join Request* après une durée donnée (dans nos simulations et expérimentations arbitraires) après 64 *uplink*. S'il reçoit un *Join Accept*, le nœud devient un *end-device*. Au prochain **Data Request** de son ancien relais, le end-device enverra une dernière fois sa donnée via un message spécial pour indiquer qu'il n'a plus besoin d'être relayé. Comme l'indique la Figure 6.1.

6.2.2.2 Disparition d'un nœud

Nœud isolé Dans ce cas, les nœuds passerelle vont détecter qu'il n'y aucun retour de la part d'un nœud isolé avec lequel les passerelles sont attachées. Nous avons mis en place un compteur qui a pour valeur arbitraire 5 essais de récolte (arbitraire). Au-delà de ces 5 essais (arbitraires) infructueux, le nœud isolé est considéré comme en panne prolongée et donc il est retiré des nœuds pris en charge par la passerelle.

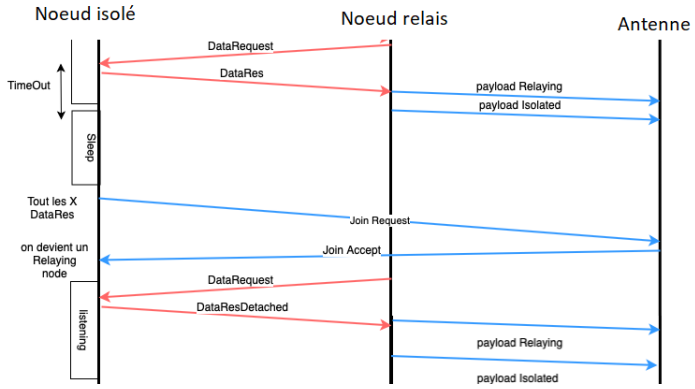


FIGURE 6.1 – Déroulement lorsqu'un isolated node devient un relais node

Nœud relais Le nœud isolé va détecter qu'il n'est plus récolté par un message **DataRequest** pendant un intervalle de 5 récoltes. Le nœud isolé va donc retourner à la phase de départ en effectuant un nouveau *Join request*. Dans le cas où il échoue, il va se considérer à nouveau comme un nœud isolé. Cependant pour éviter un grand nombre de messages **Discover**, il va envoyer un nombre aléatoire entre 1 et 5 (arbitraire). S'il ne reçoit pas de message **Accept** ce nœud va écouter sur la radio LoRa si un nouveau message **StartDiscovery** a été émis par une passerelle disponible. Ce nouveau message s'intègre à part entière dans la procédure.

Nœud de l'infrastructure Si un nœud antenne disparait, alors le relais ne verra plus d'accusé de réception en downlink. À partir de cet instant il va surcharger sa mémoire tant que c'est possible. Tout dépendant du support physique final. Dès que les accusés de réception sont à nouveau présents le nœud va envoyer tous les messages qui ont échoué auparavant.

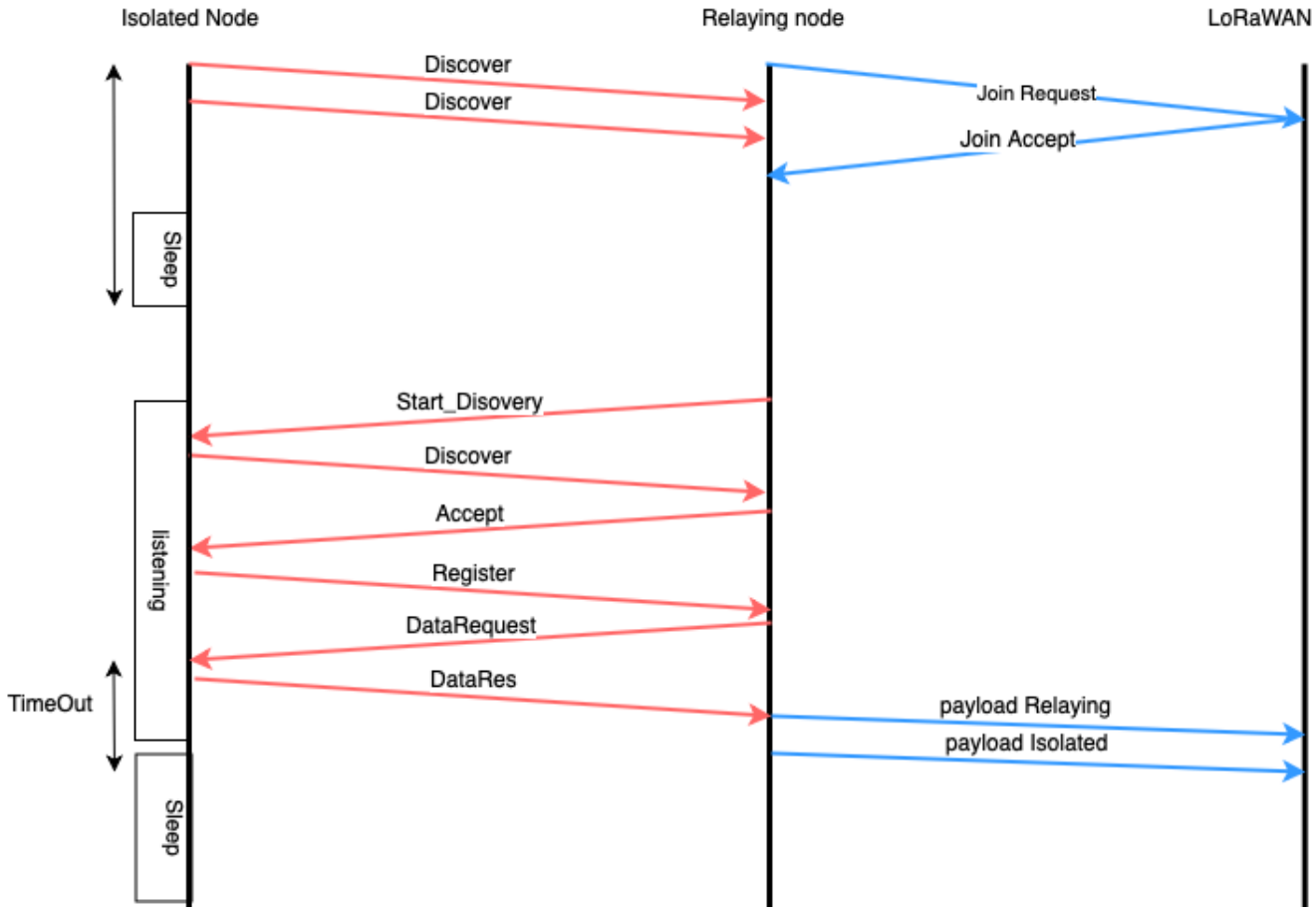


FIGURE 6.2 – Nouveau déroulement complet

6.3 Algorithme

Dans cette section nous mettons en avant les modifications apportées sur les différents algorithmes des nœuds isolés et relais.

Dans l'algorithme 6, le changement opère sur le nombre d'essais par slot pour récolter le nœud isolé et la vérification du statut de récolte pour chaque nœud isolé, si le nœud isolé n'a pu être récolté depuis un moment il est supprimé.

Du coté du nœud isolé, la modification se fait si il n'a rien reçu pendant son temps de récolte. Après un certain moment sans récolte alors il subit une ré-initialisation totale.

Algorithm 8 Algorithme nœud isolé 1-1

```
1: while (true) do
2:     flush_var()
3:                                  ▷ ——————————————————— phase d'appairage
4:     while (msg = udef) do
5:         sendLora(freq_listen, discover(udef, udef))
6:         (msg, t) = listen(freq_send, udef, candidate, timer + rnd())
7:     end while
8:     initVar(msg)
9:     sendLora(freq_send, pair(lgw, udef))
10:
11:                                 ▷ ——————————————————— Phase d'échanges
12:    while (lgw! = udef) do
13:        sleep(next_time)
14:        (msg, t) = listen(freq_send, lgw, data_request, timer)
15:        if msg! = udef then
16:            initVar(msg)
17:            sendLora(freq_send, date_response(lgw, local_data)
18:            is_harvested ← true
19:        else
20:            is_harvested ← false
21:            not_harvested ← not_harvested + 1 flush_var()
22:            if ( thennot_harvested > 5)
23:                (lgw ← udef)
24:                flush_var()
25:            end if
26:        end if
27:    end while
28: end while
```

Ce qui n'apparait pas par soucis de simplification d'écriture est tout ce qui concerne le traitement des messages à savoir s'ils sont conformes (bien formés) et que les bons sont attendus à chaque étape.

Algorithm 6 Algorithme lgw 1-n

```
1: LoRaWAN_join()
2: while (true) do
3:     while !(isTimeToCollect()) do
4:         if (in == udef) then
5:             (msg, t) = listen(freq_listen, udef, discover, timer)
6:         end if
7:         if (msg! = udef) then
8:             in ← msg.source
9:             init_var()
10:            sendLora(freq_send, candidate(in, freq_listen, slot, duration, freq_next, udef)
11:            (msg, t) = listen(freq_listen, in, pair, timer)
12:            if (msg == udef) then
13:                flush_var()
14:            end if
15:        end if
16:        if (in! = udef) then
17:            if (in NOT in tabRegisteredIN) then
18:                tabRegisteredIN.append(in)
19:                tabRegisteredData.append(−1)
20:                notHarvested.append(0)
21:            end if
22:            init_var()
23:            sendLora(freq_send, data_request(in, freq_listen, slot, duration, freq_next, udef)
24:            (msg, t) = listen(freq_listen, in, data_response, timer)
25:        end if
26:    end while
27:    if isTimeToCollect() then
28:        strTmpData ← ""
29:        for i= 0 ; i < tabRegisteredIN.length do
30:            init_var()
31:            for j= 0 ; h < maximum_try AND NOT this_one_harvested do
32:                sendLora(freq_send,data_request(tabRegisteredIN[i],
   freq_listen,slot,duration,freq_next,udef)
33:                (msg, t) = listen(freq_listen, tabRegisteredIN[i], data_response, timer)
34:                if (msg! = udef) then
35:                    tabRegisteredData[i] = msg.data
36:                    strTmpData ← str_tmp_data + ";" + tabRegisteredIN[i] + " : " +
   tabRegisteredData[i]
37:                    this_one_harvested ← true
38:                else
39:                    tabRegisteredData[i] − 1
40:                end if
41:            end for
42:        end for
43:        send_lora_data(id + " : " + local_data + ";" + strTmpData)
44:        VerifyNodesToKeep(tabRegisteredIN.length, tabRegisteredIN, notHarvested)
45:        flush_var()
46:    end if
47: end while
```

Algorithm 7 Algorithme lgw 1-n fonction VerifyNodeToKeep(length : entier , tabRegisteredData, tabRegisteredIN, notHarvested : tableau/vecteur d'entier)

```
1: i ← 0
2: terminated ← false
3: newtabRegisteredIN ← []
4: while i < length do
5:     if (tabRegisteredData[i] = −1) then
6:         notHarvested[i] ← notHarvested[i] + 1
7:     end if
8:     i ← i + 1
9: end while
10: i ← 0
11: while i < length do
12:     if (notHarvested[i] <= 4) then
13:         newtabRegisteredIN.append(tabRegisteredIN[i])
14:     end if
15:     i ← i + 1
16: end while
17: tabRegisteredIN ← newtabRegisteredIN
```

6.4 Simulation

Avant de valider notre solution en environnement réel, nous avons mené différentes campagnes de simulation. Les objectifs de ces campagnes sont :

1. Évaluer le surcoût par rapport à un comportement standard LoRaWAN
2. Évaluer le coût énergétique et l'impact des fautes sur celui-ci
3. Évaluer le coût en nombre de messages et l'impact des fautes sur celui-ci

La campagne de simulation a été réalisée dans l'environnement suivant :

— OMNeT++ 5

— Framework FLoRa

6.4.1 Framework FLoRa

FLoRa (Framework for LoRa) est un cadre de simulation permettant de réaliser des simulations de bout en bout pour les réseaux LoRa. Il est basé sur le simulateur de réseau OMNeT++ et utilise également des composants de INET.

FLoRa permet la création de réseaux LoRa avec des modules pour les nœuds LoRa, de passerelles et d'un serveur de réseau. La logique applicative peut être déployée sous forme de modules indépendants qui sont connectés au serveur de réseau. Le serveur de réseau et les nœuds prennent en charge la gestion dynamique des paramètres de configuration par le biais de l'Adaptive Data Rate (ADR). Enfin, les statistiques de consommation d'énergie sont collectées dans chaque nœud.

Ce framework a été développé par Mariusz Slabicki, Gopika Premsankar et Mario Di Francesco dans le cadre de leur étude portant sur l'impact de l'Adaptative Data Rate [SPD18].

6.4.2 Génération des simulations

Le framework permet une génération aléatoire en cercle autour d'une antenne et d'autre disposition. Celle utilisée est totalement aléatoire dans la zone déterminée.

Dans notre cas nous avons mis les paramètres suivants (à retrouver dans le fichier omnetpp.ini).

Chaque simulation est répétée avec une *seed* différente 5 fois. Le temps simulé est d'une journée. Le nombre de nœuds présents dans le système est de :

10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000

Les valeurs choisi pour le nombre de nœuds sont arbitraires. Une seule antenne est déployée dans une aire de 4000000 m^2. Ce qui correspond au cas d'usage présenté par Bouygues Télécom. On se base sur une génération de type Mersenne Twister.

Concernant notre protocole, côté relais :

— Le temps d'écoute des relais est de 10 minutes après chaque phase sleep

— Le nombre maximum de nœuds isolés appairés à un relais est de 5

— Le nombre maximal d'essais pour récolter un nœud isolé est de 5

— Il y une variation si l'agrégation est utilisée ou non

Ces paramètres ont été donnés comme seuil maximal par Bouygues Télécom. Concernant notre protocole, côté nœud isolé :

— Après 20 *uplink* envoyés auprès d'un relais, un *Join request* est effectué.

— Le nombre maximal pour découvrir un relais, s'enregistrer, ne pas être récolté auprès de lui est de 5.

— Le *hold timer* est de 300 secondes après collecte.

Les probabilités de tomber en panne sont évaluées à chaque action et ont pour valeurs arbitraires suivantes :

— Probabilité de tomber en panne : 0,2%

— Probabilité d'un nœud de réapparaitre : 2%

— Les probabilités sont tirées dans un espace uniforme

Ce qui représente 6840 simulations qui ont été calculées en 1 mois 24 jours et 59 min.

6.4.3 Représentation des échanges radio

Les échanges radio sont représentés par le framework par une interprétation du *SpreadFactor* (SF), de la puissance de transmission et de la distance qui sépare deux nœuds. Ceci est donc géré par le framework. La seule intervention faite est la répartition du SF qui est réparti aléatoirement entre un SF7 et un SF12 pour chaque nœud. Il est de même pour la puissance de transmission entre 7 et 14 dBm.

6.4.4 Exemple de simulations graphiques

Voici une capture d'écran (Figure 6.3) issue d'une simulation de 100 nœuds. On peut y observer la liaison entre l'antenne et le network server ainsi que l'architecture LoRaWAN dans son ensemble. On y voit également une répartition des nœuds homogène.

Le modèle énergétique est basé sur la documentation issue du SX1276. Il est intégré dans un fichier XML structuré comme présenté dans la Figure 6.4 :

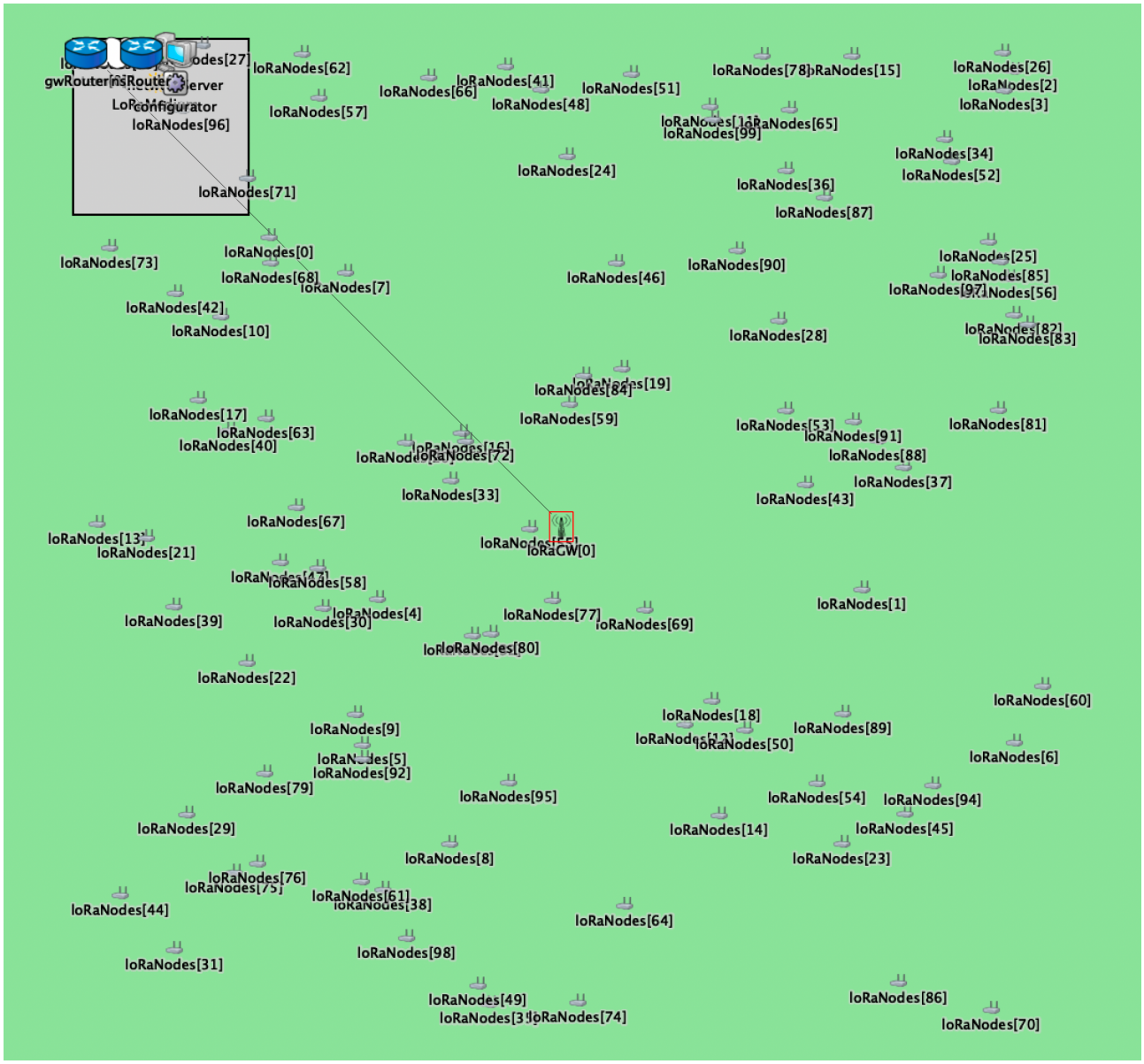


FIGURE 6.3 – Example de simulation

La tension d'alimentation est considérée de 3,3 Volt. La puissance de transmission diffère en même temps que sa consommation allant de 24 à 44 W.

Dans les simulations, les valeurs de **receiverReceivingSupplyCurrent, receiverBusySupplyCurrent, idleSupplyCurrent** sont multipliées par la valeur de **supplyVoltage** qui donne des mW.

Le framework gère l'intégralité de la gestion des modes low power et des modes émission/réception en terme de consommation d'énergie.

Les simulations intègrent le modèle via le fichier *energyConsumptionParameters.xml*.

```
<root>
        <supplyVoltage value="3.3"/>
        <receiverReceivingSupplyCurrent value="9.7"/>
        <receiverBusySupplyCurrent value="9.7"/>
        <idleSupplyCurrent value="0.0001"/>
                <txSupplyCurrents>
                        <txSupplyCurrent txPower="2" supplyCurrent="24"/>
                        <txSupplyCurrent txPower="3" supplyCurrent="24"/>
                        <txSupplyCurrent txPower="4" supplyCurrent="24"/>
                        <txSupplyCurrent txPower="5" supplyCurrent="25"/>
                        <txSupplyCurrent txPower="6" supplyCurrent="25"/>
                        <txSupplyCurrent txPower="7" supplyCurrent="25"/>
                        <txSupplyCurrent txPower="8" supplyCurrent="25"/>
                        <txSupplyCurrent txPower="9" supplyCurrent="26"/>
                        <txSupplyCurrent txPower="10" supplyCurrent="31"/>
                        <txSupplyCurrent txPower="11" supplyCurrent="32"/>
                        <txSupplyCurrent txPower="12" supplyCurrent="34"/>
                        <txSupplyCurrent txPower="13" supplyCurrent="35"/>
                        <txSupplyCurrent txPower="14" supplyCurrent="44"/>
                <txSupplyCurrents/>
<root/>
```

FIGURE 6.4 – Modèle énergétique dans *energyConsumptionParameters.xml*

6.5 Résultats des simulations

6.5.1 Résultats des simulations sur le nombre de messages

Les deux prochaines figures représentent le nombre de messages LoRa envoyés dans tout le système. La Figure 6.5, représente donc le nombre de messages envoyés dans un système allant de 10 à 1000 nœuds sans l'utilisation de notre algorithme avec des *uplinks* allant de 1 à 24 messages par jour.

Cette première donnée est représentée en bleu. La courbe verte décrit en pourcentage le taux de bonnes réceptions, c'est-à-dire sans collision. On y observe une augmentation exponentielle du nombre de messages entre chaque partie du graphique due au manque de coordination entre chaque nœud. Ceux qui sont isolés mais n'utilisent pas notre algorithme effectuent des ***Join Request*** en boucle. Le nombre de messages bien reçus par les nœuds avoisine les 82% au mieux. Sur la Figure 6.6, notre algorithme est utilisé par les nœuds du système. Le nombre de messages est en moyenne bien moins grand quand le nombre de messages par jour est faible. Cependant les mêmes résultats en nombre de messages sont atteints quand le nombre de nœuds est très grand. Le nombre de messages au lieu d'être constamment élevé sans l'utilisation de notre algorithme a une courbe exponentielle. Les meilleurs cas sont quand le nombre de messages par jour est faible. Ce qui correspond techniquement à l'usage final demandé par Bouygues Télécom. Pour le taux de réception réussi, les effets de la synchronisation se font sentir. En effet, là où nous avions sans notre algorithme 80% au mieux, nous sommes entre 87 et 95% de taux de réception. Malgré tout, cela représente un grand nombre de messages qui n'est pas reçu. Ce qui peut être le cas pour des nœuds isolés de tout nœud relais et d'une antenne.

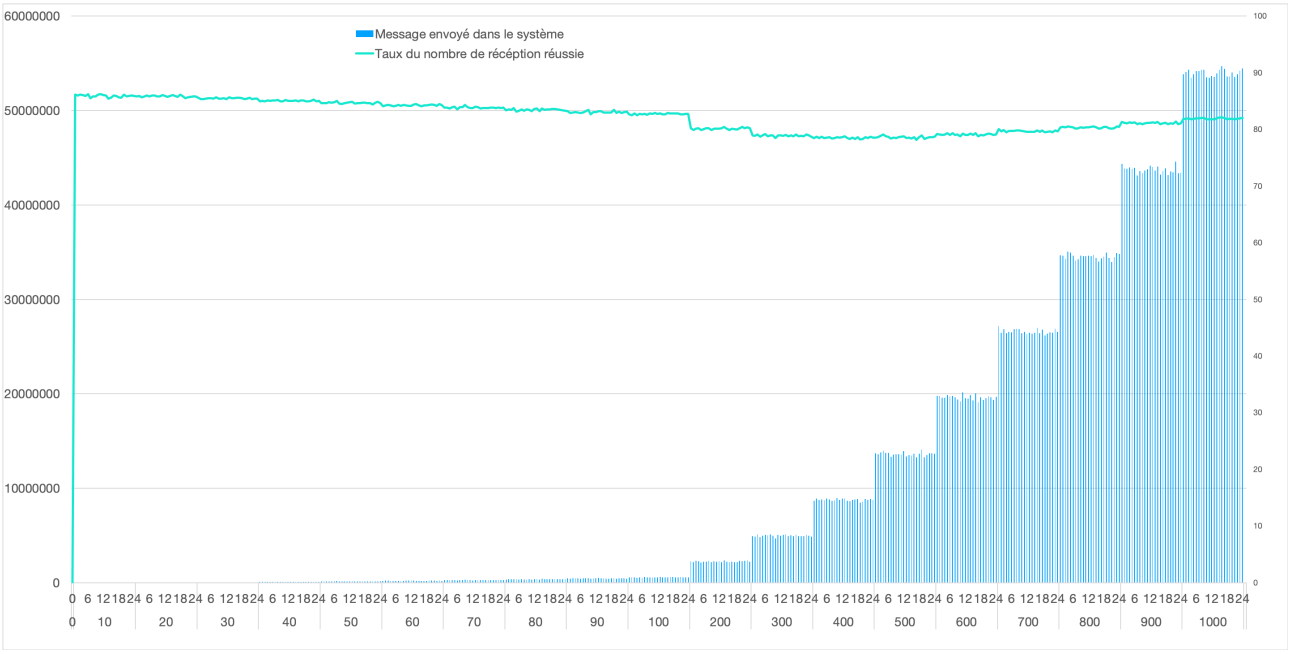


FIGURE 6.5 – Nombre de messages et taux de réception des messages dans le système en fonction du nombre de nœuds et de messages par jour, sans utilisation de notre algorithme

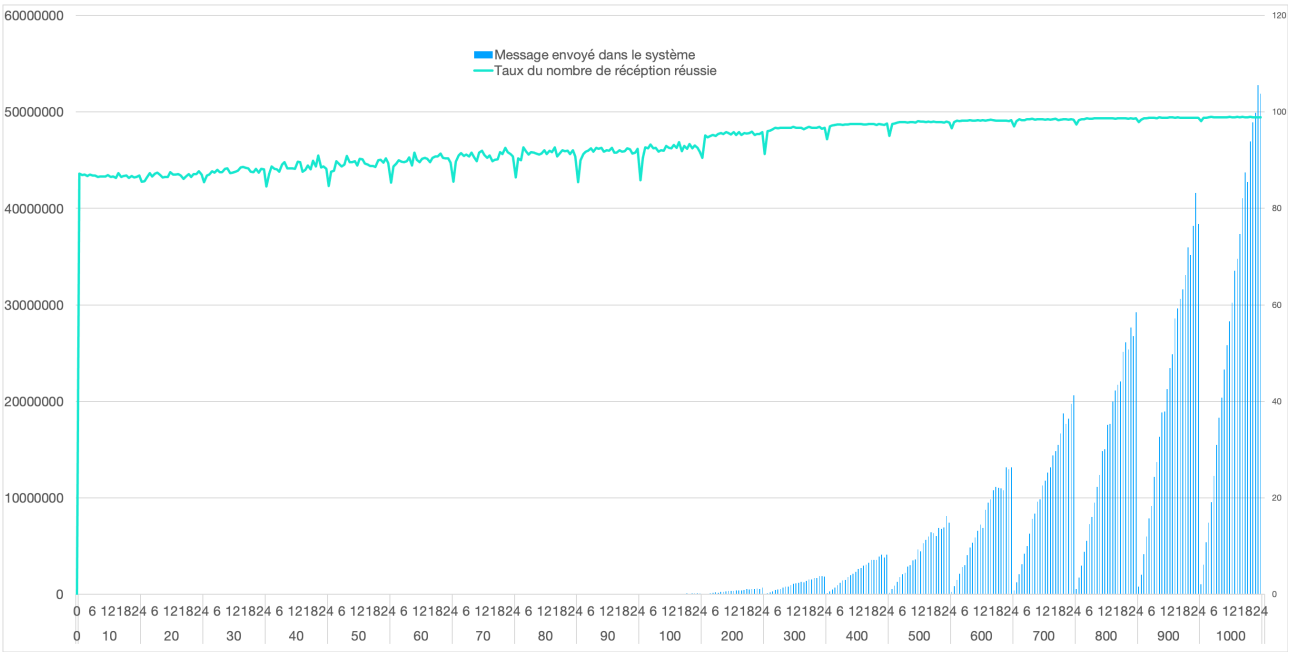


FIGURE 6.6 – Nombre de message et taux de réception des messages dans le système en fonction du nombre de nœuds et de messages par jour avec notre algorithme

6.5.1.1 Résultats des simulations sur la durée de vie

Les deux prochaines figures représentent la quantité d'énergie dépensée en moyenne par un nœud en Joule. Ce système respecte les paramètres décrits dans les sections précédentes. La Figure 6.7 représente donc l'énergie dépensée par un nœud classique (*end device*) n'utilisant donc pas notre algorithme.

La consommation oscille donc entre 1250J et 2800J. Là où lors de l'usage de notre solution nous avons dans le pire des cas 2600J consommés en un jour par un nœud. La consommation est beaucoup plus faible lorsque le nombre de messages à envoyer par jour est faible mais

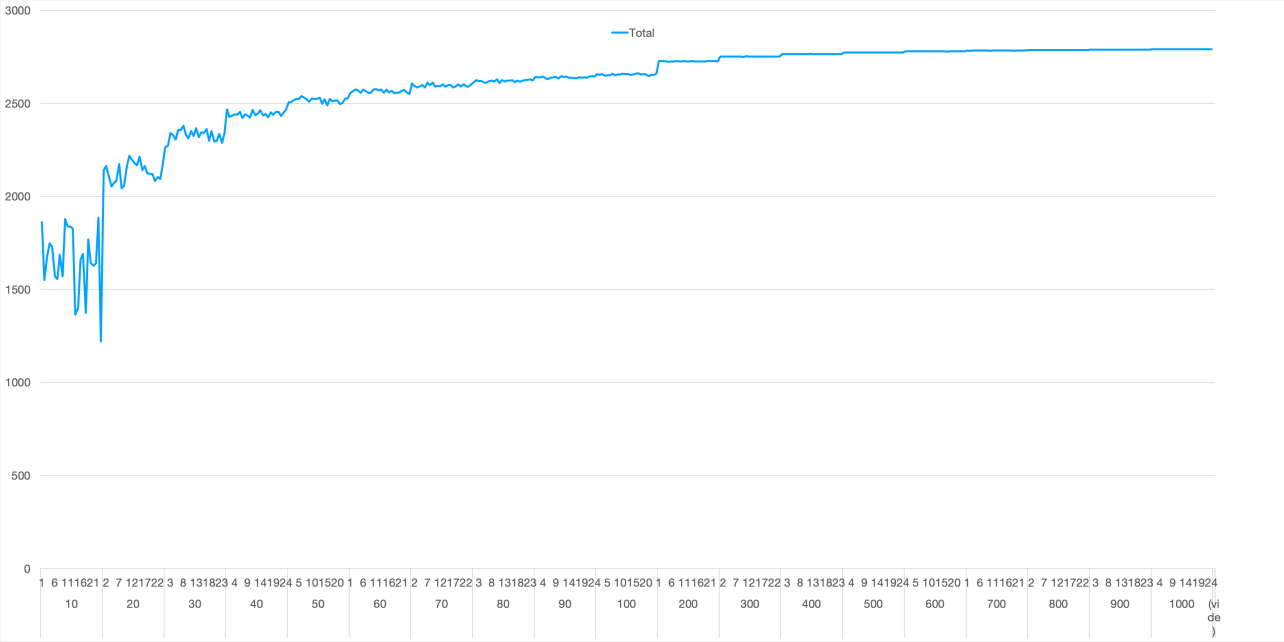


FIGURE 6.7 – Consommation energétique d'un nœud en fonction du nombre de messages par jour et du nombre de nœuds, sans notre algorithme

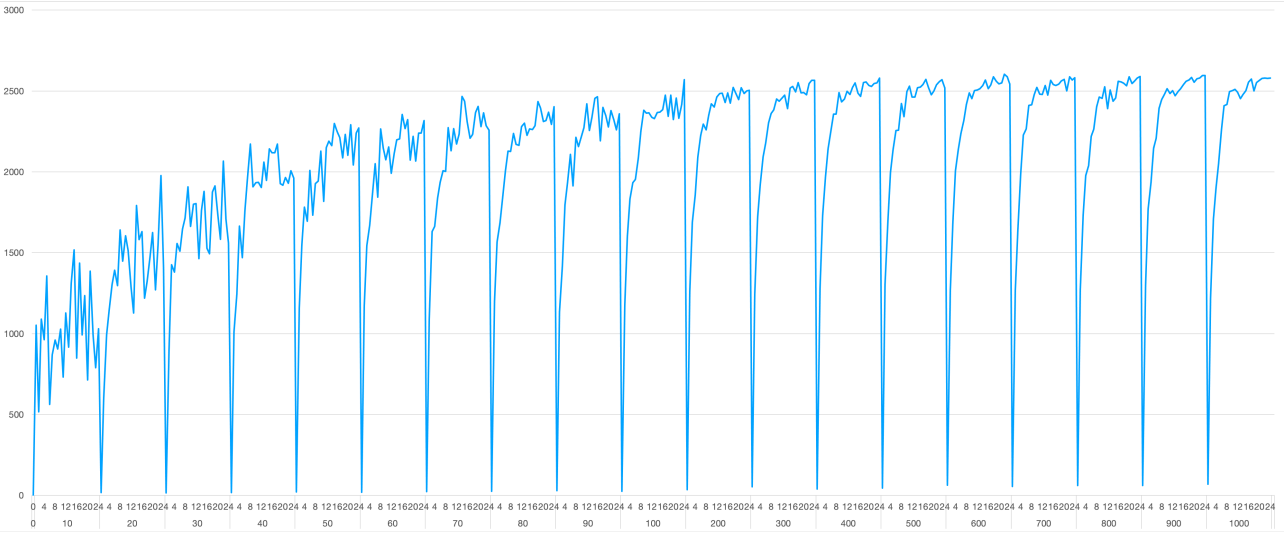


FIGURE 6.8 – Consommation énergétique d'un nœud en fonction du nombre de messages par jour et du nombre de nœuds, avec notre algorithme

beaucoup plus faible que lorsque notre solution n'est pas employée comme on peut le voir avec la Figure 6.8. Lorsqu'il s'agit d'envoyer 1 à 2 messages par jour la consommation est de l'ordre de **100-200J**. Ce qui est dû au processus de synchronisation et des mécaniques de low power qui sont exploitées. Le tableau 6.1 représente les écarts entre les deux approches ainsi que le gain apporté par notre solution.

6.5.1.2 Impact de l'agrégation sur les messages et l'énergie

Les résultats obtenus avec et sans l'utilisation d'agrégation sont trop peu différents sur les courbes pour en faire un graphique. Le nombre de messages demandés reste le critère le plus déterminant sur la consommation. Ce dernier impacte la durée durant laquelle le capteur est en mode économie d'énergie et donc la consommation mais pas suffisamment pour que l'agrégation

TABLE 6.1 – Consommation énergétique dans le meilleur et pire cas en Joule

	Sans notre solution	Avec notre solution	Gain
Consommation dans le meilleur des cas en J	1219,87	15,9292	98,6942 %
Consommation dans le pire des cas en J	2791,508547	2570,81	7,81 %
Consommation moyenne tout cas confondu en J	2584,48	2015,11	22,03 %

ait de l'impact sur celle-ci. Le nombre est certes plus élevé mais le taux de réception est sensiblement identique, la différence opère au centième de pourcentage. Le nombre de messages est quant à lui plus grand de 7% en moyenne.

Nous en déduisons que l'agrégation est négligeable compte tenu des autres dépenses énergétiques.

6.6 Expérimentation sur la plateforme SPOT

Pour mettre en place cette expérimentation nous avons déployé 3 capteurs avec comme rôle de départ un nœud qui fait le relais et deux autres nœuds qui sont isolés. Ils ont été mis en place le **22 juillet 2019** avec une première version logicielle de notre solution tolérante aux fautes. Celle-ci jusqu'au **28 juin 2021** est restée en activité et réagit à toutes les fautes prévues comme attendu.

La période d'expérimentation sur la plateforme est donc très longue : 23 mois. Les graphiques en vert représentent le nombre de messages en *uplink* et en bleu le nombre de messages en *downlink*.

Les nœuds sont programmés pour envoyer un message de données toutes les heures.

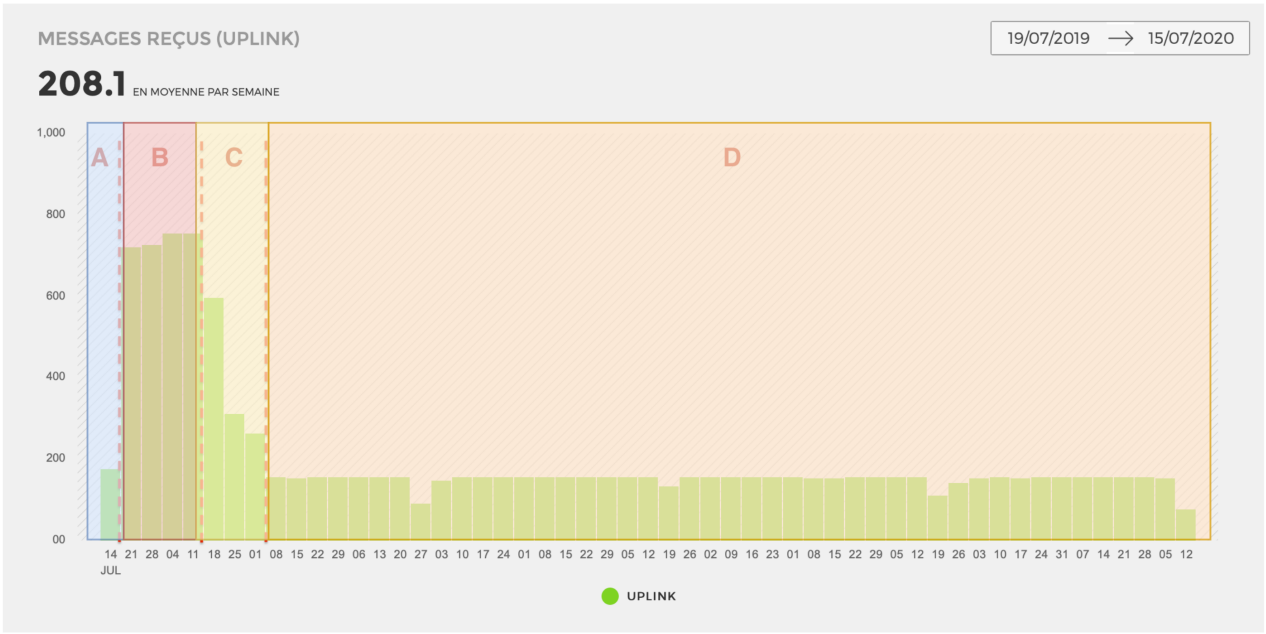


FIGURE 6.9 – Uplink et Downlink d'un nœud relais

La partie bleue/A de la figure représente le nombre de messages envoyés par le relais sans avoir un nœud isolé. C'est le comportement classique d'un *end-device*. Environ 200 messages par

semaine ou environ 1 message par heure.

La partie rouge/B représente le relais avec deux nœuds isolés. Cela explique l'augmentation du nombre de messages qui est de 700. On observe une perte de messages éventuellement due à des collisions radio.

La partie jaune/C est le moment où un nœud isolé a réussi à devenir un relais. Il reste un nœud isolé encore couplé au relais.

La partie orange/D est un retour à la phase bleue/A où les deux nœuds isolés sont devenus des nœuds relais. Cela explique la baisse significative à environ 200 messages par semaine.

Ces graphiques (Figures 6.11 et 6.9) montrent que le fait qu'un nœud isolé devienne un relais n'a pas d'impact sur le fonctionnement du nœud de relais de départ.

Les deux autres graphiques concernent les deux nœuds qui sont isolés depuis le début et qui deviennent des nœuds classiques (4 *end-devices*) vers septembre 2019. Le deuxième nœud isolé a commencé à devenir un *end-device* vers le début du mois de septembre à la vue de son nombre de messages plus important vis-à-vis du premier nœud isolé qui lui en a moins.

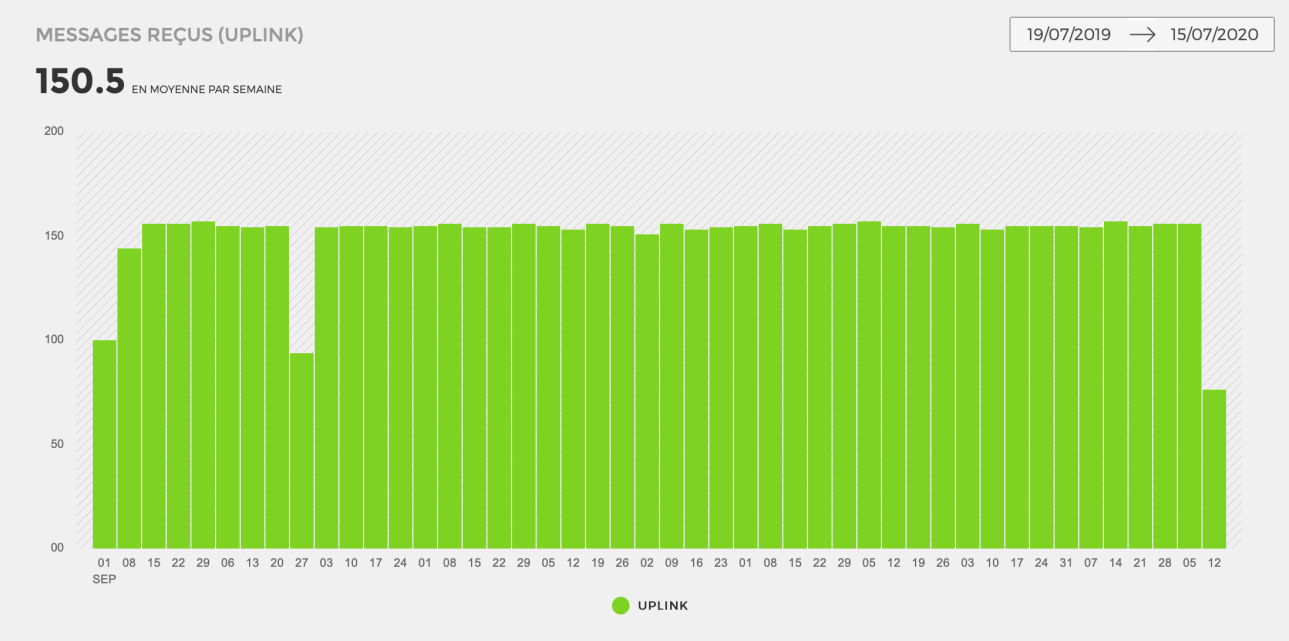


FIGURE 6.10 – Uplink et Downlink d'un premier nœud qui a démarré isolé

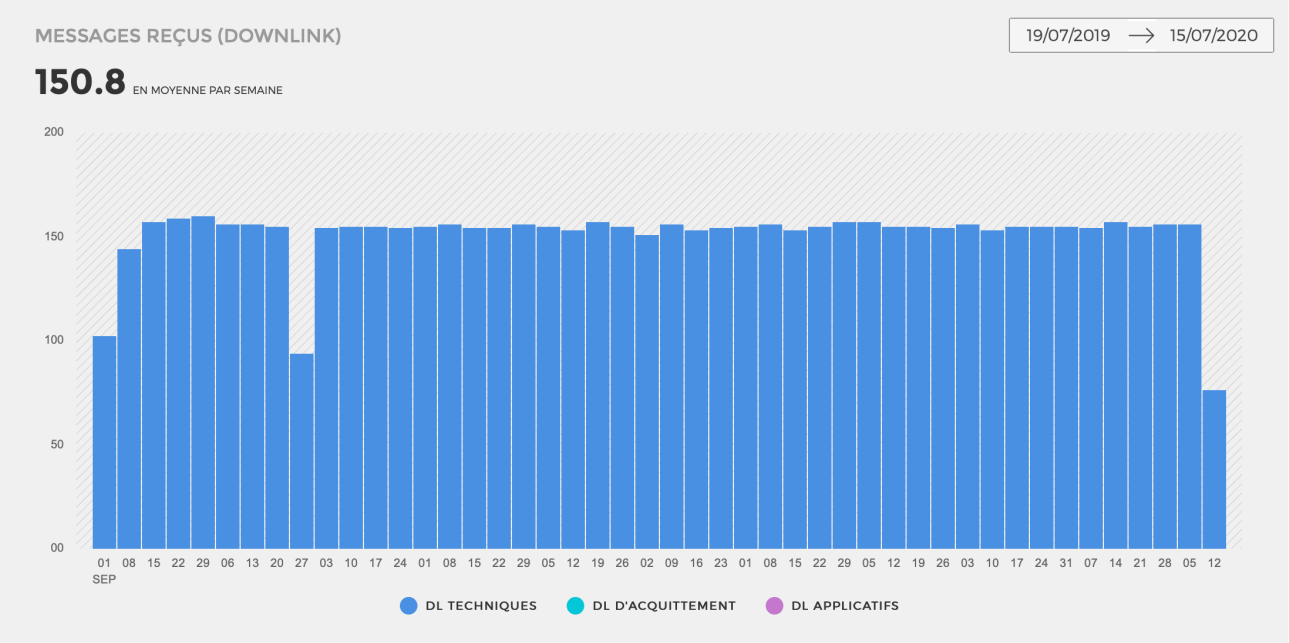


FIGURE 6.11 – Downlink d'un premier nœud qui a démarré isolé

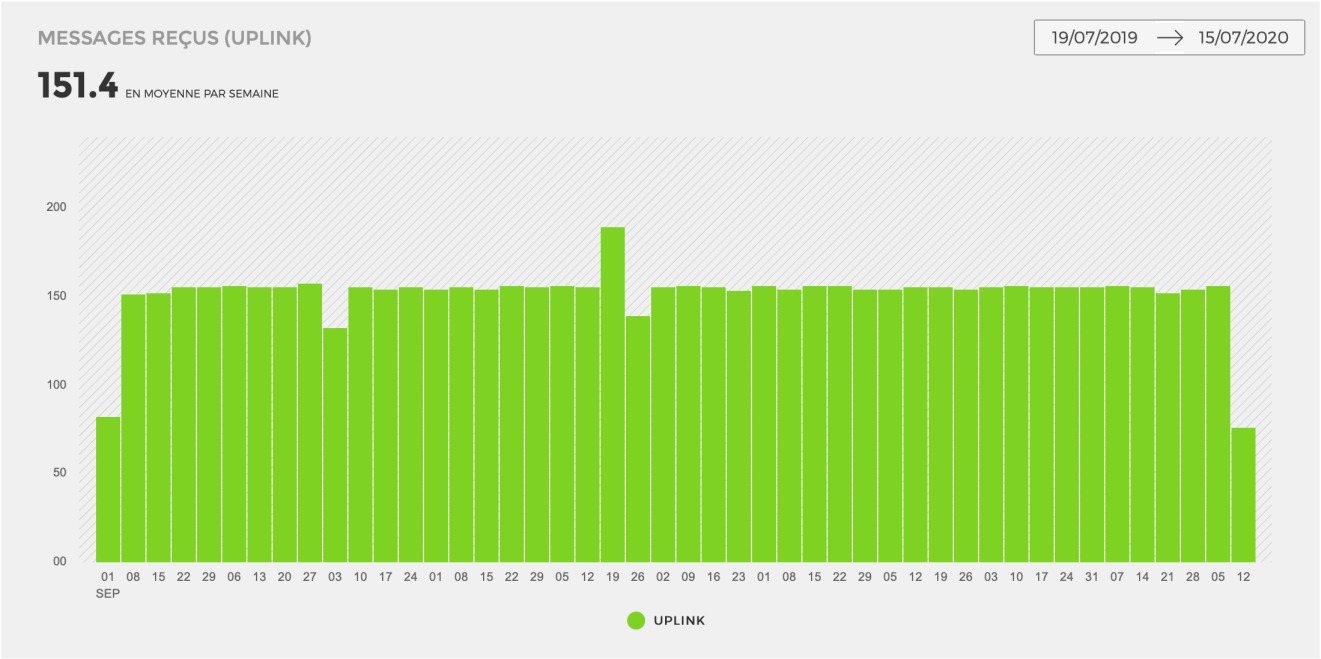


FIGURE 6.12 – Uplinkd'un deuxième nœud qui a démarré isolé

L'ensemble de ces travaux autour des adaptations de notre travail pour rendre notre algorithme apte à gérer des comportements dynamiques du réseau ont fait l'objet d'un papier publié en conférence internationale avec comité de lecture [Fla+20a].

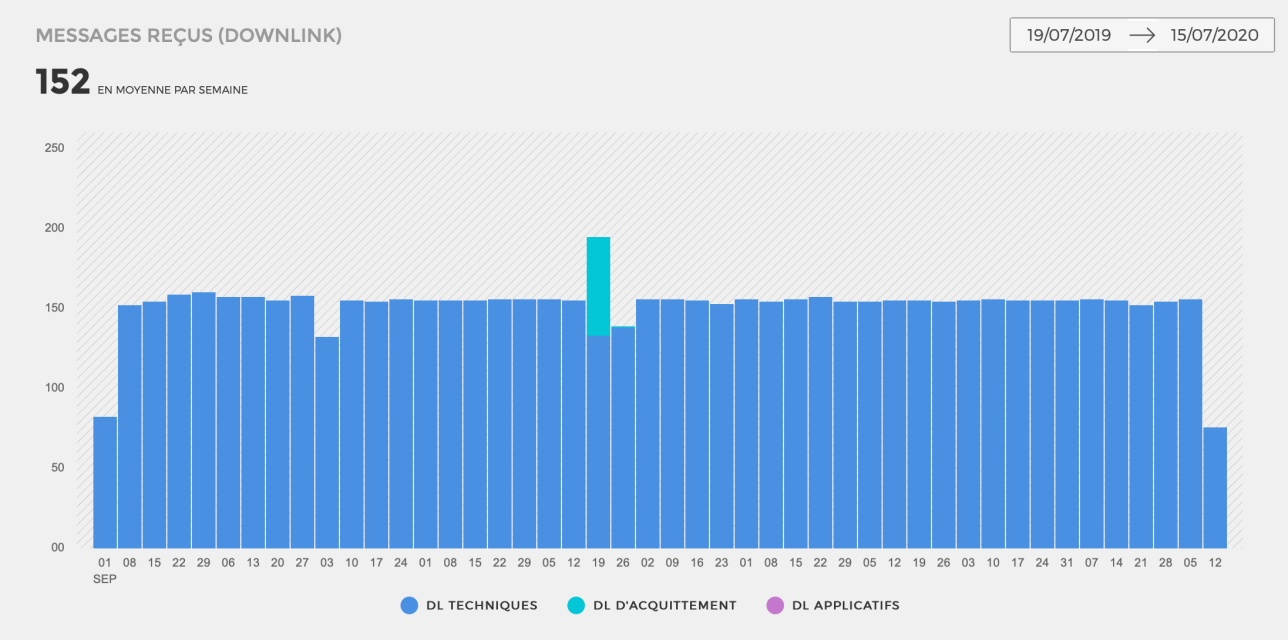


FIGURE 6.13 – Downlink d'un deuxième nœud qui a démarré isolé

6.7 Conclusion

Dans ce chapitre nous avons présenté la version qui supporte la dynamicité et la sûreté de fonctionnement basée sur la solution sécurisée. Nous avons présenté les résultats de simulations basés sur le framework FLoRa qui facilite le calcul de collisions mais aussi la mesure de dépense énergétique par rapport à un comportement normal. Le taux de réception est mesuré nativement, les simulations montrent que les collisions sont réduites avec notre solution en passant de 80% de taux de bonnes réceptions à 95%. La consommation, elle, est 22% meilleur en moyenne et au mieux 98%. Nous avons présenté les expérimentations qui ont démarré en juillet 2019 et sont toujours en fonctionnement en 2021. Deux ans d'expérimentation sur la plateforme Bouygues Objenious où des pannes sur certains services comme celui des Join sont intervenues. Plusieurs nœuds ont été enlevés à la main pour simuler la disparition de nœuds isolés ou relais. Ces interventions n'ont eu aucun impact sur le fonctionnement des nœuds actifs.

Chapitre 7
Maillage (MESH-FT-S-LLWURP)

Résumé : *Dans ce chapitre nous présentons une amélioration par rapport à la solution précédente qui n'était qu'une solution un saut. Maintenant nous intégrons un aspect maillé/multi-sauts. Nous avons mis en place une simulation algorithmique afin d'estimer le temps de convergence en fonction du nombre de nœuds présents dans le système. Ensuite nous parlons de la mise en place de l'implémentation de cette nouvelle version dans l'environnement STM32. Pour comparer à la précédente version de la solution, nous utilisons le même environnement de simulation FLoRa. Nous en sortons des résultats autour de la consommation énergétique, du nombre de messages et du pourcentage de bonnes réceptions.*

Sommaire

7.1 Introduction

Dans ce chapitre, nous présentons l'algorithme de maillage sans distinction comme auparavant. C'est à dire qu'il y a un nœud initiateur et des liaisons entre nœuds qui se créent. Par la suite nous parlons de l'adaptation du principe algorithmique avec notre approche de relais-nœuds isolés via des chronogrammes. Enfin nous parlons des simulations et des différentes comparaisons.

7.2 Algorithme

Nous allons donc présenter l'algorithme de maillage sur lequel nous nous sommes basés.

7.2.1 Principe

L'initiateur est le nœud 0, il est son propre père. L'identifiant de chaque nœud définit l'ordre des envois et réceptions. Par exemple le nœud 0 parle en premier, suivi du nœud 1, 2, 3 et 4. Les identifiants sont aléatoirement répartis. Le nœud initiateur est le premier à diffuser un message de construction avec les informations nécessaires autour de lui, à savoir : l'arbre couvrant actuel et les slots pris. Si un nœud n'a pas de père et qu'il reçoit un message de construction. Il considère que l'émetteur de ce message est son père. Une fois le message de construction diffusé. Les nœuds vont émettre un message de données à leur père. Toujours avec le même principe mais cette fois-ci dans l'ordre inverse le nœud 4 en premier (s'il a un père), 3, 2, 1 etc. Chaque nœud met à jour son arbre en local et ainsi se termine un **round**. L'algorithme continue jusqu'à avoir le réseau entièrement couvert.

7.2.2 Constantes et conditions

Voici les différentes conditions et constantes présentes dans notre algorithme :

- Nombre de nœuds maximum dans le système, noté $N_{\max}$
- Identifiant des nœuds allant de 0 à $N_{\max}$ -1, noté id_n
- Le nœud initiateur choisi arbitrairement, noté nœud X ?
- Le seuil de différence de round, noté S

7.2.3 Variables

Les différentes variables utilisées sont celles-ci :

- $Round$: round actuel du nœud
- id_n : identifiant du nœud
- $Slots$: structure avec deux variables :
  1. tableau de slot de taille $N_{\max} - 1$
  2. size
- TX_{delay} : temps associé à chaque nœud pour émettre durant ce tour
- $data$: données du nœud
- i, j : variables d'itérations
- $NextSlot$: temps dans lequel le prochain round aura lieu
- $Father$: identifiant du nœud parent du nœud actuel
- $pairedW$: tableau d'identifiants des nœuds dont il est le père
- RX_{delay} : le temps minimum pour recevoir un message

7.2.4 Messages

Chaque message est constitué d'un timestamp basé sur le numéro du dernier tour connu par le nœud émetteur, puis de l'identifiant de l'émetteur, l'identifiant du destinataire, du tableau des slots, le temps d'émission pour un nœud et de la donnée du nœud ainsi que le temps avant le prochain tour. Soit un message est structuré comme ceci :

$$M = Round; id_s; id_d; Slots; TX_{\text{delay}}; NextSlot; data$$

Le champ data est structuré ainsi :

$$data = data_I; id_0; data_0; id_1; data_1; id_2; data_2;$$

7.2.5 Fonctions

Les différentes fonctions non décrites entièrement sont les suivantes :

- $Sleep(T)$: S'endort pour un temps X
- $Listen(void||T)$: Sans paramètre : écoute via le medium jusqu'à réception, traite le message et retourne en écoute. Avec T en paramètre : écoute durant le temps T avant de s'arrêter
- $Send(data)$: Envoie le paramètre data via le medium
- $PrepareFrame(Round, id_n, id_dest, Slots, TX_{\text{delay}}, NextSlot, data)$: Retourne un message formaté avec les données nécessaire
- $UpdateData(data)$: Met à jour le tableau local des slots et les datas en fonction d'un message reçu
- $getId()$ Retourne l'identifiant unique du nœud
- $Init()$ Initialise : $Slots$, $NextSlot$, TX_{delay}, $NextSlot$, $paired$
- $OnListenDone(M_r)$: traite le message reçu
- $getRole()$: Détermine si le nœud est un nœud racine

7.2.6 Algorithme formel

Tous les blocs de l'algorithme sont décrit ici.

Algorithm 9 Ad-hoc initiateur

$end \leftarrow 0$
$Round \leftarrow 1$
$id_n \leftarrow getId()$
$M \leftarrow NULL$
$M_r \leftarrow NULL$
$pairedW \leftarrow NULL$
$father \leftarrow id_n$
$Init()$
$j \leftarrow N_{\max}$
while $end \neq 1$ **do**
  $M \leftarrow PrepareFrame(Round, id_n, NULL, Slots, TX_{\text{delay}}, NextSlot, data)$
  $Send(M)$
  for $k \leftarrow 1$ to j **do**
    $M_r \leftarrow Listen(TX_{\text{delay}} * 2)$
    $OnListenDone(M_r, j)$
  end for
  $Sleep(NextSlot)$
  $Round \leftarrow Round + 1$
end while

Algorithm 10 OnListenDone : M, j

Require: $M \neq NULL$
if $M.id_d = id_n$ AND $M.id_d$ is not in $pairedW$ **then**
  $pairedW.add(M.id_s)$
  $UpdateData(M)$
  $j \leftarrow Slots.size$
end if

Algorithm 11 OnListenDone : M

Require: $M \neq NULL$
if $|M.Round - Round| > S$ **then**
  $Round \leftarrow M.Round$
end if
if $M.id_s = father$ OR $M.id_s$ is in $pairedW$ **then**
  $UpdateData(M)$
end if

Algorithm 12 Ad-hoc non initiateur

```
end ← 0
Round ← 1
id_n ← getId()
M ← NULL
M_r ← NULL
pairedW ← NULL
father ← NULL
Init()
while end ≠ 1 do
    while father ≠ NULL do
        M_r ← Listen(RX_delay)
        if M_r ≠ NULL then
            if father = NULL then
                father ← M_r.id_s
                UpdateData(M_r)
                Sleep(NextSlot)
                Round ← Round + 1
            end if
        end if
    end while
    for k ← 0 to id_n do
        if k is in Slots.id then
            M_r ← Listen(TX_delay * 2)
            OnListenDone(M_r)
        end if
    end for
    M ← PrepareFrame(Round, id_n, father, Slots, TX_delay, NextSlot, data)
    Send(M)
    Sleep(NextSlot)
    Round ← Round + 1
end while
```

Algorithm 13 Full ad-hoc

```
Init()
root ← getRole()
while end ≠ 1 do
    if root then
        M ← PrepareFrame(Round, id_n, NULL, Slots, TX_delay, NextSlot, data)
        Send(M)
        for k ← 1 to j do
            M_r ← Listen(TX_delay * 2)
            if M_r ≠ NULL AND M.id_d = id_n AND M.id_d is not in pairedW then
                pairedW.add(M.id_s)
                UpdateData(M)
                j ← Slots.size
            end if
        end for
    else
        while father ≠ NULL do
            M_r ← Listen(RX_delay)
            if M_r ≠ NULL then
                if father = NULL then
                    father ← M_r.id_s
                    UpdateData(M_r)
                    Sleep(NextSlot)
                    Round ← Round + 1
                end if
            end if
        end while
        for k ← 0 to id_n do
            if k is in Slots.id then
                M_r ← Listen(TX_delay * 2)
                if M_r ≠ NULL AND |M.Round − Round| > S then
                    Round ← M.Round
                end if
                if M_r ≠ NULL AND M.id_s = father OR M.id_s is in pairedW then
                    UpdateData(M)
                end if
            end if
        end for
        M ← PrepareFrame(Round, id_n, father, Slots, TX_delay, NextSlot, data)
        Send(M)
    end if
    Sleep(NextSlot)
    Round ← Round + 1
end while
```

7.3 Simulations de l'algorithme

Afin de déterminer un temps de convergence vers une couverture du réseau. Nous avons réalisé des simulations purement algorithmiques sur OMNeT++ de la même manière qu'auparavant le nombre de nœud va varier par pas de 1 de 5 à 1000 nœuds. Leurs positions et identifiants sont aléatoirement générés.

Les résultats sont les suivants, la première Figure 7.1 représente le nombre de rounds nécessaires pour converger vers une couverture totale de l'arbre. On observe via cette première figure un allure logarithmique. Cela monte très vite vers 3 rounds nécessaires pour ensuite commencer à réduire. Nous observons donc que le temps de convergence est court. Afin d'éclaircir la tendance

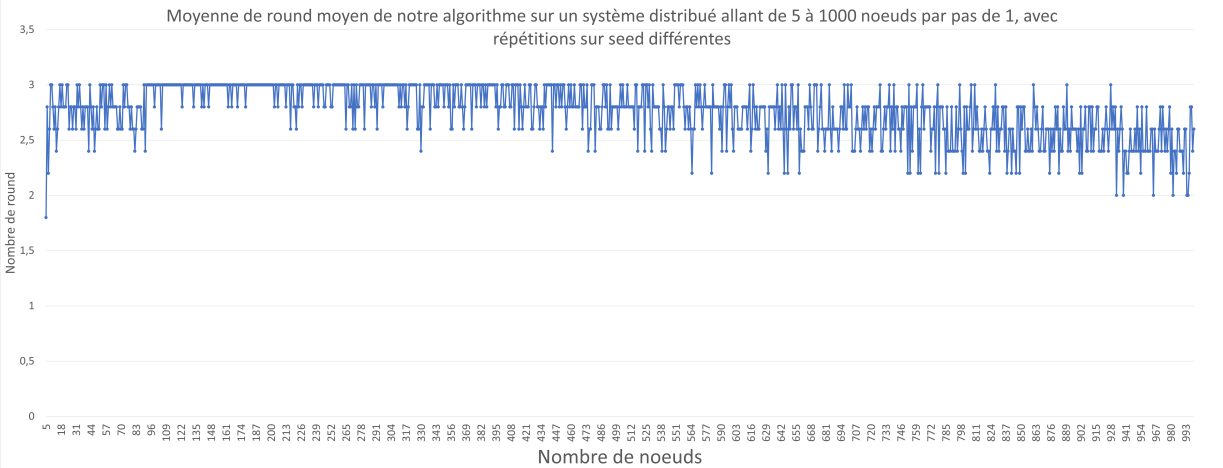


FIGURE 7.1 – Nombre de rounds pour converger à la couverture d'un réseaux de taille variant de 5 à 1000 nœuds par pas de 1

nous avons rajouté à cette précédente figure l'intervalle de confiance des valeurs ainsi que une courbe de tendance dans la Figure 7.2. On peut donc conclure que l'algorithme proposé a une

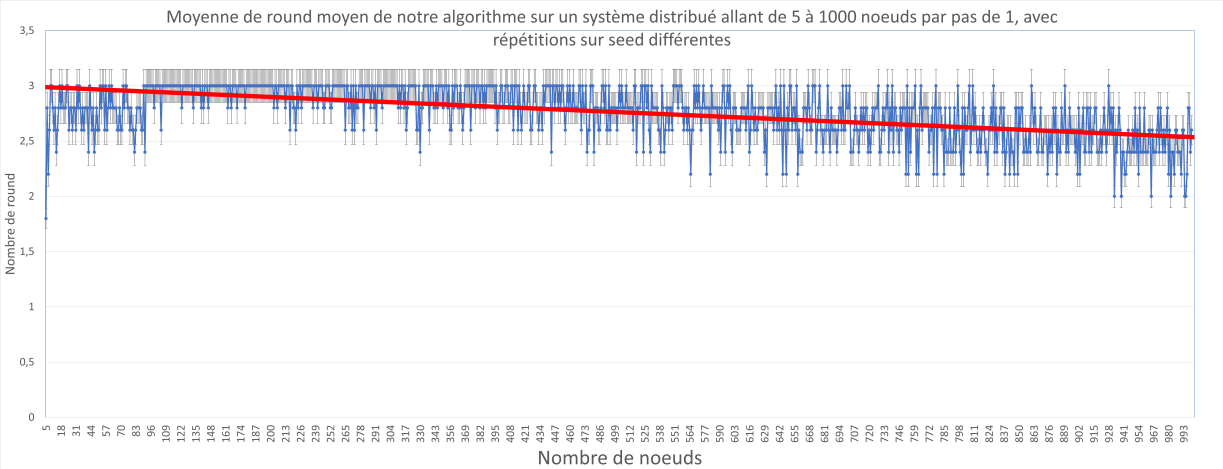


FIGURE 7.2 – Nombre de rounds pour converger à la couverture d'un réseau de taille variant de 5 à 1000 nœuds par pas de 1 avec la courbe de tendance

convergence rapide sur la couverture d'un réseau. Cependant il faut évaluer l'impact d'une couche physique sur cet algorithme et l'adapter.

7.4 Implémentation STM32

Pour implémenter cet algorithme dans une solution et un environnement LoRa et LoRaWAN. Nous avons fait un ensemble de choix qui vont être expliqués dans cette section.

7.4.1 Principe

Pour être en accord avec notre algorithme, une phase d'émission ou d'appairage va être résumée comme la synchronisation à double poignée de mains et la récolte se fera toujours à l'initiative du relayeur. La donnée remontera donc bien des nœuds les plus éloignés en premier pour remonter jusqu'au nœud relais initial. Ainsi pour permettre un appairage d'un nœud isolé avec un autre nœud isolé qui auparavant a été appelé Full Isolated node ou nœud complètement isolé. Après la récolte du nœud isolé, au lieu de s'endormir jusqu'au prochain slot, une valeur arbitraire, par exemple de 10 minutes, est choisie pour réaliser des appairages suivi de 5 minutes de récoltes. Par exemple, si le prochain slot est dans 1 heure, alors le nœud va se réveiller dans 45 minutes, envoyer un **Start Dicovery** et effectuer, si possible, des appairages. S'il a un nœud isolé attaché à lui, il va le récolter sur les 5 minutes allouées. Ensuite, il entre en phase d'écoute pour être récolté par son relais. Pour transmettre l'information d'un relais de nœud supplémentaire à son relais, le nœud isolé va envoyer dans un champ du message du **Data request** le nœud du prochain nœud à récolter. Ainsi le relais va le considérer et envoyer un **Data request** concernant ce nœud à son nœud isolé de premier niveau. Celui-ci va envoyer la donnée du nœud concerné et s'il y a un autre nœud, il rempli à nouveau le champ avec l'identifiant du prochain nœud.

7.4.2 Chronogrammes

Le déroulement est ainsi expliqué grâce au chronogramme de la Figure 7.3 qui représente la première partie qui récapitule le travail déjà effectué. La deuxième Figure 7.4 représente la suite avec l'appairage et le relayage.

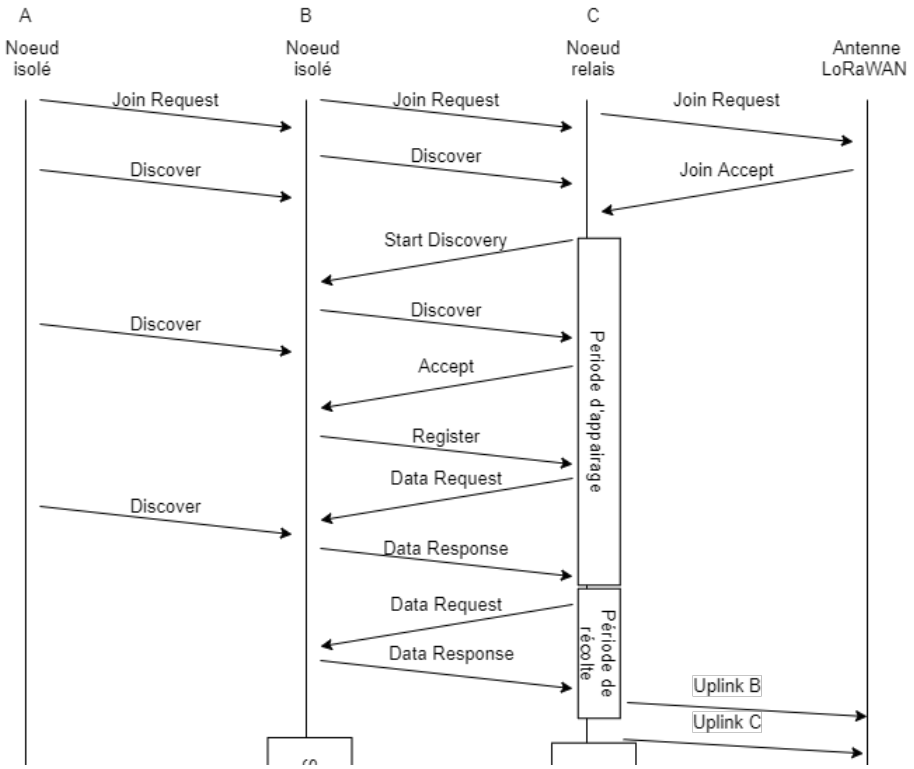


FIGURE 7.3 – Chronogramme exemple en 1/1/1 première partie

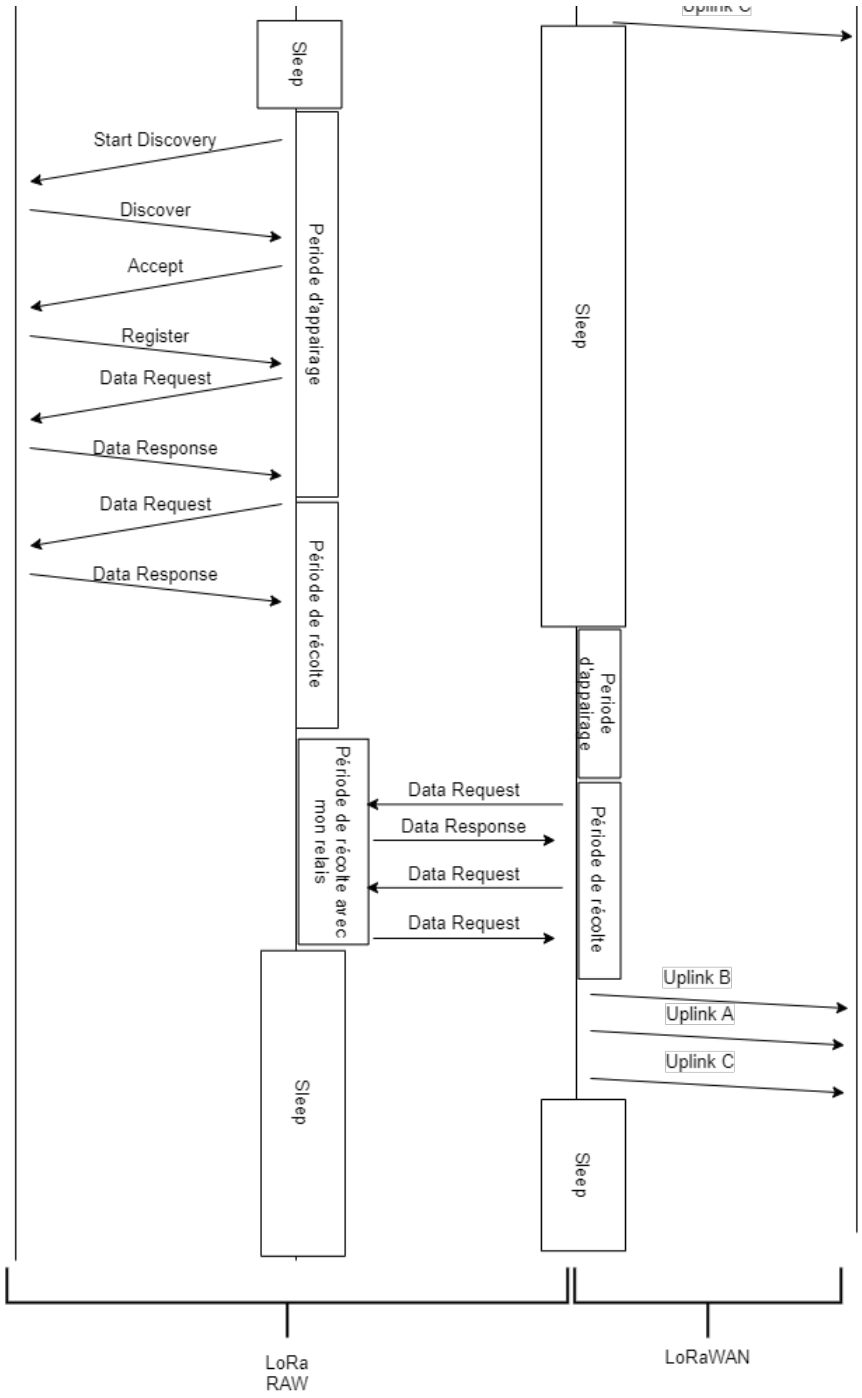


FIGURE 7.4 – Chronogramme exemple en 1/1/1 deuxième partie

7.5 Simulations FLoRa

Pour rappel ici l'environnement de simulation est le même que celui utilisé dans le dernier chapitre à la section 6.4. Nous avons les mêmes paramètres, sur le nombre de nœuds, le modèle de construction des graphes et le modèle de consommation énergétique. Comme l'environnement de simulation est le même, nous présentons les résultats et la comparaison avec les précédents résultats afin d'établir une différence, de dégager une tendance.

7.5.1 Résultats sur la consommation énergétique

La première partie des résultats est celle sur la consommation énergétique, les deux prochaines courbes concernent la consommation énergétique moyenne d'un nœud au sein du réseau en fonction du nombre de messages à envoyer par nœud par jour, ainsi que le nombre de nœuds présents dans le système. Ici il n'y a pas de comparaison directe avec notre version du précédent

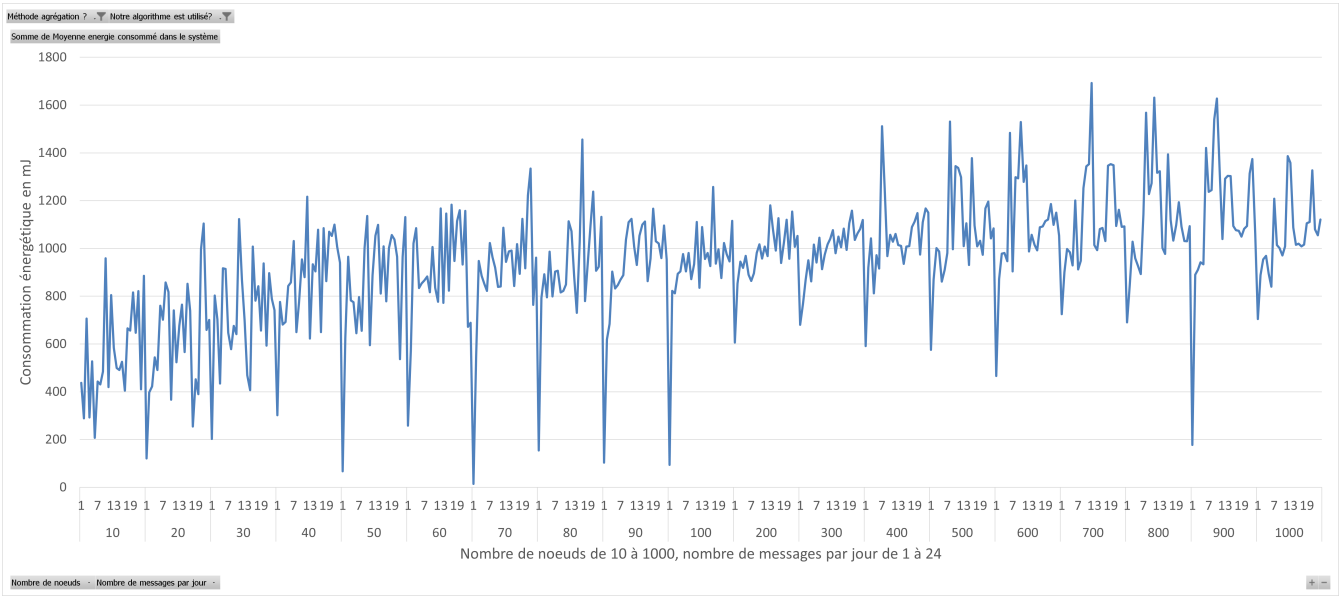


FIGURE 7.5 – Consommation énergétique de notre solution avec agrégation des messages

chapitre, ce n'est que la version MESH. La différence entre les deux graphiques est l'utilisation au niveau des nœuds relais : à savoir les IN avec des IN attachés à eux et les relais de l'agrégation des messages. La Figure 7.5 est celle avec l'agrégation des messages, et la Figure 7.6 est celle sans l'utilisation de l'agrégation. Les trois prochains graphiques sont des comparaisons de la

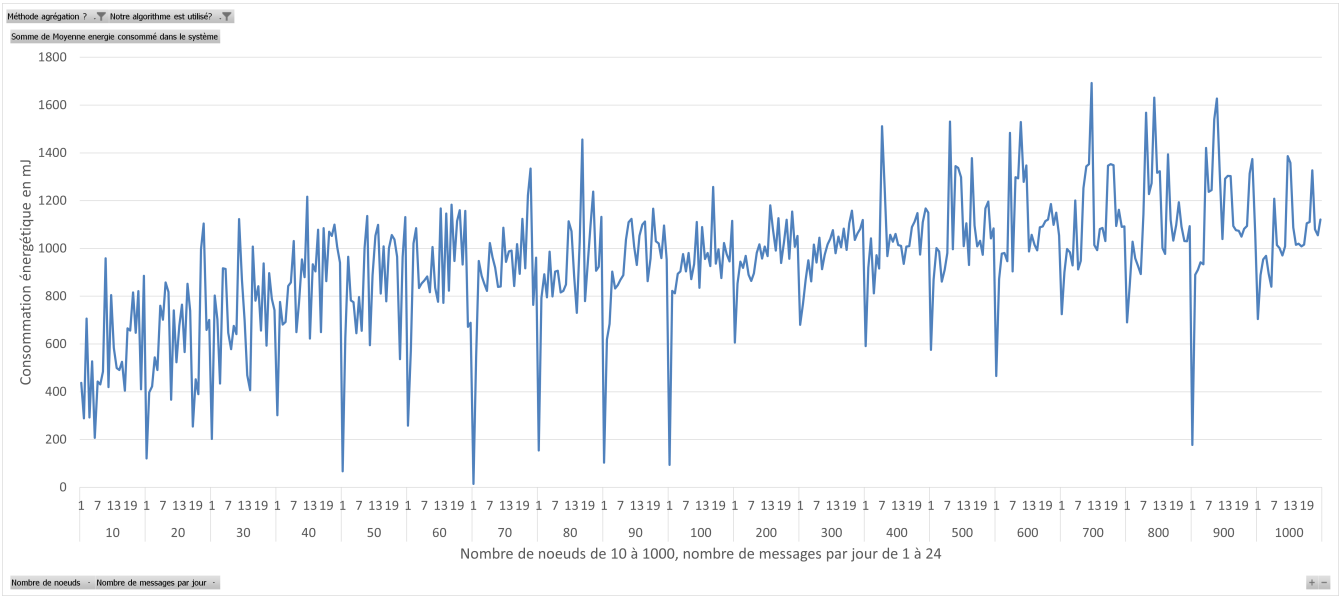


FIGURE 7.6 – Consommation énergétique de notre solution sans agrégation des messages

solution dynamique et présentée au chapitre précédent avec celle de ce chapitre au niveau de la consommation énergétique. La Figure 7.7 met en avant, avec la courbe rouge, la solution maillée, qui est jusqu'à deux fois moins consommatrice dans tous les cas que la solution avec

dynamicité représentée en bleu. Dans la Figure 7.8, le jeu de données est celui uniquement issu

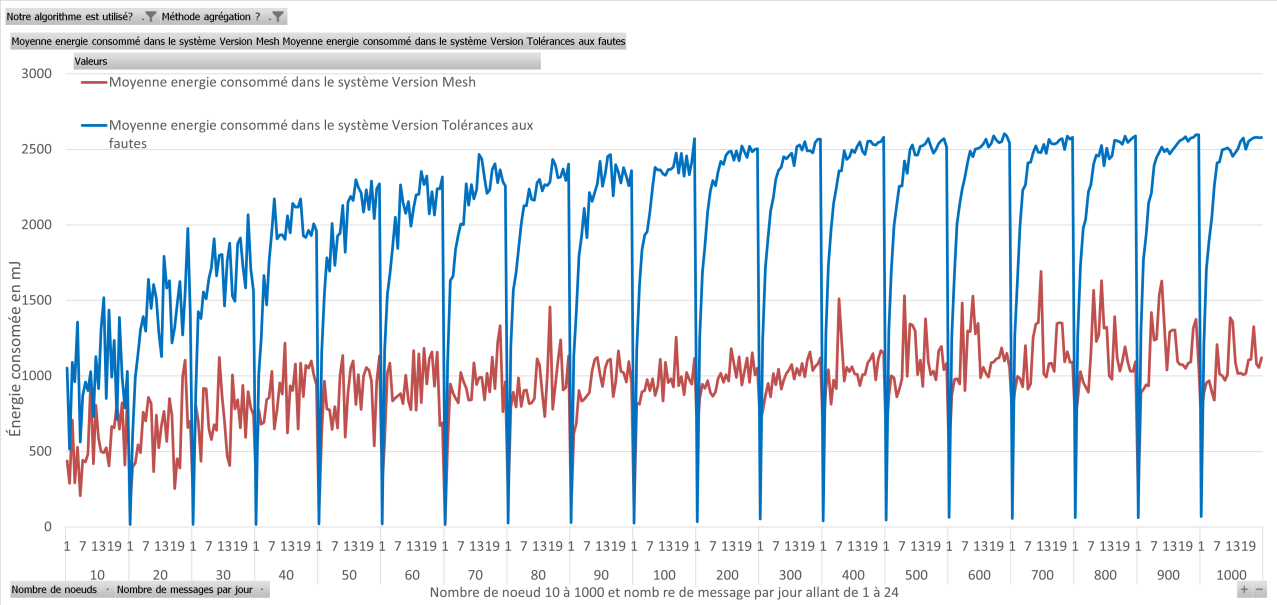


FIGURE 7.7 – Comparaison de la moyenne d'énergie consommé par un noeud dans un système entre les version maillé et tolérantes aux fautes

de la version maillée de notre algorithme. Le nombre de messages par jour est toujours de 1 à 24 et le nombre de nœuds dans le système est de 10 à 1000. Nous observons un résultat beaucoup moins lisse que sur le même graphique avec le jeu de données de la version avec dynamicité qui est représenté dans la Figure 7.9. En effet, les résultats sont des moyennes de 5 simulations avec une seed différente pour un même réseau, l'ordonnancement des événements peut augmenter la consommation de message due à des décalages ou à des apparitions d'erreurs (fautes). Ainsi comme plus de nœuds sont couverts par l'algorithme puisque la solution devient multi-saut, des nœuds auparavant complètement isolés et en dépense d'énergie par manque de synchronisation se retrouvent synchronisés et ne provoquent plus de collisions. C'est une première explication quant à la baisse de consommation. La tendance dans les cas extrêmes montre que certes la consommation est grande mais deux fois moins que dans une solution avec un seul saut.

Pour approfondir la comparaison, revenons avec la Figure 7.10 qui représente de la même manière simultanément les Figures 7.8 et 7.9. Cette fois-ci, le jeu de données représenté n'est celui d'aucune solution, mais du comportement standard d'un appareil. À savoir faire un Join request tant qu'il n'y a pas de Join accept. Là où le maximum de consommation avoisine les 2800 mJ dans cette version, dans la version dynamique on atteint les 2500 et dans la version maillée les 1600 mJ. Le gain est donc perceptible malgré l'augmentation de la taille du réseau et la synchronisation. Si nous n'avions pris dans la solution dynamique que les données issues des nœuds qui avaient un appairage, la consommation relevée serait beaucoup plus faible que celle relevée en moyenne dans le réseau. Après avoir comparé les résultats sur la consommation énergétique, nous passons aux résultats sur les messages.

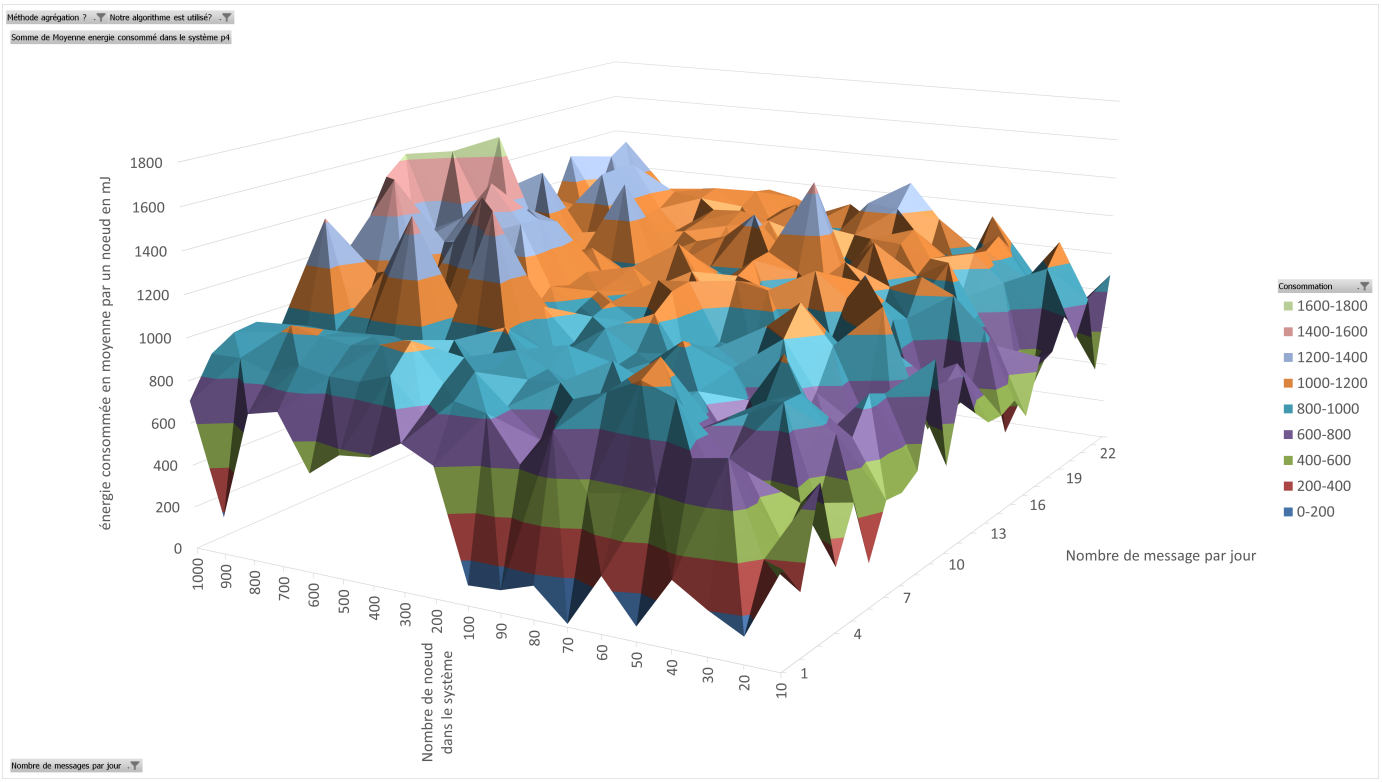


FIGURE 7.8 – Consommation journalière dans le système d'un nœud en moyenne dans un réseaux de 10 à 1000 nœuds en fonction du nombre de message 1 à 24 messages par jour avec notre solution maillée

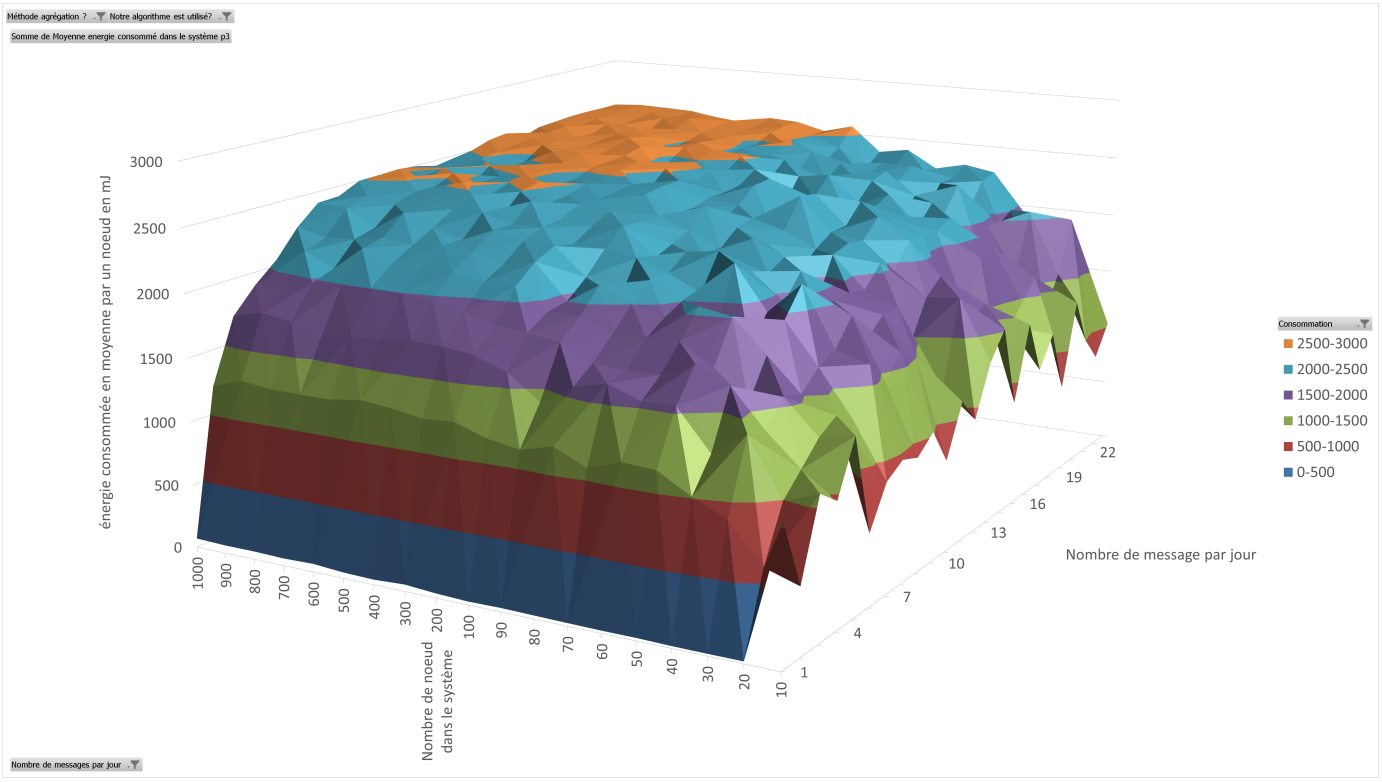


FIGURE 7.9 – Consommation journalière dans le système d'un nœud en moyenne dans un réseau de 10 à 1000 nœuds en fonction du nombre de message 1 à 24 messages par jour avec notre solution tolérante aux fautes

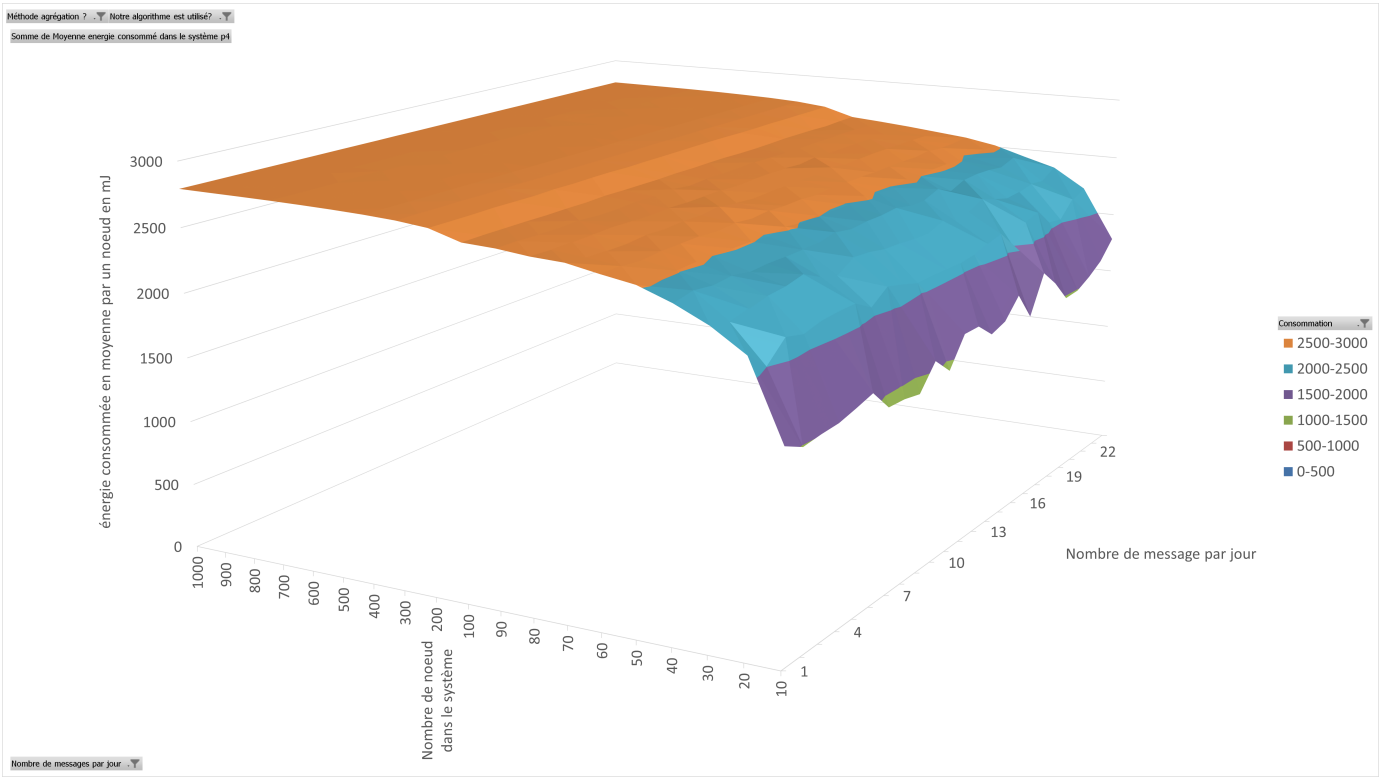


FIGURE 7.10 – Consommation journalière dans le système d'un nœud en moyenne dans un réseau de 10 à 1000 nœuds en fonction du nombre de message 1 à 24 messages par jour sans notre solution

7.5.2 Résultats sur le nombre de messages

Cette seconde partie est structurée autour du nombre de messages. Ainsi la première Figure 7.11 expose le nombre de messages de la solution présentée dans ce chapitre et son taux de réception. Nous allons uniquement nous concentrer sur la partie nombre de messages.

Afin de comparer par rapport à la version avec dynamicité, nous avons créé un deuxième graphique qui compare la version maillée et la version précédente dans la Figure 7.12. Ce qui en est dégagé est que le nombre de messages dans le système est deux fois moins grand que la version avec dynamicité. Ceci s'explique de la même manière que la consommation moyenne dans le réseau. Le nombre de messages dans le réseau est moins grand car le nombre de nœuds sans appairage diminue et donc il y a moins de Join request, moins de Discover, Register infructueux.

Pour une meilleure visibilité, nous avons mis en place dans la Figure 7.13 un graphique en 3D pour mieux observer la présence de messages dans le système sur certain cas. Ce que l'on observe c'est que le nombre de messages est le plus grand lorsque le nombre de récoltes par jour et le nombre de nœuds sont plus élevés. Ce qui est plus notable c'est que le nombre de nœuds influe plus rapidement sur le nombre de messages dans le système. Lorsque le nombre de messages requis dans le système est entre un et deux par jours, le nombre de messages est situé dans le même ordre de grandeur à partir de 300 nœuds. Ce qui en ressort donc c'est que le nombre de nœuds influence plus rapidement que le nombre de messages à récolter par jour.

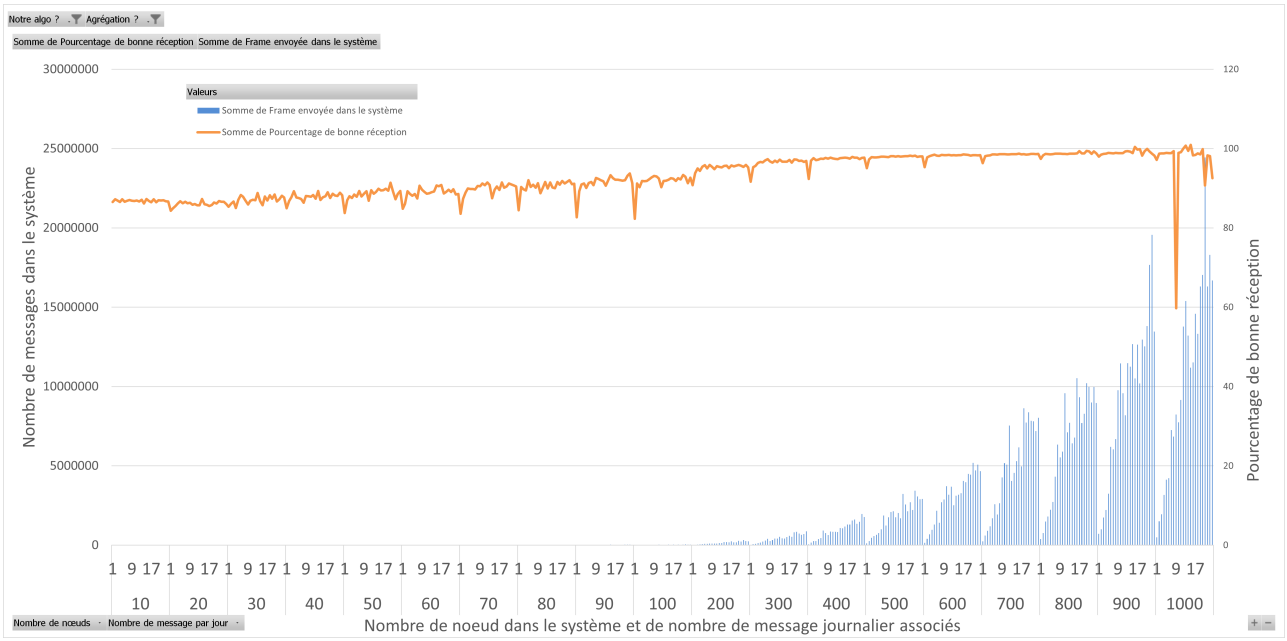


FIGURE 7.11 – Nombre de messages de la solution maillée émis dans le système dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24 avec le taux de bonne réception

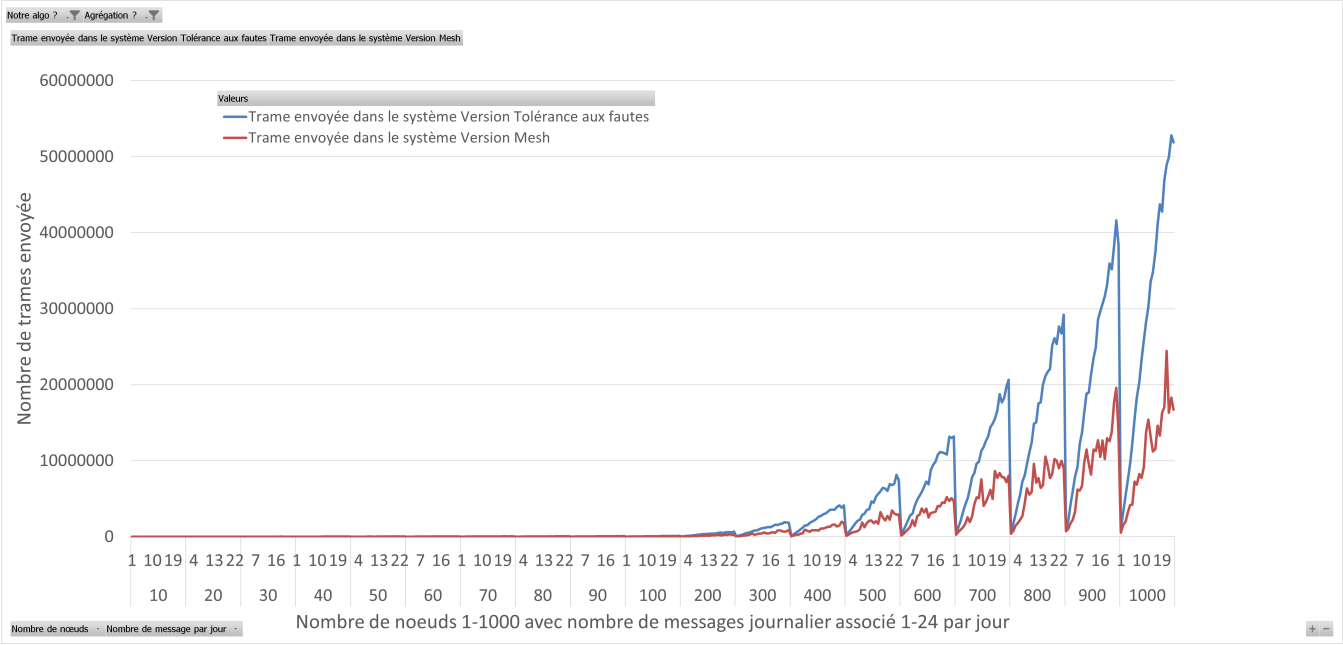


FIGURE 7.12 – Comparaison entre la solution maillée et la solution tolérante aux fautes en matière de nombre de messages émis dans le système dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24

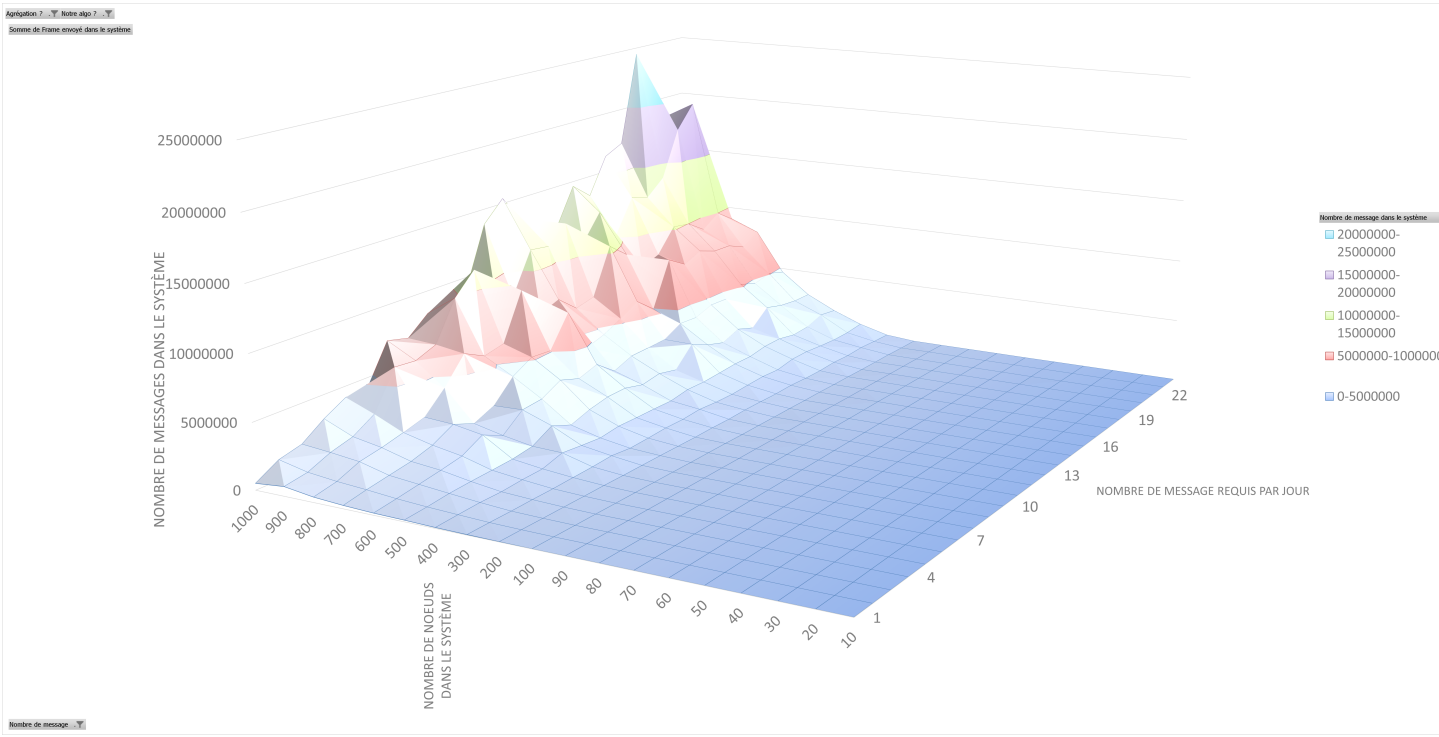


FIGURE 7.13 – Nombre de message de la solution maillée émis dans le système dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24

7.5.3 Résultats sur le pourcentage de bonne réception

Nous avons en premier lieu, le taux de réception lors d'un comportement basique sans solution particulière utilisée, représentée dans la Figure 7.14.

Le taux, dans ce cas, oscille entre 76% et 86%. Pour comparer nous avons le deuxième graphique 7.15, cette fois-ci nous avons la solution maillée utilisée où le taux de bonnes réceptions est entre 87% à 99% dans la plupart des cas. Ce qui s'explique sur le bien fait de la mise en place de la synchronisation multi-saut qui est répartie à travers tout le réseau. Ce qui explique l'absence de collisions malgré le nombre de messages assez élevé présents dans le système.

Pour comparer la solution précédente et l'actuelle, nous avons la dernière Figure 7.16 qui représente le taux de réception en rouge de la version maillée et en bleue la version avec dynamicité. Le gain est sensible mais se ressert assez vite avec l'augmentation du nombre de nœuds et du nombre de messages.

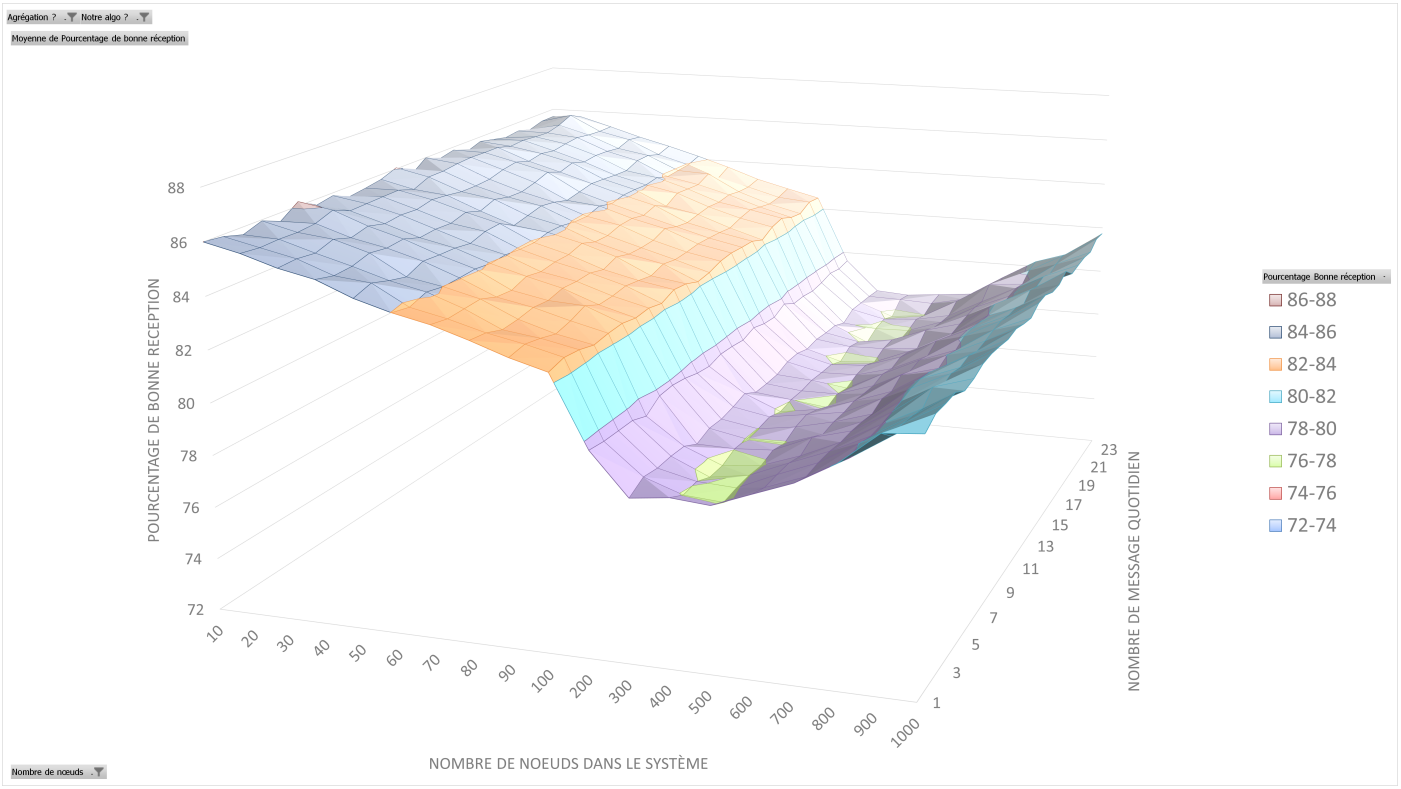


FIGURE 7.14 – Taux de bonnes réceptions de la version sans notre algorithme dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24

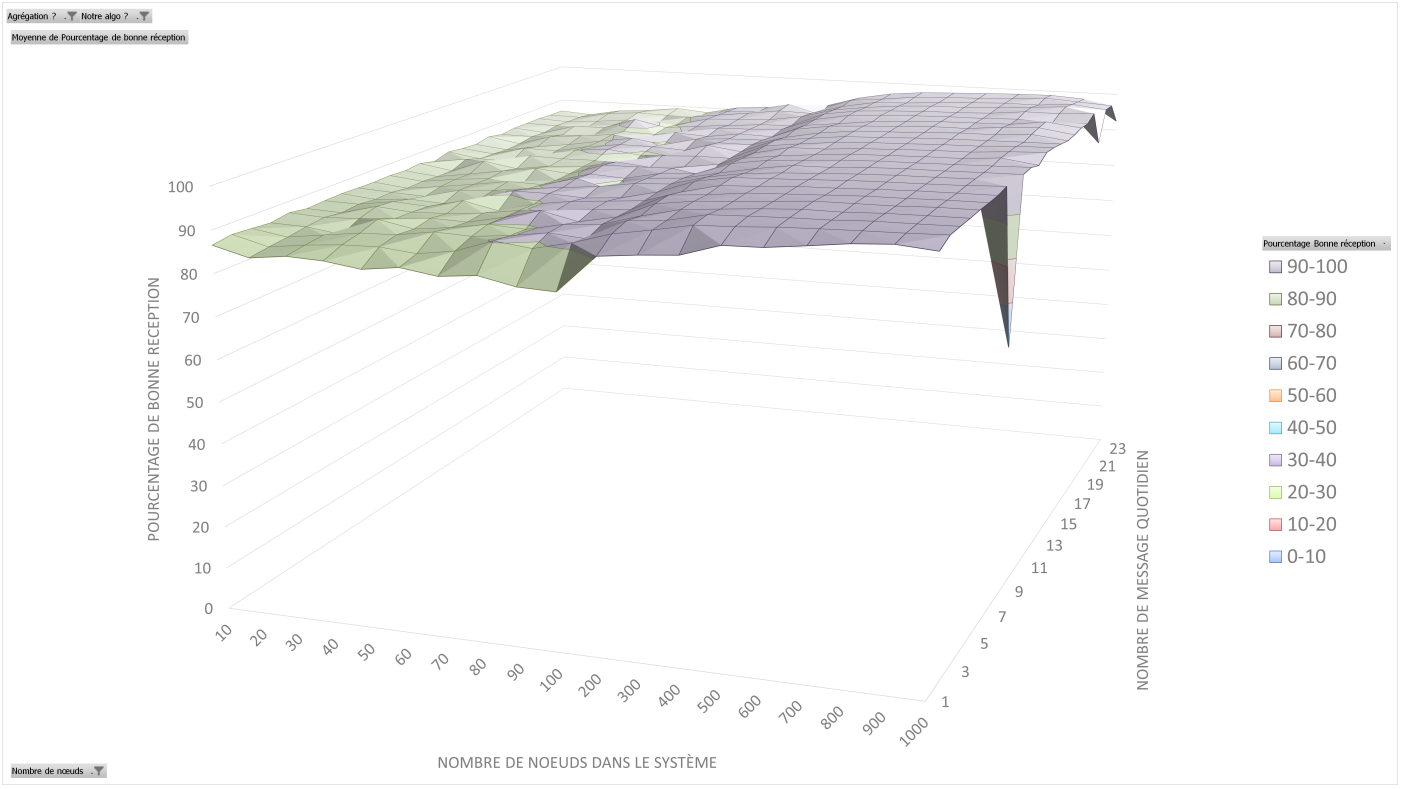


FIGURE 7.15 – Taux de bonnes réceptions de la version maillée dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24

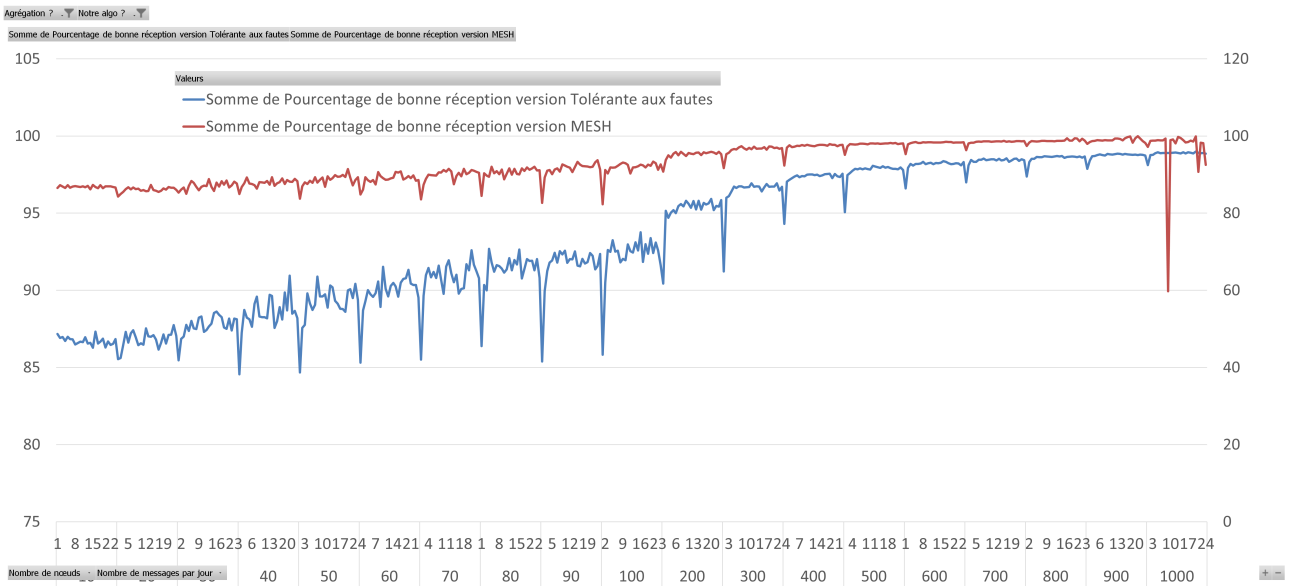


FIGURE 7.16 – Taux de bonnes réceptions comparé entre la version maillée et tolérante aux fautes dans un réseau de 10 à 1000 nœuds en fonction du nombre de messages allant de 1 à 24

7.5.4 Conclusion sur les simulations

La première conclusion consiste à dire que la version maillée est plus efficace en terme de consommation énergétique, en terme de nombre de messages et de taux de réception par rapport à un réseau sans coordination. C'est ce que montrent les simulations. Il n'est pas possible de conclure que c'est le cas entre la version avec dynamicité et sûreté et la version maillée. En effet, les résultats de la version avec dynamicité sont la moyenne à travers le réseau. Dans le réseau nous n'avons pas relevé le nombre de nœuds qui n'étaient pas synchronisés et pourtant ils sont dans la moyenne. Donc, plus il y a de synchronisation entre les nœuds, plus il y a de récolte et moins il y a de collisions. Mais, pour autant, plus le réseau est petit, moins il va consommer pour récolter tous les nœuds.
On peut donc dégager un critère à prendre principalement en compte dans le choix de la solution : la taille du réseau. En effet, si celle-ci est faible, alors une version simple saut est à privilégier. Et si elle est importante, alors il est préférable de choisir la solution multi-saut au détriment de la consommation énergétique.

7.6 Conclusion

Nous avons présenté dans ce chapitre notre solution algorithmique de maillage au sein d'un réseau ainsi que son adaptation à travers notre solution adaptée au LoRa. Nous avons effectué une campagne de simulations dans le même cadre que la précédente pour les comparer. Nous avons également réalisé une campagne algorithmique pour montrer le temps de convergence avant de couvrir le réseau. Nous avons montré dans ce chapitre que la solution maillée se trouve grandement efficace dans un réseau avec beaucoup de nœuds. Cependant, il resterait à comparer la couverture effective des deux solutions et mesurer l'efficacité de chacune pour en tirer une meilleure comparaison.

Conclusion

Dans cette thèse, nous nous sommes intéressés au développement d'une solution de relais de données. Cette solution a pour but d'atténuer les problèmes d'un déploiement massif d'objets connectés au sein d'une même zone : mauvaise connectivité, perte de donnée.

Nous avons proposé l'étude d'une solution incrémentale en fonction des différents besoins :

- Relais de données d'un nœud isolé en un seul saut
- Échange sécurisé des données entre un relais et un isolé
- Gestion de la dynamicité et des différentes fautes dans le système
- Gestion multi-saut des nœuds isolés

Dans le contexte de l'Internet des Objets, des réseaux ad hoc, nous avons évalué nos solutions à l'aide de l'outil de simulation OMNeT++. Pour chaque proposition avancée durant notre étude, nous avons évalué son impact par rapport aux autres propositions et à l'existant. Les résultats ont montré un meilleur taux de réception et une consommation réduite. Nous avons constaté qu'en fonction de la taille du réseau une solution multi-saut est plus efficace sur certaines topologies.

Les travaux de recherches que nous avons menés ont abouti à plusieurs résultats validés au sein de la communauté scientifique via des publications en conférence avec actes et comité de lecture. À savoir six conférences à audience internationale [Fla+20b] [Fla+20d] [FHN20] [Fla+20e] [Fla+20a] [Fla+20c].

7.7 Perspectives

Nous avons à travers cette thèse développé plusieurs axes qui peuvent être encore approfondis. Par exemple, pousser la comparaison entre la solution qui assure la gestion de la dynamicité et la version maillée afin de voir l'efficacité d'une solution par rapport à une autre en fonction du réseau. Une autre piste serait de connaitre un jeu de données cible pour le cas d'étude : la taille des données n'étant pas connue, l'agrégation est peut être un procédé impossible à réaliser. La taille d'un payload LoRaWAN et d'une trame LoRa étant limitées (en fonction du datarate), celle-ci peut être insuffisante. Par ailleurs, nous nous sommes basé sur des consommations aux valeurs annoncées dans nos modèles de simulation , il aurait été plus approprié d'avoir une maquette et d'en tirer les valeurs de consommation énergétique. Avec ces valeurs, nous aurions pu estimer au mieux la durée de vie, l'impact de notre solution sur cet appareil, et donc sur le cas d'étude. Une avant-dernière perspective serait de faire une expérimentation à plus grande échelle et de plus grande durée que réalisée jusqu'alors, nous n'avons réalisé nos tests qu'avec huit appareils STM32 Discovery kit. Il aurait été intéressant d'en déployer un nombre plus

important, comme par exemple cinquante objets. Nous aurions pu ainsi avoir des comportements plus réalistes quant aux problématiques relevées à travers cette thèse : collisions, environnement. Cependant, un facteur non négligeable et qui n'a pu être reproduit durant nos expérimentations est celui de générateur de bruit radio afin de simuler un environnement urbain. Une dernière piste de perspectives serait d'élargir encore plus l'adaptation à différents protocoles radio.

Annexe

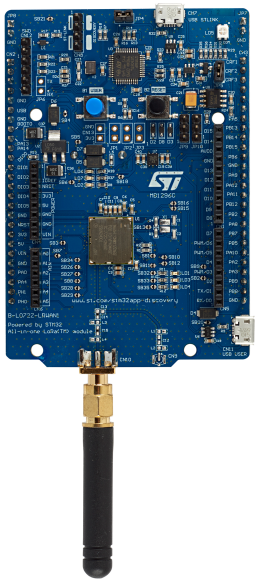
7.8 Datasheet tb-l072z-lrwan1

life.augmented

B-L072Z-LRWAN1

Data brief

Discovery kit for LoRaWAN™, Sigfox™, and LPWAN protocols with STM32L0



Picture is not contractual.

Product status link
B-L072Z-LRWAN1

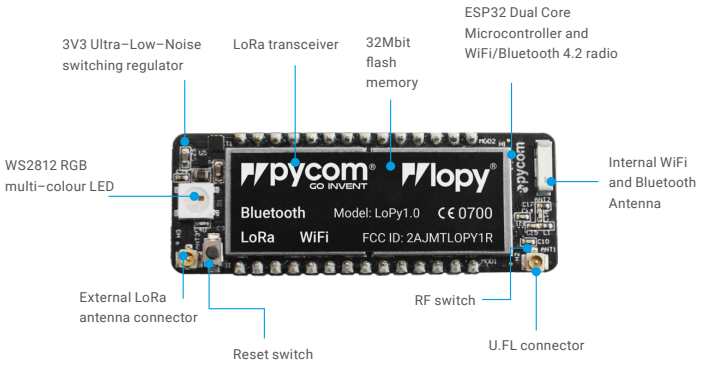
arm MBED Enabled

Features

- CMWX1ZZABZ-091 LoRa®/Sigfox™ module (Murata)
  - Embedded ultra-low-power STM32L072CZ MCU, based on Arm® Cortex®-M0+ core, with 192 Kbytes of Flash memory, 20 Kbytes of RAM, 20 Kbytes of EEPROM
  - Frequency range: 860 MHz - 930 MHz
  - USB 2.0 FS
  - 4-channel,12-bit ADC, 2 × DAC
  - 6-bit timers, LP-UART, I^2C and SPI
  - Embedded SX1276 transceiver
  - LoRa®, FSK, GFSK, MSK, GMSK, and OOK modulations (+ Sigfox™ compatibility)
  - +14 dBm or +20 dBm selectable output power
  - 157 dB maximum link budget
  - Programmable bit rate up to 300 kbit/s
  - High sensitivity: down to -137 dBm
  - Bullet-proof front end: IIP3 = -12.5 dBm
  - 89 dB blocking immunity
  - Low Rx current of 10 mA, 200 nA register retention
  - Fully integrated synthesizer with a resolution of 61 Hz
  - Built-in bit synchronizer for clock recovery
  - Sync word recognition
  - Preamble detection
  - 127 dB+ dynamic range RSSI
  - LoRaWAN™ Class A certified
- SMA and U.FL RF interface connectors
- Including 50-ohm SMA RF antenna
- 7 LEDs:
  - 4 general-purpose LEDs
  - 5 V power LED
  - ST-LINK-communication LED
  - Fault-power LED
- 1 user and 1 reset push-buttons
- On-board ST-LINK/V2-1 debugger/programmer with USB re-enumeration capability: mass storage, Virtual COM port, and debug port
- Arduino™ Uno V3 connectors
- Board power supply through the USB bus or external VIN/3.3 V supply voltage or batteries
- 3 × AAA-type battery holder for standalone operation
- Support of a wide choice of Integrated Development Environments (IDEs) including IAR™, Keil®, GCC-based IDEs, Arm® Mbed™

DB3090 - Rev 4 - June 2019
For further information contact your local STMicroelectronics sales office.
www.st.com

7.9 Datasheet LoPy



Size
55mm x 20mm x 3.5mm
(excluding headers)

Operating temperature:
–40 to 85 degrees celsius

1.0 Overview

With LoRa, Wifi and BLE, the LoPy is the only triple bearer MicroPython enabled micro controller on the market today – the perfect enterprise grade IoT platform for your connected Things. With the latest Espressif chipset the LoPy offers a perfect combination of power, friendliness and flexibility. Create and connect your things everywhere. Fast.

2.0 Features

- Powerful CPU, BLE and state of the art WiFi radio. 1KM Wifi Range
- Can also double up as a Nano LoRa gateway
- MicroPython enabled
- Fits in a standard breadboard (with headers)
- Ultra–low power usage: a fraction compared to other connected micro controllers
- Available with or without pin headers soldered on

3.0 Specifications

3.1 CPU
- Xtensa® dual–core 32–bit LX6 microprocessor(s), up to 600 DMIPS
- Hardware floating point acceleration
- Python multi–threading
- An extra ULP–coprocessor that can monitor GPIOs, the ADC channels and control most of the internal peripherals during deep–sleep mode while only consuming 25uA.

3.2 Memory
- RAM: 512KB
- External flash: 4MB

3.3 WiFi
- 802.11b/g/n 16mbps

3.4 Bluetooth
- Low energy and classic

3.5 LoRa
- LoRaWAN 1.0.2 stack - Class A and C devices
- Node range: Up to 40km
- Nano-gateway: Up to 22km (Capacity up to 100 nodes)

3.6 RTC
- Running at 150kHz

3.7 Security
- SSL/TLS support
- WPA Enterprise security

3.8 Hash / encryption
- SHA
- MD5
- DES
- AES

4.0 Block Diagram

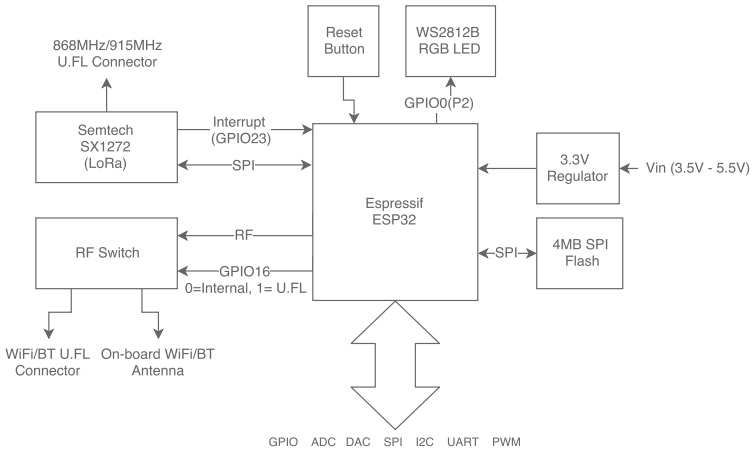


Figure 1 – System block diagram

7.10 Datasheet nRF9160

NORDIC SEMICONDUCTOR

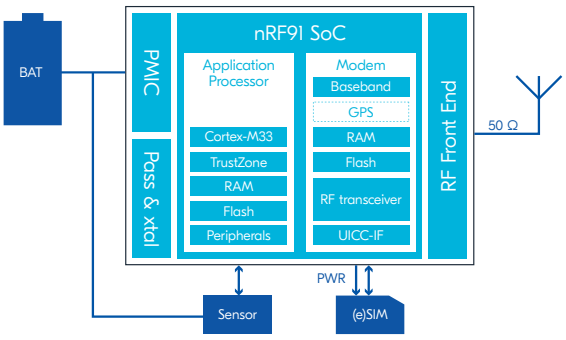
nRF9160 cellular IoT System-in-Package

Low power SiP with integrated LTE-M and NB-IoT wireless modem

Overview
The nRF9160 SiP is making the latest LTE technology accessible for a wide range of applications and developers. Through the high integration and pre-certification for global operation, it solves the complex wireless design challenges as well as the comprehensive set of qualifications needed to utilize cellular technology.

By integrating an application processor, multimode LTE-M/NB-IoT modem, RF front end (RFFE), GPS and power management in a 10×16×1 mm package, it offers the most compact solution for cellular IoT (cIoT) on the market.

Targeting asset tracking applications, the nRF9160 SiP has built-in assisted GPS. It combines location data from the Cloud with GPS satellite trilateration to allow remote monitoring of the device position.



Application Circuit

Application processor
The nRF9160 SiP offers a modern and powerful 64 MHz Arm Cortex-M33 CPU with on-chip flash and RAM exclusively for application use.

A range of analog and digital peripherals supports the powerful application processor and enables advanced single chip cellular IoT products.

The integrated cryptographic and security features enables the nRF9160 to meet the latest requirements on internet security and authentication. By including trusted execution capability on the application processor, it takes security a step further by securing the most critical processes and peripherals in the application. The on-chip modem is its own security island.

KEY FEATURES

- Fully integrated SiP for cellular IoT
- Dedicated application processor and memory
- Multimode LTE-M/NB-IoT modem with integrated RFFE
- Assisted GPS
- Single variant certified for global operation
- Certified LTE bands: B1, B2, B3, B4, B5, B8, B12, B13, B14, B17, B18, B19, B20, B25, B26, B28 and B66
- Mobile network operator certifications: AIS, AT&T, Bell, China Telecom, Deutsche Telekom, Telstra, Verizon and Vodafone
- Global and regulatory certifiations: GCF, PTCRB, FCC, CE, ISED, ACMA RCM, NCC, IMDA, MIC, MSIP, IFT, ICASA, NBTC
- Overview of all certifications: ***nordicsemi.com/9160cert***
- 10×16×1 mm LGA package

LTE-M/NB-IoT modem
- Integrated RFFE
- 700-2200 MHz LTE band support
- 23 dBm output power
- Assisted GPS
- eDRX and PSM power saving modes
- Coverage enhancement modes
- SMS, IPv4/IPv6
- TCP/UDP, TLS/DTLS
- Single pin 50 Ω antenna interface
- UICC interface

Application processor
- 64 MHz Arm® Cortex®-M33 CPU
- Arm TrustZone® for trusted execution
- Arm CryptoCell 310 for accelerated cryptography
- 1 MB Flash & 256 KB RAM
- 4 x SPI/UART/TWI
- PDM, I2S, PWM, ADC
- Automated power and clock management
- 32 GPIOs

APPLICATIONS

- Logistics and asset tracking
- Smart city
- Smart agriculture
- Predictive maintenance & industrial
- Wearables & medical

7.11 Datasheet nRF9160

LTE-M/NB-IoT modem
The nRF9160 LTE modem integrates RFFE, radio and baseband. It supports operation worldwide, enabling cIoT products without regional specific variants.

The LTE modem supports half-duplex FDD operation and all power saving and coverage enhancement modes. A single pin antenna interface is available.

The LTE stack layers L1-L3, IPv4/IPv6, TCP/UDP, TLS/DTLS are all part of the modem firmware. The application processor communicates with the LTE modem through a BSD secure sockets API and contains the application layer protocols such as HTTP, CoAP, MQTT or LWM2M, and the application itself.

The nRF9160 LTE modem supports both SIM and eSIM, plug-in or soldered. It provides power to the SIM and handles all communication automatically.

Designed for true low power cIoT
The nRF9160 SiP is specifically designed to take full advantage of the energy efficiency possibilities associated with the LTE-M and NB-IoT standards. Nordic designs all hardware and software, and as such offers an unparalleled, high efficient and optimized low power cIoT solution.

It supports both the PSM and eDRX power saving modes, enabling the nRF9160 to sleep for longer periods of time. For both LTE-M and NB-IoT the PSM floor current is as low as 2.7 uA, and with an eDRX interval of 655 seconds the average current is 6 uA for LTE-M and 9 uA for NB-IoT.

Get started today
The nRF Connect SDK is the software development kit for the nRF9160 SiP, including everything needed to get started, and much more. It integrates the Zephyr RTOS, application layer protocols such as HTTP, CoAP, MQTT and LWM2M, and application examples covering a wide range of use cases. It also includes software for secure boot, and secure firmware over-the-air (FOTA) for both application and modem firmware. The necessary firmware for the LTE modem is offered as pre-certified and precompiled downloads.

The nRF9160 DK is an affordable, pre-certified single board development kit for the nRF9160 SiP, facilitating development with LTE-M, NB-IoT and GPS.

RELATED PRODUCTS

nRF9160 DK	Development kit for the nRF9160 SiP
Nordic Thingy:91	Cellular IoT protoyping platform
nRF Connect SDK	Cellular IoT software development kit

KEY DATA

LTE-M/NB-IoT modem	
Frequency range	700-2200 MHz
Throughput (DL/UL)	LTE-M: 300/375 kbps NB-IoT: 30/60 kbps
Output power	Up to 23 dBm
RX sensitivity	LTE-M: -108 dBm NB-IoT: -114 dBm GPS: -155 dBm
Mode	HD-FDD

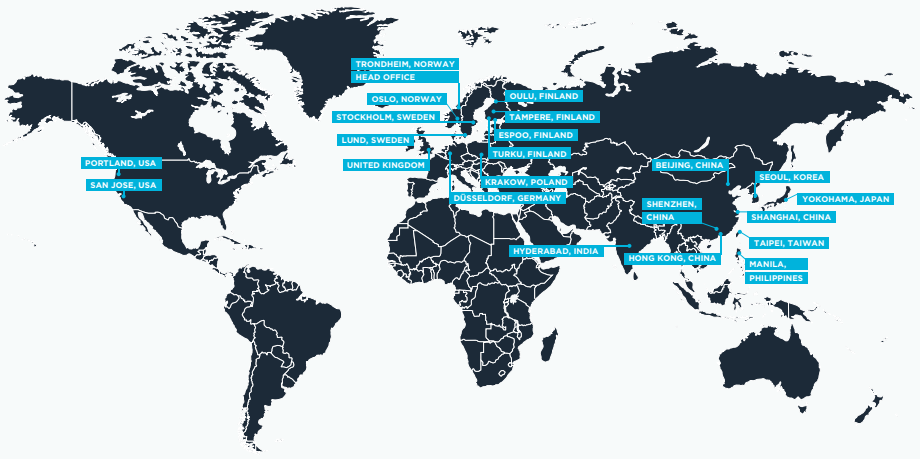
Application processor	
CPU	64 MHz Arm Cortex-M33 Arm TrustZone
Flash	1 MB
RAM	256 KB
Peripherals	Arm Cryptocell 310 3 × TIMER, 2 × RTC WDT
Interfaces	4 × SPI (M/S) / UARTE / TWI (M/S) 4 × PWM, PDM, I2S 12 bit/200 ksps ADC

Current consumption (23 dBm TX power, 3.7 V supply)	
PSM floor current	LTE-M: 2.7 uA NB-IoT: 2.7 uA
eDRX, 655 seconds	LTE-M: 6 uA NB-IoT: 9 uA

Operating conditions and package	
Supply voltage	3.0-5.5 V
Temperature	-40-85 °C
Package	10×16×1 mm LGA

WORLD WIDE OFFICE LOCATIONS

Headquarters:
Trondheim, Norway
Tel: +47 72 89 89 00



For more information
Visit **nordicsemi.com** for the complete product specification about this and any other wireless ULP products.

About Nordic Semiconductor
Nordic Semiconductor is a fabless semiconductor company specializing in ULP short-range wireless communication. Nordic is a public company listed on the Norwegian stock exchange.

LTE-M/NB-IoT modem
The nRF9160 LTE modem integrates RFFE, radio and baseband. It supports operation worldwide, enabling cIoT products without regional specific variants.

The LTE modem supports half-duplex FDD operation and all power saving and coverage enhancement modes. A single pin antenna interface is available.

The LTE stack layers L1-L3, IPv4/IPv6, TCP/UDP, TLS/DTLS are all part of the modem firmware. The application processor communicates with the LTE modem through a BSD secure sockets API and contains the application layer protocols such as HTTP, CoAP, MQTT or LWM2M, and the application itself.

The nRF9160 LTE modem supports both SIM and eSIM, plug-in or soldered. It provides power to the SIM and handles all communication automatically.

Designed for true low power cIoT
The nRF9160 SiP is specifically designed to take full advantage of the energy efficiency possibilities associated with the LTE-M and NB-IoT standards. Nordic designs all hardware and software, and as such offers an unparalleled, high efficient and optimized low power cIoT solution.

It supports both the PSM and eDRX power saving modes, enabling the nRF9160 to sleep for longer periods of time. For both LTE-M and NB-IoT the PSM floor current is as low as 2.7 uA, and with an eDRX interval of 655 seconds the average current is 6 uA for LTE-M and 9 uA for NB-IoT.

Get started today
The nRF Connect SDK is the software development kit for the nRF9160 SiP, including everything needed to get started, and much more. It integrates the Zephyr RTOS, application layer protocols such as HTTP, CoAP, MQTT and LWM2M, and application examples covering a wide range of use cases. It also includes software for secure boot, and secure firmware over-the-air (FOTA) for both application and modem firmware. The necessary firmware for the LTE modem is offered as pre-certified and precompiled downloads.

The nRF9160 DK is an affordable, pre-certified single board development kit for the nRF9160 SiP, facilitating development with LTE-M, NB-IoT and GPS.

RELATED PRODUCTS

nRF9160 DK	Development kit for the nRF9160 SiP
Nordic Thingy:91	Cellular IoT protoyping platform
nRF Connect SDK	Cellular IoT software development kit

KEY DATA

LTE-M/NB-IoT modem	
Frequency range	700-2200 MHz
Throughput (DL/UL)	LTE-M: 300/375 kbps NB-IoT: 30/60 kbps
Output power	Up to 23 dBm
RX sensitivity	LTE-M: -108 dBm NB-IoT: -114 dBm GPS: -155 dBm
Mode	HD-FDD

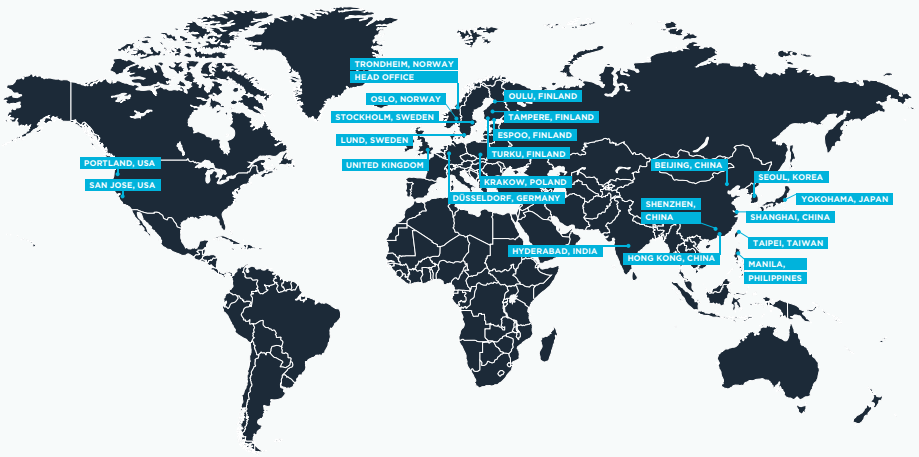
Application processor	
CPU	64 MHz Arm Cortex-M33 Arm TrustZone
Flash	1 MB
RAM	256 KB
Peripherals	Arm Cryptocell 310 3 × TIMER, 2 × RTC WDT
Interfaces	4 × SPI (M/S) / UARTE / TWI (M/S) 4 × PWM, PDM, I2S 12 bit/200 ksps ADC

Current consumption (23 dBm TX power, 3.7 V supply)	
PSM floor current	LTE-M: 2.7 uA NB-IoT: 2.7 uA
eDRX, 655 seconds	LTE-M: 6 uA NB-IoT: 9 uA

Operating conditions and package	
Supply voltage	3.0-5.5 V
Temperature	-40-85 °C
Package	10×16×1 mm LGA

WORLD WIDE OFFICE LOCATIONS

Headquarters:
Trondheim, Norway
Tel: +47 72 89 89 00



For more information
Visit **nordicsemi.com** for the complete product specification about this and any other wireless ULP products.

About Nordic Semiconductor
Nordic Semiconductor is a fabless semiconductor company specializing in ULP short-range wireless communication. Nordic is a public company listed on the Norwegian stock exchange.

7.12 Datasheet Feather

HOPERF ELECTRONIC

RFM95/96/97/98（W）

RFM95/96/97/98(W) - Low Power Long Range Transceiver Module V1.0

GENERAL DESCRIPTION

The RFM95/96/97/98(W) transceivers feature the LoRa™ long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using Hope RF's patented LoRa™ modulation technique RFM95/96/97/98(W) can achieve a sensitivity of over -148dBm using a low cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRa™ also provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

These devices also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4g. The RFM95/96/97/98(W) deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.

RFM95/96/97/98(W)

KEY PRODUCT FEATURES

- LoRa™ Modem.
- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs. V supply.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -148 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention.
- Fully integrated synthesizer with a resolution of 61 Hz.
- FSK, GFSK, MSK, GMSK, LoRa™ and OOK modulation.
- Built-in bit synchronizer for clock recovery.
- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC.
- Packet engine up to 256 bytes with CRC.
- Built-in temperature sensor and low battery indicator.
- Modue Size：16*16mm

APPLICATIONS

- Automated Meter Reading.
- Home and Building Automation.
- Wireless Alarm and Security Systems.
- Industrial Monitoring and Control
- Long range Irrigation Systems

Tel: +86-755-82973805 Fax: +86-755-82973550 E-mail: sales@hoperf.com http://www.hoperf.com

7.13 Datasheet SAMR34

MICROCHIP

SAM R34/R35

SAM R34/R35 Low Power LoRa® Sub-GHz SiP Datasheet

Introduction

The SAM R34/R35 is a family of ultra-low power microcontrollers combined with a UHF transceiver communication interface. It uses a 32-bit ARM® Cortex® -M0+ processor and offers up to 256 KB of Flash and 40 KB of SRAM, including an area of battery backed-up SRAM. The UHF transceiver supports LoRa® and FSK modulation. LoRa technology is a spread spectrum protocol optimized for low data-rate, ultra-long range signaling. It is ideal for battery-powered remote sensors and controls.

The SAM R34 includes an integrated microcontroller with USB and the UHF transceiver, making it suitable for USB dongle applications or for software updates via USB. The SAM R35 offers the same microcontroller functions along with the UHF transceiver without the USB interface.

Features

Operational Features

- Processor:
  - ARM Cortex -M0+ CPU running at up to 48 MHz (2.46CoreMark®/MHz)
  - Single-Cycle Hardware Multiplier
  - Micro Trace Buffer (MTB)
- Memory:
  - In-System Self-Programmable Flash Memory, with options for sizes - 256 KB, 128 KB or 64 KB
  - Static Random Access Memory (SRAM) with options for sizes - 32 KB, 16 KB or 8 KB
  - Low power SRAM Memory with option for sizes - 4 KB or 8 KB
- System:
  - Power-on Reset (POR) and Brown-out Reset
  - Internal and External Clock Options with 48 MHz Digital Frequency Locked Loop (DFLL48M) and 48 MHz to 96 MHz Fractional Digital Phase Locked Loop (FDPLL96M)
  - External Interrupt Controller (EIC)
  - Up to 16 External Interrupts
  - One Non-Maskable Interrupt
  - Two Pin Serial Wire Debug (SWD) Programming, Test and Debugging Interfaces
- Operating Voltage: 1.8V- 3.6V
- Low Power Consumption
  - Transceiver:
    - RX = 16 mA (typical)
    - RFO_HF = 33 mA (typical)
    - PA_BOOST = 95 mA (typical)

  - MCU:
    - Idle, Standby, and Backup Sleep Modes
    - SleepWalking peripherals
- Temperature Range: -40°C to +85°C (Industrial)

RF/Analog Features

- Integrated LoRa Technology Transceiver:
  - Tri-band Coverage
    - 137 MHz to 175 MHz
    - 410 MHz to 525 MHz
    - 862 MHz to 1020 MHz
  - +20 dBm (100 mW) Max Power (VDDANA > 2.4 VDC)
  - +17 dBm (50 mW) Max Power (Regulated PA)
  - +13 dBm (20 mW) High-efficency PA
- High Sensitivity:
  - Down to -136 dBm (LoRaWAN™ protocol compliant modes)
  - Down to -148 dBm (proprietary narrowband modes)
- Up to 168 dB Maximum Link Budget
- Robust Front-End: IIp3 = -11 dBm
- Excellent Blocking Immunity
- Low RX Current of 17 mA (typical)
- Fully Integrated Synthesizer with a Resolution of 61 Hz
- LoRa Technology, (G)FSK, (G)MSK and OOK Modulation
- Preamble Detection
- 127 dB Dynamic Range RSSI
- Automatic RF Sense and CAD with Ultra-Fast Automatic Frequency Control (AFC) Packet Engine up to 256 bytes with Cyclic Redundancy Check (CRC)

Peripheral Information

- 16-Channel Direct Memory Access Controller (DMAC)
- 12-Channel Event System
- Three 16-bit Timer/Counters (TC), configurable as either of the following:
  - One 8-bit TC with compare/capture channels
  - One 16-bit TC with compare/capture channels
  - One 32-bit TC with compare/capture channels, by using two TCs
- Two 24-bit and one 16-bit Timer/Counters for Control (TCC), with Extended Functions:
  - Up to four compare channels with optional complementary output
  - Generation of synchronized Pulse Width Modulation (PWM) pattern across port pins
  - Deterministic fault protection, fast decay and configurable dead-time between complementary output
  - Dithering that increases resolution with up to five bit and reduces quantization error
- 32-bit Real Time Counter (RTC) with Clock/Calendar Function
- Watchdog Timer (WDT)

- CRC-32 Generator
- One Full-Speed (12 Mbps) Universal Serial Bus (USB) 2.0 Interface:
  - Embedded host and device function
  - Eight endpoints
- Up to Five Serial Communication Interfaces (SERCOM), each configurable to operate as either of the following:
  - USART with full-duplex and single-wire half-duplex configuration
  - I2C up to 3.4 MHz
  - Serial Peripheral Interface (SPI)
  - Local Interconnect Network (LIN) Slave
- One 12-bit, 1 Msps Analog-to-Digital Converter (ADC) with up to Eight External Channels:
  - Differential and single-ended input
  - Automatic offset and gain error compensation
  - Oversampling and decimation in hardware to support 13-, 14-, 15-, or 16-bit resolution
- Two Analog Comparators (AC) with Window Compare Function
- Peripheral Touch Controller (PTC):
  - 18-channel capacitive touch and proximity sensing

Package Information

- 27 Programmable I/O Pins
- 64 Lead Ball Grid Array (BGA)

7.14 Datasheet STEVAL-STRTK01

life.augmented

STEVAL-STRKT01

Data brief

LoRa® IoT tracker

Product summary	
LoRa IoT tracker	STEVAL-STRKT01
STM32Cube function pack for IoT tracker node with LoRa connectivity, GNSS and sensors	FP-ATR-LORA1
Ultra-low-power ARM Cortex-M0+ MCU	STM32L072CZ
Tiny GNSS module	Teseo-LIV3F
Li-Ion linear battery charger	STBC02
USB Type-C controller	STUSB1600A
2 Mbit serial SPI bus EEPROM	M95M02-DR
400 mA nano-quiescent synchronous step-down converter	ST1PS01

Features

- Optimized IoT tracker solution over LoRaWAN™ network with simultaneous multi-constellation GNSS positioning and geofencing support
- Battery operated solution with smart power management architecture
- First IoT ST reference embedding a USB Type-C connector and a port controller
- Environmental and motion sensors
- Data logging
- STM32Cube function pack (FP-ATR-LORA1)
- High flexibility to cover different application profiles:
  - asset tracker
  - people and animal tracker
  - fleet management
- WEEE and RoHS compliant
- 2006/66/EC Directive compliant
- Contains trasmitter module FCC ID: VPYCMABZ and IC ID: 772C-CMABZ
- CE certified

Description

The STEVAL-STRKT01 LoRa® IoT tracker is designed and optimized to implement the latest technologies in IoT tracker applications such as asset, people and animal tracking as well as fleet management.

The evaluation board simplifies prototyping, evaluation and development of tracker innovative solutions. It comes with comprehensive software, firmware libraries, tools, battery, cables and plastic case.

Thanks to the STM32L072CZ embedded in the CMWX1ZZABZ-091 LoRa® module (by Murata), the STEVAL-STRKT01 allows acquiring position, managing geofence and data logging from Teseo-LIV3F GNSS module and monitoring motion (LIS2DW12) and environmental (HTS221 and LPS22HB) sensors.

The board also transmits and receives data, configurations and events to and from the cloud over a LoRaWAN™ network, or stores data locally in the M95M02-DR EEPROM.

The STEVAL-STRKT01 is a LiPo battery operated solution and implements low power strategies thanks to an enhanced power/battery management design, based on the STBC02 battery charger and the ST1PS01 step-down converter, to ensure long battery autonomy. The STUSB1600A addresses 5 V USB Type-C port management and offers high voltage protection against short circuits.

DB3766 - Rev 1 - November 2018
For further information contact your local STMicroelectronics sales office.
www.st.com

7.15 Datasheet CC1350

TEXAS INSTRUMENTS
CC1350
SWRS183B – JUNE 2016 – REVISED JULY 2018

CC1350 SimpleLink™ Ultra-Low-Power Dual-Band Wireless MCU

1 Device Overview

1.1 Features

- World's First Dual-Band (Sub-1 GHz and 2.4 GHz) Wireless Microcontroller
- Microcontroller
  - Powerful Arm® Cortex®-M3 Processor
  - EEMBC CoreMark® Score: 142
  - EEMBC ULPBench™ Score: 158
  - Clock Speed up to 48-MHz
  - 128KB of In-System Programmable Flash
  - 8KB of SRAM for Cache (or as General-Purpose RAM)
  - 20KB of Ultra-Low-Leakage SRAM
  - 2-Pin cJTAG and JTAG Debugging
  - Supports Over-the-Air (OTA) Update
- Ultra-Low-Power Sensor Controller
  - Can Run Autonomously From the Rest of the System
  - 16-Bit Architecture
  - 2KB of Ultra-Low-Leakage SRAM for Code and Data
- Efficient Code-Size Architecture, Placing Parts of TI-RTOS, Drivers, *Bluetooth*® low energy Controller and Bootloader in ROM
- RoHS-Compliant Package
  - 7-mm × 7-mm RGZ VQFN48 (30 GPIOs)
  - 5-mm × 5-mm RHB VQFN32 (15 GPIOs)
  - 4-mm × 4-mm RSM VQFN32 (10 GPIOs)
- Peripherals
  - All Digital Peripheral Pins Can Be Routed to Any GPIO
  - Four General-Purpose Timer Modules (Eight 16-Bit or Four 32-Bit Timers, PWM Each)
  - 12-Bit ADC, 200 ksamples/s, 8-Channel Analog MUX
  - Continuous Time Comparator
  - Ultra-Low-Power Clocked Comparator
  - Programmable Current Source
  - UART
  - 2× SSI (SPI, MICROWIRE, TI)
  - I^2C, I2S
  - Real-Time Clock (RTC)
  - AES-128 Security Module
  - True Random Number Generator (TRNG)
  - Support for Eight Capacitive Sensing Buttons
  - Integrated Temperature Sensor
- External System
  - On-Chip Internal DC/DC Converter
  - Seamless Integration With the SimpleLink™ CC1190 and CC2592 Range Extender
- Low Power
  - Wide Supply Voltage Range: 1.8 to 3.8 V
  - RX: 5.4 mA (Sub-1 GHz), 6.4 mA (Bluetooth low energy, 2.4 GHz)
  - TX at +10 dBm: 13.4 mA (Sub-1 GHz)
  - TX at +9 dBm: 22.3 mA (Bluetooth low energy, 2.4 GHz)
  - TX at +0 dBm: 10.5 mA (Bluetooth low energy, 2.4 GHz)
  - Active-Mode MCU 48 MHz Running Coremark: 2.5 mA (51 µA/MHz)
  - Active-Mode MCU: 48.5 CoreMark/mA
  - Active-Mode Sensor Controller at 24 MHz: 0.4 mA + 8.2 µA/MHz
  - Sensor Controller, One Wakeup Every Second Performing One 12-Bit ADC Sampling: 0.95 µA
  - Standby: 0.7 µA (RTC Running and RAM and CPU Retention)
  - Shutdown: 185 nA (Wakeup on External Events)
- RF Section
  - 2.4-GHz RF Transceiver Compatible With Bluetooth low energy 4.2 Specification
  - Excellent Receiver Sensitivity –124 dBm Using Long-Range Mode, –110 dBm at 50 kbps (Sub-1 GHz), –87 dBm at Bluetooth low energy
  - Excellent Selectivity (±100 kHz): 56 dB
  - Excellent Blocking Performance (±10 MHz): 90 dB
  - Programmable Output Power up to +15 dBm (Sub-1 GHz) and +9 dBm at 2.4 GHz (Bluetooth low energy)
  - Single-Ended or Differential RF Interface
  - Suitable for Systems Targeting Compliance With Worldwide Radio Frequency Regulations
    - ETSI EN 300 220, EN 303 204 (Europe)
    - EN 300 440 Class 2 (Europe)
    - EN 300 328 (Europe)
    - FCC CFR47 Part 15 (US)
    - ARIB STD-T66 (Japan)
    - ARIB STD-T108 (Japan)

  - Wireless M-Bus (EN 13757-4) and IEEE® 802.15.4g PHY
- Tools and Development Environment
  - Full-Feature and Low-Cost Development Kits
  - Multiple Reference Designs for Different RF Configurations
  - Packet Sniffer PC Software
  - Sensor Controller Studio
  - SmartRF™ Studio
  - SmartRF Flash Programmer 2
  - IAR Embedded Workbench® for Arm
  - Code Composer Studio™ (CCS) IDE
  - CCS UniFlash

1.2 Applications

- 315-, 433-, 470-, 500-, 779-, 868-, 915-, 920-MHz and 2.4-GHz ISM and SRD Systems
- Low-Power Wireless Systems With 50-kHz to 5-MHz Channel Spacing
- Home and Building Automation
- Wireless Alarm and Security Systems
- Industrial Monitoring and Control
- Bluetooth low energy Beacon Management
- Bluetooth low energy Commissioning
- Smart Grid and Automatic Meter Reading
- Wireless Healthcare Applications
- Wireless Sensor Networks
- Active RFID
- IEEE 802.15.4g, IP-Enabled Smart Objects (6LoWPAN), Wireless M-Bus, KNX Systems, Wi-SUN™, and Proprietary Systems
- Energy-Harvesting Applications
- Electronic Shelf Label (ESL)
- Long-Range Sensor Applications
- Heat-Cost Allocators

1.3 Description

The CC1350 device is a cost-effective, ultra-low-power, dual-band RF device from Texas Instruments™ that is part of the SimpleLink™ microcontroller (MCU) platform. The platform consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz, Ethernet, Zigbee®, Thread, and host MCUs. These devices all share a common, easy-to-use development environment with a single core software development kit (SDK) and a rich tool set. A one-time integration of the SimpleLink platform enables users to add any combination of devices from the portfolio into their design, allowing 100 percent code reuse when design requirements change. For more information, visit www.ti.com/simplelink.

With very low active RF and MCU current consumption, in addition to flexible low-power modes, the device provides excellent battery life and allows long-range operation on small coin-cell batteries and in energy harvesting applications.

The CC1350 is a device in the CC13xx and CC26xx family of cost-effective, ultra-low-power wireless MCUs capable of handling both Sub-1 GHz and 2.4-GHz RF frequencies. The CC1350 device combines a flexible, very low-power RF transceiver with a powerful 48-MHz Arm® Cortex®-M3 microcontroller in a platform supporting multiple physical layers and RF standards. A dedicated Radio Controller (Cortex®-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus ensuring ultra-low power and flexibility to handle both Sub-1 GHz protocols and 2.4 GHz protocols (for example Bluetooth low energy). This enables the combination of a Sub-1 GHz communication solution that offers the best possible RF range together with a Bluetooth low energy smartphone connection that enables great user experience through a phone application. The Sub-1 GHz only device in this family is the CC1310.

The CC1350 device is a highly integrated, true single-chip solution incorporating a complete RF system and an on-chip DC/DC converter.

Sensors can be handled in a very low-power manner by a dedicated autonomous ultra-low-power MCU that can be configured to handle analog and digital sensors; thus the main MCU (Arm® Cortex®-M3) can maximize sleep time.

The power and clock management and radio systems of the CC1350 device require specific configuration and handling by software to operate correctly, which has been implemented in the TI-RTOS. TI recommends using this software framework for all application development on the device. The complete TI-RTOS and device drivers are offered free of charge in source code.

7.16 Datasheet Silicon EFR32

SILICON LABS

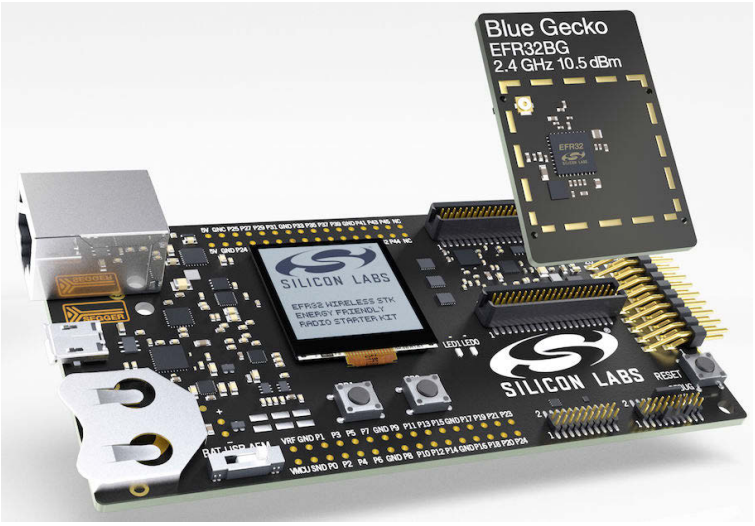
UG144: EFR32BG1 2.4 GHz 10.5 dBm Radio Board User's Guide

A Wireless Starter Kit with the BRD4100A Radio Board is an excellent starting point to get familiar with the EFR32™ Blue Gecko Wireless System-on-Chip, and it provides all necessary tools for developing a Silicon Labs wireless application.

BRD4100A is a plug-in board for the Wireless Starter Kit Mainboard. It is a complete reference design for the EFR32BG1 Wireless SoC, with matching network and a PCB antenna for 10.5 dBm output power in the 2.4 GHz band.

The Wireless Starter Kit Mainboard contains an on-board J-Link debugger with a Packet Trace Interface and a Virtual COM port, enabling application development and debugging of the attached radio board as well as external hardware. The mainboard also contains sensors and peripherals for easy demonstration of some of the EFR32's many capabilities.

This document describes how to use the BRD4100A Radio Board together with a Wireless Starter Kit Mainboard.



BRD4100A RADIO BOARD FEATURES

- EFR32BG1 Blue Gecko Wireless SoC with 256 kB Flash, and 32 kB RAM (EFR32BG1P232F256GM48)
  - 2.4 GHz integrated radio transceiver
  - 10.5 dBm output power
  - Inverted-F PCB antenna
- 8 Mbit low-power serial flash for over-the-air upgrades.

WIRELESS STK MAINBOARD FEATURES

- Advanced Energy Monitor
- Packet Trace Interface
- Virtual COM Port
- SEGGER J-Link on-board debugger
- External device debugging
- Ethernet and USB connectivity
- Silicon Labs Si7021 Relative Humidity and Temperature sensor
- Low Power 128x128 pixel Memory LCD
- User LEDs / Pushbuttons
- 20-pin 2.54 mm EXP header
- Breakout pads for Wireless SoC I/O
- CR2032 coin cell battery support

SOFTWARE SUPPORT

- Simplicity Studio™
- Energy Profiler
- Network Analyzer

ORDERING INFORMATION

- SLWSTK6020B
- SLWRB4100A

Bibliographie

[101] ETSI TS 136 101. *LTE ; Evolved Universal Terrestrial Radio Access (E-UTRA) ; User Equipment (UE) radio transmission and reception (3GPP TS 36.101 version 10.3.0 Release 10)*.

[115] ISO/IEC JTC 1. *Internet of Things (IoT) Preliminary Report 2014*. 2015.

[16a] "Groupe de travail 6Lo IETF." In : *International wireless communications and mobile computing conference (IWCMC)*. 2016.

[16b] "Groupe de travail 6Lo IETF." In : *international wireless communications and mobile computing conference (IWCMC)*. 2016.

[19a] *Groupe de travail T2TRG IRTF*. 2019. URL : http://www.irtf.org/t2trg.

[19b] *P2413 - Standard for an Architectural Framework for the Internet of Things (IoT)*. 2019. URL : https://standards.ieee.org/standard/2413-2019.html.

[219] Technical Report 21.915. *3rd Generation Partnership Project ; Technical Specification Group Services and System Aspects ; Release 15 Description ; Summary of Rel-15 Work Items (Release 15)*.

[Ada] ADAFRUIT. *Feather 32u4 RFM95 LoRa Radio- 868*. URL : https://www.adafruit.com/product/3078.

[Agg04] G. AGGELOU. "Mobile Ad Hoc Networks : From Wireless LANs to 4G Networks". In : 2004.

[Aky+02] Ian F AKYILDIZ, Weilian SU, Yogesh SANKARASUBRAMANIAM et Erdal CAYIRCI. "Wireless sensor networks : a survey". In : *Computer networks* 38.4 (2002), p. 393-422.

[AKY90] Yehuda AFEK, Shay KUTTEN et Moti YUNG. "Memory-efficient self stabilizing protocols for general networks". In : *International Workshop on Distributed Algorithms*. Springer. 1990, p. 15-28.

[All] LoRa ALLIANCE. *About the LoRaWAN® Standards*. URL : http://lora-alliance.org/lorawan-for-developers/.

[All15] Lora ALLIANCE. *A Technical Overview of LoRa® and LoRaWAN™*. 2015.

[AM14] Ashlyn ANTOO et A Rameez MOHAMMED. "EEM-LEACH : Energy efficient multi-hop LEACH routing protocol for clustered WSNs". In : *2014 international conference on control, instrumentation, communication and computational technologies (ICCICCT)*. IEEE. 2014, p. 812-818.

[Ama] AMAZON. *Alexa*. URL : https://alexa.amazon.fr/.

[Ame15] 4G AMERICAS. *Cellular Technologies Enabling the Internet of Things*. 2015.

[And14] M. Andersson. *Use case possibilities with Bluetooth low energy in IoT applications*. 2014.

[Appa] Apple. *Homekit*. URL : https://www.apple.com/fr/shop/accessories/all/homekit.

[Appb] Apple. *Homepod*. URL : https://www.apple.com/fr/homepod-2018/.

[Arn92] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants*. Etudes et Recherches en Informatique. Masson, 1992. ISBN : 9782225827464. URL : https://books.google.fr/books?id=bBhnAAAACAAJ.

[Ash+09] Kevin Ashton et al. "That 'internet of things' thing". In : *RFID journal* 22.7 (2009), p. 97-114.

[Ass] GSM Association. *NB-IoT DEPLOYMENT GUIDE to Basic Feature set Requirements*.

[AW04] Hagit Attiya et Jennifer Welch. *Distributed Computing : Fundamentals, Simulations and Advanced Topics*. USA : John Wiley et Sons, Inc., 2004. ISBN : 0471453242.

[AY07] Ameer Ahmed Abbasi et Mohamed Younis. "A survey on clustering algorithms for wireless sensor networks". In : *Computer communications* 30.14-15 (2007), p. 2826-2841.

[BA12] Deena M Barakah et Muhammad Ammad-uddin. "A survey of challenges and applications of wireless body area network (WBAN) and role of a virtual doctor server in existing architecture". In : *2012 Third International Conference on Intelligent Systems Modelling and Simulation*. IEEE. 2012, p. 214-219.

[Bas99] Stefano Basagni. "Distributed clustering for ad hoc networks". In : *Proceedings Fourth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*. IEEE. 1999, p. 310-315.

[BCV03] Lélia Blin, Alain Cournier et Vincent Villain. "An improved snap-stabilizing PIF algorithm". In : *Symposium on Self-Stabilizing Systems*. Springer. 2003, p. 199-214.

[BFH12] A. Bretto, A. Faisant et F. Hennecart. *Eléments de théorie des graphes*. Collection IRIS. Springer Paris, 2012. ISBN : 9782817802800. URL : https://books.google.fr/books?id=M-RFXwAACAAJ.

[BIO18] BioHarness | BIOPAC. *BioHarness*. 2018. URL : https://www2.meethue.com/fr-fr/decouvrir-hue.

[Biz15] BiztechAfrica. *stNet announces Africa's first dedicated M2M network and IoT developer academy*. 2015.

[BK07] Michel Barbeau et Evangelos Kranakis. *Principles of Ad-Hoc Networking*. Wiley Publishing, 2007. ISBN : 0470032901.

[Bov+18] Giampaolo Bovenzi, Domenico Ciuonzo, Valerio Persico, Antonio Pescapè et Pierluigi Salvo Rossi. "IoT-enabled distributed detection of a nuclear radioactive source via generalized score tests". In : *International symposium on signal processing and intelligent recognition systems*. Springer. 2018, p. 77-91.

[BPR11] Lélia Blin, Maria Gradinariu Potop-Butucaru et Stephane Rovedakis. "Self-stabilizing minimum degree spanning tree within one from the optimal degree". In : *Journal of Parallel and Distributed Computing* 71.3 (2011), p. 438-449.

[Bri16] Ken Briodagh. *Japan Opens New LoRaWAN Network for IoT Testing, IoT Evolution World*. 2016.

[CA06] Carlos De Morais Cordeiro et Dharma Prakash Agrawal. *Ad hoc & sensor networks : theory and applications*. World scientific, 2006.

[CAS09] CASAGRAS. *Coordination and Support Action for Global RFID-related Activities and Standardisation*. 2009.

[CCL03] Imrich Chlamtac, Marco Conti et Jennifer J-N Liu. "Mobile ad hoc networking : imperatives and challenges". In : *Ad hoc networks* 1.1 (2003), p. 13-64.

[Cha+17] Subhajit Chatterjee, Shreya Chatterjee, Soumyadeep Choudhury, Sayan Basak, Srijan Dey, Suparna Sain, Kali Shreyo Ghosal, Niket Dalmia et Sachet Sircar. "Internet of things and body area network-an integrated future". In : *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*. IEEE. 2017, p. 396-400.

[Cha82] E. J. H. Chang. "Echo Algorithms : Depth Parallel Operations on General Graphs". In : *IEEE Trans. Softw. Eng.* 8.4 (juill. 1982), p. 391-401. issn : 0098-5589. doi : 10.1109/TSE.1982.235573. url : http://dx.doi.org/10.1109/TSE.1982.235573.

[CNI18] CNIL. *Qu'est-ce ce qu'une donnée de santé ?* 2018.

[Com] European Commission. *Internet of Things Initiative | Projects | FP7-ICT | CORDIS |*.

[com] LoRa Alliance Technical committee. *Regional Parameters Workgroup.*

[Com17] International Electrotechnical Commission. *IEC role in the IoT, International Electrotechnical Commission*. 2017.

[com18] LoRa Alliance Technical committee. *LoRaWAN 1.0.3 Specification. LoRa Alliance.* 2018.

[Cra09] S. W. Lin B. Miller J. Durand G. Bleakley A. Chigani R. Martin B. Murphy M. Crawford. *Industrial Internet Reference Architecture version 1.7*. 2009.

[Cra17] P. Thubert C. Bormann L. Toutain R. Cragie. *IPv6 over Low-PowerWireless Personal Area Network (6LoWPAN) Routing Header*. 2017.

[DFG06] Vadim Drabkin, Roy Friedman et Maria Gradinariu. "Self-stabilizing wireless connected overlays". In : *International Conference On Principles Of Distributed Systems*. Springer. 2006, p. 425-439.

[DFG83] Edsger W. Dijkstra, W. H. J. Feijen et A. J. M. van Gasteren. "Derivation of a Termination Detection Algorithm for Distributed Computations". In : *Inf. Process. Lett.* 16.5 (1983), p. 217-219. doi : 10.1016/0020-0190(83)90092-3. url : https://doi.org/10.1016/0020-0190(83)90092-3.

[DH76] W. Diffie et M. Hellman. "New directions in cryptography". In : *IEEE Transactions on Information Theory* 22.6 (1976), p. 644-654. doi : 10.1109/TIT.1976.1055638.

[Die97] Reinhard Diestel. *Graph Theory*. 1997.

[Dij74] Edsger W. Dijkstra. "Self-stabilizing Systems in Spite of Distributed Control". In : *Commun. ACM* 17.11 (nov. 1974), p. 643-644. issn : 0001-0782. doi : 10.1145/361179.361202. url : http://doi.acm.org/10.1145/361179.361202.

[DIM91] Shlomi Dolev, Amos Israeli et Shlomo Moran. "Resource Bounds for Self Stabilizing Message Driven Protocols". In : *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*. PODC '91. Montreal, Quebec, Canada : ACM, 1991, p. 281-293. isbn : 0-89791-439-2. doi : 10.1145/112600.112624. url : http://doi.acm.org/10.1145/112600.112624.

[DLV08] Ajoy K Datta, Lawrence L Larmore et Priyanka Vemula. “Self-stabilizing leader election in optimal space”. In : *Symposium on Self-Stabilizing Systems*. Springer. 2008, p. 109-123.

[DM12] Charalampos Doukas et Ilias Maglogiannis. “Bringing IoT and cloud computing towards pervasive healthcare”. In : *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE. 2012, p. 922-926.

[Dre08] Falko Dressler. *Self-organization in sensor and actor networks*. John Wiley & Sons, 2008.

[DS80] Edsger W Dijkstra et Carel S Scholten. “Termination detection for diffusing computations”. In : *Information Processing Letters* 11.1 (1980), p. 1-4.

[DT08] Ali Dada et Frédéric Thiesse. “Sensor applications in the supply chain : the example of quality-based issuing of perishables”. In : (2008), p. 140-154.

[Dwo05] Morris Dworkin. *NIST Special Publication 800-38B Recommendation for Block Cipher Modes of Operation : The CMAC Mode for Authentication.* 2005.

[Ech18] Echelon. *Echelon - Street Lighting*. 2018. url : https://www.echelon.com/applications/pl-rf-outdoor-lighting.

[Ent18] Alcatel Lucent Enterprise. *The Internet of Things in Transportation*. 2018.

[Eri18] Ericsson. “Massive IoT e-healths”. In : (2018). url : https://doi.org/10.1080/23738871.2017.1366536.

[Eva11] Dave Evans. “The internet of things : How the next evolution of the internet is changing everything”. In : *CISCO white paper* 1.2011 (2011), p. 1-11.

[EWB87] Anthony Ephremides, Jeffrey E Wieselthier et Dennis J Baker. “A design concept for reliable mobile radio networks with frequency hopping signaling”. In : *Proceedings of the IEEE* 75.1 (1987), p. 56-73.

[Fea16] Nicholas Fearn. *Orange deploys LoRa network in thousands of French towns. Internet of Business*. 2016.

[FHN20] Olivier Flauzac, Joffrey Herard et Florent Nolot. “Synchronization Solution to Optimize Power Consumption in Linear Sensor Network”. In : *International Conference on Fog and Mobile Edge Computing (FMEC)*. Paris, France, 2020, p. 295-300. doi : 10.1109/FMEC49853.2020.9144887. url : https://hal.archives-ouvertes.fr/hal-02909570.

[FL11] Jackie Fenn et Hung LeHong. “Hype cycle for emerging technologies”. In : *Gartner, Stamford* (2011).

[Fla+20a] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. “A Fault Tolerant LoRa/LoRaWAN Relay Protocol Using LoRaWAN Class A Devices”. In : *International Conference on Ad-Hoc Networks and Wireless (ADHOC-NOW)*. Bari, Italy, 2020, p. 295-302. doi : 10.1007/978-3-030-61746-2_22. url : https://hal.univ-reims.fr/hal-02978424.

[Fla+20b] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. “A low power LoRa-LoRaWan relay function with a single input, single output device”. In : *International Conference on Embedded Wireless Systems and Networks (EWSN)*. Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks. Lyon, France, 2020, p. 283-288. url : https://hal.univ-reims.fr/hal-02852171.

[Fla+20c] Olivier Flauzac, Joffrey Herard, Florent Nolot et Philippe Cola. "LLWURP : LoRa/LoRaWAN Uniform Relay Protocol with a Single Input, Single Output Device". In : *Future Technologies Conference (FTC) 2020*. T. 2. San Francisco, United States, 2020, p. 862-876. doi : 10.1007/978-3-030-63089-8_56. url : https://hal.univ-reims.fr/hal-02989181.

[Fla+20d] Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. "Comparison between different platform using a Uniform Relay Protocol in Linear Sensor Network". In : *International Conference on the Internet of Things Companion (IoT)*. Malmö, Sweden : ACM, 2020, p. 1-4. doi : 10.1145/3423423.3423462. url : https://hal.univ-reims.fr/hal-02978428.

[Fla+20e] Olivier Flauzac, Joffrey Herard, Florent Nolot et Florian Desrumaux. "Comparison Between Different Types of Sensor Architecture Using a Uniform Relay Protocol". In : *International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2020 8th International Conference on Wireless Networks and Mobile Communications (WINCOM). Reims, France : IEEE, 2020, p. 1-6. doi : 10.1109/WINCOM50532.2020.9272428. url : https://hal.univ-reims.fr/hal-03042336.

[Flo+08] Christian Floerkemeier, Marc Langheinrich, Elgar Fleisch, Friedemann Mattern et Sanjay E Sarma. *The Internet of Things : First International Conference, IOT 2008, Zurich, Switzerland, March 26-28, 2008, Proceedings*. T. 4952. springer, 2008.

[for] marko23 membre du forum. *Forum ST Community*. url : https://community.st.com/s/global-search/low%20power%20discovery.

[Fos+11] Luca Foschini, Tarik Taleb, Antonio Corradi et Dario Bottazzi. "M2M-based metropolitan platform for IMS-enabled road traffic management in IoT". In : *IEEE Communications Magazine* 49.11 (2011), p. 50-57.

[Fre+18] James Freed, Charles Lowe, Gerd Flodgren, Rachel Binks, Kevin Doughty et Jyrki Kolsi. "Telemedicine : Is it really worth it ? A perspective from evidence and experience." In : *Journal of innovation in health informatics* 25.1 (2018), p. 014-018.

[Fri13] Mounir Frikha. *Ad Hoc Networks : Routing, QoS and Optimization*. John Wiley et Sons, 2013.

[Gac+12] Diego Gachet, Manuel de Buenaga, Fernando Aparicio et Vıctor Padrón. "Integrating internet of things and cloud computing for health services provisioning : The virtual cloud carer project". In : *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE. 2012, p. 918-921.

[Gal77] Robert G Gallager. "Finding a leader in a network with O (e)+ O (n log n) messages". In : *Unpublished Note* (1977).

[Gho+12] Avik Ghose, Provat Biswas, Chirabrata Bhaumik, Monika Sharma, Arpan Pal et Abhinav Jha. "Road condition monitoring and alert application : Using in-vehicle smartphone as internet-connected sensor". In : *2012 IEEE international conference on pervasive computing and communications workshops*. IEEE. 2012, p. 489-491.

[Gib85] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985. isbn : 9780521288811. url : https://books.google.fr/books?id=Be6t04pgggwC.

[Gil+13] Ekaterina Gilman, Oleg Davidyuk, Xiang Su et Jukka Riekki. "Towards interactive smart spaces". In : *Journal of Ambient Intelligence and Smart Environments* 5.1 (2013), p. 5-22.

[GKS18a] Oscar Garcia-Morchon, Sandeep Kumar et Mohit Sethi. "State-of-the-Art and Challenges for the Internet of Things Security". In : *Online]. https ://tools. ietf. org/html/draft-irtf-t2trg-iot-seccons-15* (2018).

[GKS18b] Oscar Garcia-Morchon, Sandeep Kumar et Mohit Sethi. "State-of-the-Art and Challenges for the Internet of Things Security". In : *Online]. https ://tools. ietf. org/html/draft-irtf-t2trg-iot-seccons-16* (2018).

[GM90] M. Gondran et M. Minoux. *Graphes et algorithmes*. Collection de la Direction des Etudes et Recherches d'Electricité de France. Eyrolles, 1990. url : https://books.google.fr/books?id=fg0MngEACAAJ.

[GM91] M. G. Gouda et N. J. Multari. "Stabilizing communication protocols". In : *IEEE Transactions on Computers* 40.4 (avr. 1991), p. 448-458. issn : 0018-9340. doi : 10.1109/12.88464.

[Goo] Google. *Google Home*. url : https://store.google.com/product/google_home.

[gov] UK governmnent. *UK CITE – UK Connected Intelligent Transport Environment*.

[gov12] UK governmnent. *Intelligent Transport Systems in the UK*. 2012.

[HCB00] Wendi Rabiner Heinzelman, Anantha Chandrakasan et Hari Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks". In : *Proceedings of the 33rd annual Hawaii international conference on system sciences*. IEEE. 2000, 10-pp.

[Hec07] Oliver M Heckmann. *The competitive Internet service provider : network architecture, interconnection, traffic engineering and network design*. John Wiley et Sons, 2007.

[Hél+86] Jean-Michel Hélary, Aomar Maddi, Noël Plouzeau et Michel Raynal. "Parcours et apprentissage dans un réseau de processus communicants". Thèse de doct. INRIA, 1986.

[Hik18] Hikob. *Systèmes d'acquisition de données stationnaires par Hikob*. 2018. url : https://www.hikob.com/instant/..

[HJL18] Zijiang Hao, Raymond Ji et Qun Li. "Fastpay : A secure fast payment method for edge-IoT platforms using blockchain". In : *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE. 2018, p. 410-415.

[HPR88] Jean-Michel Helary, Noel Plouzeau et Michel Raynal. "A distributed algorithm for mutual exclusion in an arbitrary network". In : *The Computer Journal* 31.4 (1988), p. 289-295.

[htt] https ://www.oasis-open.org/. *OASIS | Advancing open standards for the information society*.

[htt12] https ://tools.ietf.org/wg/6lo/. *SÉRIE Y : INFRASTRUCTURE MONDIALE DE L'INFORMATION, PROTOCOLE INTERNET ETRÉSEAUX DE PROCHAINE GÉNÉRATION*. 2012.

[I S09] R. Ma Y.W. Kim E. Wlak C. Harmon P. Chartier P. Guillemin D. Armstrong I. Smith K. Sakamura A. Furness. "RFID and the Inclusive Model for the Internet of Things". In : *CASAGRAS*. 2009.

[IHS16] Markit IHS. *Rapid Expansion Projected for Smart Home devices*. 2016.

[Ily17] Mohammad Ilyas. *The handbook of ad hoc wireless networks*. CRC press, 2017.

[IMM08] Krzysztof Iniewski, Carl McCrosky et Daniel Minoli. *Network infrastructure and architecture : designing high-availability networks*. John Wiley et Sons, 2008.

[Ini18] Krzysztof Iniewski. *Internet Networks : Wired, Wireless, and Optical Technologies*. CRC Press, 2018.

[Ins] Texas Instrument. *LAUNCHXL-CC1350 SimpleLink™ Dual-Band CC1350 Wireless MCU LaunchPad Development Kit*. url : https://www.ti.com/tool/LAUNCHXL-CC1350#tech-docs.

[IoT19] IoT.Business.News. *LoRa Alliance : interview with CEO, Donna Moore MWC 2019*. 2019.

[Ism18] Mikael Gidlund Ismail Butun Nuno Pereira. *Demystifying Security of LoRaWAN v1.1*. 2018. url : https://doi.org/10.20944/preprints201811.0531.v1.

[ISO] IEC - ISO/IEC. *IEC - ISO/IEC JTC 1/SC 25*.

[ISO17] IEC - ISO/IEC. *International Electrotechnical Commission, " Orchestrating infrastructure for sustainable Smart Cities"*. 2017.

[ITU15] ITU.T. *Short range narrow-band digital radiocommunication transceivers – PHY, MAC, SAR and LLC layer specifications*. 2015.

[JJK17] RC Jisha, Aiswarya Jyothindranath et L Sajitha Kumary. "Iot based school bus tracking and arrival time prediction". In : *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE. 2017, p. 509-514.

[JM11] Colette Johnen et Fouzi Mekhaldi. "Self-stabilization versus robust self-stabilization for clustering in ad-hoc network". In : *European Conference on Parallel Processing*. Springer. 2011, p. 117-129.

[KA] Kittelson et Inc. Associates. *ParkDC : Penn Quarter/Chinatown*.

[KH16] Nicolas Sornin Miguel Luis Thomas Eirich Thorsten Kramp et Olivier Hersent. *LoRaWAN Specification. LoRa Alliance*. 2016.

[Kir15] Richard Kirk. "Cars of the future : the Internet of Things in the automotive industry". In : *Network Security* 2015.9 (2015), p. 16-18.

[Kon+12] N. Kong, Park Jungsoo, N. Crespi, G. Lee et Ilyoung Chong. "The Internet of Things - Concept and Problem Statement". In : 2012.

[KP93] Shmuel Katz et Kenneth J. Perry. "Self-stabilizing Extensions for Message-passing Systems". In : *Distrib. Comput.* 7.1 (nov. 1993), p. 17-26. issn : 0178-2770. doi : 10.1007/BF02278852. url : http://dx.doi.org/10.1007/BF02278852.

[Kum08] Sarkar Subir Kumar. *Ad hoc mobile wireless networks [Texte imprimé] : principles, protocols, and applications / Subir Kumar Sarkar, T.G. Basavaraju, C. Puttamadappa*. eng. New York, 2008.

[Kuo11] Mu-Hsing Kuo. "Opportunities and challenges of cloud computing to improve health care services". In : *Journal of medical Internet research* 13.3 (2011), e67.

[KW07] Holger Karl et Andreas Willig. *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[Lab] Silicon Labs. *SLWSTK6020B EFR32BG*. url : https://www.silabs.com/development-tools/wireless/bluetooth/efr32bg-bluetooth-soc-starter-kit.

[Lam18] Good Night Lamp. *Good Night Lamp*. 2018. url : http://goodnightlamp.com.

[Lav95] C. Lavault. *Evaluation des algorithmes distribues : analyse, complexite, methodes*. Hermès, 1995. url : https://books.google.fr/books?id=Nm9_AAAACAAJ.

[Le 77] Gérard Le Lann. “Distributed Systems-Towards a Formal Approach.” In : *IFIP congress*. T. 7. Toronto. 1977, p. 155-160.

[Lec18] Lechal. *Lechal Wearble Tech and GPS Navigation Device*. 2018. url : http://www.lechal.com/.

[LI06] JunHyuk Song Radha Poovendran Jicheol Lee et Tetsua Iwata. *The AES-CMAC Algorithm. RFC 4493*. 2006.

[Liu+19] Yi Liu, Chao Yang, Li Jiang, Shengli Xie et Yan Zhang. “Intelligent edge computing for IoT-based energy management in smart cities”. In : *IEEE Network* 33.2 (2019), p. 111-117.

[Liv18] Livehoods. *Livehoods*. 2018. url : http://livehoods.org/.

[LN16] IEEE Standard for Local et Metropolitan Area Networks. *Low-Rate Wireless Personal Area Networks*. 2016.

[LNS09] Hai Liu, Amiya Nayak et Ivan Stojmenović. “Fault-tolerant algorithms/protocols in wireless sensor networks”. In : *Guide to wireless sensor networks*. Springer, 2009, p. 261-291.

[Loc] Lockitron. *Lockitron.* url : https://lockitron.com/.

[LP09] Theofanis P Lambrou et Christos G Panayiotou. “A survey on routing techniques supporting mobility in sensor networks”. In : *2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks*. IEEE. 2009, p. 78-85.

[LPS03] Philippe Ph. Lacomme, Christian Prins et Marc Sevaux. *Algorithmes de Graphes*. ISBN 2-212-11385-4. Eyrolles, 2003, p. 425. url : https://hal.archives-ouvertes.fr/hal-00009111.

[LT87] Nancy A Lynch et Mark S Tuttle. *Hierarchical Correctness Proofs for Distributed Algorithms.* Rapp. tech. MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1987.

[LUI14] UIT-T SECTEUR DE LA NORMALISATIONDES TÉLÉCOMMUNICATIONSDE L’UIT. *SERIES Y : GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTSAND NEXT-GENERATION NETWORKS*. 2014.

[LWJ05] Hai Liu, Peng-Jun Wan et Xiaohua Jia. “Fault-tolerant relay node placement in wireless sensor networks”. In : *International Computing and Combinatorics Conference*. Springer. 2005, p. 230-239.

[Lyn96] Nancy A. Lynch. *Distributed Algorithms*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1996. isbn : 1558603484.

[Mad+15] Somayya Madakam, Vihar Lake, Vihar Lake, Vihar Lake et al. “Internet of Things (IoT) : A literature review”. In : *Journal of Computer and Communications* 3.05 (2015), p. 164.

[Mae85] Mamoru Maekawa. “An algorithm for mutual exclusion in decentralized systems”. In : *ACM Transactions on Computer Systems (TOCS)* 3.2 (1985), p. 145-159.

[Map17] Carsten Maple. “Security and privacy in the internet of things”. In : *Journal of Cyber Policy* 2.2 (2017), p. 155-184. doi : 10.1080/23738871.2017.1366536. eprint : https://doi.org/10.1080/23738871.2017.1366536. url : https://doi.org/10.1080/23738871.2017.1366536.

[Mar11] Luigi Martirano. “A smart lighting control to save energy”. In : *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*. T. 1. IEEE. 2011, p. 132-138.

[mar16] martCitiesWorld. *Semtech LoRa chosen for new IoT network in New Zealand.* 2016.

[Mat17] R. Materese. *Internet of Things (IoT*. 2017.

[MBR15] Roberto Minerva, Abyi Biru et Domenico Rotondi. "Towards a definition of the Internet of Things (IoT)". In : *IEEE Internet Initiative* 1.1 (2015), p. 1-86.

[ME18] Rim Marah et Abdelaaziz El Hibaoui. "Algorithms for smart grid management". In : *Sustainable cities and society* 38 (2018), p. 627-635.

[Mic] Microship. *SAMR34*. url : https://www.microchip.com/en-us/products/wireless-connectivity/lora-technology/sam-r34-r35.

[Mim18] Mimo. *Mimo*. 2018. url : https://www.mimobaby.com/.

[Mis83] Jayadev Misra. "Detecting termination of distributed computations using markers". In : *Proceedings of the second annual ACM symposium on Principles of distributed computing*. Citeseer. 1983, p. 290-294.

[MSH11] M.SHAKIR. *Wireless Sensor Networks*. Lambert Academic Publisjing, 2011.

[Naf+18] Nazmus S Nafi, Khandakar Ahmed, Mark A Gregory et Manoj Datta. "Software defined neighborhood area network for smart grid applications". In : *Future Generation Computer Systems* 79 (2018), p. 500-513.

[Nes] Nest. *Thermostats Nest*. url : https://nest.com/fr/thermostats/.

[NGK16] Keith E Nolan, Wael Guibene et Mark Y Kelly. "An evaluation of low power wide area network technologies for the Internet of Things". In : *2016 international wireless communications and mobile computing conference (IWCMC)*. IEEE. 2016, p. 439-444.

[Nok19] Nokia. *Smart city | Nokia Networks*. 2019.

[Nor] Nordic. *nrf9160*. url : https://www.nordicsemi.com/Products/Low-power-cellular-IoT/nRF9160.

[Nuk] Nuki. *Nuki SMartLock*. url : https://nuki.io/fr/.

[OA16] Orange et Actility. *LoRa Device Developer Guide – Orange Connected Objects and Partnerships*. 2016.

[Obj] Objenious. *Objenious accompagne Airbus dans l'adoption de l'internet des objets grâce au réseau dédié*. url : https://objenious.com/wp-content/uploads/2019/02/Objenious-accompagne-Airbus-dans-l%E2%80%99adoption-de-l%E2%80%99Internet-des-Objets-gr%C3%A2ce-au-r%C3%A9seau-%C2%AB-d%C3%A9di%C3%A9-%C2%BB-.pdf.

[Phi18] Éclairage intelligent Philips Hue | Philips Hue. *Hue*. 2018. url : https://www2.meethue.com/fr-fr/decouvrir-hue.

[PK00] Gregory J Pottie et William J Kaiser. "Wireless integrated network sensors". In : *Communications of the ACM* 43.5 (2000), p. 51-58.

[PyC] PyCom. *LoPy*. url : https://pycom.io/.

[R+13] Stachel Joshua R, Ervin Sejdić, Ogirala Ajay et Marlin H Mickle. "The impact of the internet of Things on implanted medical devices including pacemakers, and ICDs". In : *2013 IEEE international instrumentation and measurement technology conference (I2MTC)*. IEEE. 2013, p. 839-844.

[RA80] Glenn Ricart et Ashok K Agrawala. *An Algorithm for Mutual Exclusion in Computer Networks.* Rapp. tech. MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE, 1980.

[Rab80] Michael O Rabin. "Probabilistic algorithm for testing primality". In : *Journal of number theory* 12.1 (1980), p. 128-138.

[Ran83] SP Rana. "A distributed solution of the distributed termination problem". In : *Information Processing Letters* 17.1 (1983), p. 43-46.

[Ray13] Michel Raynal. *Distributed Algorithms for Message-Passing Systems*. Springer, 2013, p. I-XXX, 1-500. url : https://hal.inria.fr/hal-00922219.

[Ray85] M. Raynal. *Algorithmes distribués et protocoles*. Eyrolles, 1985. url : https://books.google.fr/books?id=FNePOAAACAAJ.

[Ray92] M. Raynal. *Synchronisation et etat global dans les systemes repartis : tome 2 d'une introduction aux principes des systemes repartis*. Collection de la Direction des etudes et recherches d'Electricite de France. Eyrolles, 1992. url : https://books.google.fr/books?id=oJkWGQAACAAJ.

[RB12] GK Ragesh et K Baskaran. "An overview of applications, standards and challenges in futuristic wireless body area networks". In : *International Journal of Computer Science Issues (IJCSI)* 9.1 (2012), p. 180.

[RH88] M. Raynal et J.M. Hélary. *Synchronisation et contrôle des systèmes et des programmes répartis*. Collection de la Direction des études et recherches d'Electricité de France. Eyrolles, 1988. isbn : 9782212082746. url : https://books.google.fr/books?id=LnkOvAEACAAJ.

[Rob18] Insight Robotics. *Forestry-Focused Risk Management*. 2018. url : https://www.insightrobotics.com/en/.

[RW70] Lawrence G. Roberts et Barry D. Wessler. "Computer Network Development to Achieve Resource Sharing". In : *Proceedings of the May 5-7, 1970, Spring Joint Computer Conference*. AFIPS '70 (Spring). Atlantic City, New Jersey : Association for Computing Machinery, 1970, p. 543-549. isbn : 9781450379038. doi : 10.1145/1476936.1477020. url : https://doi.org/10.1145/1476936.1477020.

[SBP13] Subir Kumar Sarkar, TG Basavaraju et C Puttamadappa. *Ad Hoc Mobile Wireless Networks : Principles, Protocols, and Applications*. CRC Press, 2013.

[Sen18] SenseAwar. *SenseAwar*. 2018. url : https://www.senseaware.com/fr/.

[Sho18] Shoal. *Shoal.* 2018. url : https://roboshoal.com/e.

[Sin91] Mukesh Singhal. "A class of deadlock-free Maekawa-type algorithms for mutual exclusion in distributed systems". In : *Distributed Computing* 4.3 (1991), p. 131-138.

[SL85] Fred B. Schneider et Leslie Lamport. "Paradigms for Distributed Programs". In : *Distributed Systems : Methods and Tools for Specification, An Advanced Course, April 3-12, 1984 and April 16-25, 1985 Munich*. London, UK, UK : Springer-Verlag, 1985, p. 431-480. isbn : 3-540-15216-4. url : http://dl.acm.org/citation.cfm?id=647435.726399.

[Sma16] SmartCitiesWorld. *IoT connectivity for 100 million homes in China.* 2016.

[SMZ07] Kazem Sohraby, Daniel Minoli et Taieb Znati. *Wireless sensor networks : technology, protocols, and applications*. John wiley & sons, 2007.

[Sol18a] BigBelly Solar. *Transforming Trash Collection Operations*. 2018. url : https://www.telit.com/resources/case-studies/bigbelly-solar/.

[Sol18b] All Traffic Solutions. *All Traffic Solutions IoT Solutions for Smart Parking and Transportation Mgmt.* 2018. URL : http://www.alltrafficsolutions.com/.

[Sor17] Nicolas Sornin. *LoRaWAN 1.1 Specification. LoRa Alliance, version 1.1, 11/10/2017.* 2017.

[SPD18] Mariusz Slabicki, Gopika Premsankar et Mario Di Francesco. "Adaptive configuration of LoRa networks for dense IoT deployments". In : *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium.* IEEE. 2018, p. 1-9.

[STa] ST. *LoRa® IoT tracker Steval STRTK01.* URL : https://www.st.com/en/evaluation-tools/steval-strkt01.html.

[STb] ST. *STM32L0 Discovery kit LoRa.* URL : https://www.st.com/en/evaluation-tools/b-l072z-lrwan1.html#overview&secondary=st_description_sec-nav-tab.

[STc] NIST | National Institute of Standards et Technology. *"Nist".* URL : https://www.nist.gov/.

[ST02] National Institute Of Standards et Technology. *NIST FIPS 197 Specification forthe Advanced Encryption Standard (AES).* 2002.

[Sta12] Statista. "IoT : number of connected devices worldwide 2012–2025". In : (2012).

[Swa+19] K Narendra Swaroop, Kavitha Chandu, Ramesh Gorrepotu et Subimal Deb. "A health monitoring system for vital signs using IoT". In : *Internet of Things* 5 (2019), p. 116-129.

[Sys17] Cantaloupe Systems. *Cantaloupe Systems.* 2017. URL : https://www.cantaloupesys.com/.

[Tad18] Tado. *Tado.* 2018. URL : https://www.tado.com/fr.

[Tan96] Andrew S. Tanenbaum. *Computer Networks (3rd Ed.)* USA : Prentice-Hall, Inc., 1996. ISBN : 0133499456.

[Tel01] Gerard Tel. *Introduction to Distributed Algorithms.* 2nd. New York, NY, USA : Cambridge University Press, 2001. ISBN : 0521794838.

[TRB12] Andrew Tinka, Mohammad Rafiee et Alexandre M Bayen. "Floating sensor networks for river studies". In : *IEEE Systems Journal* 7.1 (2012), p. 36-49.

[VD08] Jean-Philippe Vasseur et Adam Dunkels. "Ip for smart objects". In : *White Paper* 1 (2008).

[Whi+10] David L White, Samuel Esswein, Jason O Hallstrom, Farha Ali, Shashank Parab, Gene Eidson, Jill Gemmill et Christopher Post. "The Intelligent River© : Implementation of Sensor Web Enablement technologies across three tiers of system architecture : Fabric, middleware, and application". In : *2010 International Symposium on Collaborative Technologies and Systems.* IEEE. 2010, p. 340-348.

[Woe10] H. Sundmaeker P. Guillemin P. Friess S. Woelfflé. "Vision and Challenges for Realising the Internet of Things". In : CERP-IoT, 2010.

[Wu+10] Miao Wu, Ting-Jie Lu, Fei-Yang Ling, Jing Sun et Hui-Ying Du. "Research on the architecture of Internet of Things". In : *2010 3rd international conference on advanced computer theory and engineering (ICACTE).* T. 5. IEEE. 2010, p. V5-484.

[XCL10] Yang Xiao, Hui Chen et Frank Haizhon Li. *Handbook on sensor networks.* World Scientific, 2010.

[YH11] Xiaojing Ye et Junwei Huang. "A framework for cloud-based smart home". In : *Proceedings of 2011 international conference on computer science and network technology.* T. 2. IEEE. 2011, p. 894-897.

[ZXM07] Weiyi Zhang, Guoliang Xue et Satyajayant Misra. “Fault-tolerant relay node placement in wireless sensor networks : Problems and algorithms”. In : *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE. 2007, p. 1649-1657.

[ZYZ11] Min Zhang, Tao Yu et Guo Fang Zhai. “Smart transport system based on “The Internet of Things””. In : *Applied mechanics and materials*. T. 48. Trans Tech Publ. 2011, p. 1073-1076.

Structuration autonome des réseaux IoT de type LoRa / LoRaWAN

Depuis l'extension et la démocratisation de l'Internet des Objets ces dernières années, les problématiques de couverture et d'accès aux réseaux opérateurs se développent. Ces réseaux opérateurs, qui sont des réseaux avec infrastructures, visent à gérer un ensemble d'appareils connectés tout en prenant en compte l'aspect énergétique. Le nombre de ceux-ci ne fait que croître depuis plusieurs années. Des défauts de couverture peuvent apparaître, ainsi des appareils ne peuvent se connecter à l'infrastructure mise en place. Dans cette thèse, nous nous focalisons sur le cas du LoRaWAN. Dans le cadre de cette thèse, nous concevons une solution de relais LoRa/LoRaWAN permettant d'étendre la couverture d'un réseau LoRaWAN sans ajouter de composant spécifique au réseau ni interférer sur le déploiement et l'infrastructure de l'opérateur. Nous redéfinissons ainsi les acteurs qu'étaient les end-devices et les antennes LoRaWAN. Ils deviennent des nœuds relais, des nœuds isolés, des nœuds totalement isolés, et des antennes LoRaWAN. Les échanges entre les nœuds isolés et des relais se font en exploitant le protocole LoRa, ceux entre le relais et l'infrastructure en utilisant l'architecture LoRaWAN. Nous proposons dans un premier temps un algorithme original de relais (LLRP), puis des solutions tolérantes aux fautes (FT-LLWURP), et sécurisée (S-LLWURP). Enfin, nous proposons, une solution de structuration de nœuds isolés en réseau maillé LoRa (MESH-FT-S-LLWURP), et assurons le relais des informations vers un réseau LoRaWAN. Nous validons nos solutions et en montrons l'efficacité des solutions proposées à chaque étape par le biais de simulations à grande échelle et d'expérimentations de longue durée sur l'infrastructure de Objenious par Bouygues Télécom.

Algorithme distribué, réseau ad hoc, Internet des objets, passerelle IoT, LoRa, LoRaWAN

Autonomous structuring of IoT networks type of LoRa / LoRaWAN

Since the extension and popularization of the Internet of Things over the previous few years, the issues of coverage and access to an operator networks are growing. These operator networks, which are networks with infrastructure, want to manage a set of connected devices while taking into account the energy aspect. The number of these networks has been fastly growing for several years. Defects of coverage can appear, so some devices cannot connect to the infrastructure set up. In this thesis, we focus on the case of LoRaWAN. We designed a LoRa/LoRaWAN relay solution allowing us to extend the coverage of a LoRaWAN network without adding any specific component to the network nor interfering on the deployment and infrastructure of the operator. In order to manage this, firstly we redefined the actors that were the end-devices and the LoRaWAN antennas. They become relay nodes, isolated nodes, fully isolated nodes, and LoRaWAN antennas. Exchanges between isolated nodes and relays are done using the LoRa protocol, those between the relay and the infrastructure using the LoRaWAN architecture. We first propose an original relay algorithm (LLRP), then fault tolerant (FT-LLWURP) and secure (S-LLWURP) solutions. Finally, we propose a solution for structuring autonomously isolated nodes into a LoRa mesh network (MESH-FT-S-LLWURP), and relay the information to a LoRaWAN network. We validate our solutions and showed the increase in efficiency regarding the power consumption, number of messages that transit inside the system, of the proposed solutions at each step through large-scale simulations and long-term experiments of the order of one year, using the infrastructure of Objenious.

Distributed algorithm, Adhoc networks, Internet of things, IoT gateway, LoRa, LoRaWAN

Discipline : INFORMATIQUE

Spécialité : informatique

Université de Reims Champagne-Ardenne

CRESTIC - EA 3804

BP 1039 – 51687 Reims CEDEX 2