Université Paris-Est-Créteil

UFR des sciences et technologie

MSINF1

Ce projet a été partiellement financé par la région

Paris Ile-de-France Region via une subvention du DIM RFSI (projet EPINE).

Rapport de stage

Energy and Performance Issues for a NEtwork in a Smart-city: EPINES

Présenté par :

Mr. KIES Akram-Walid

Encadrants :

Mme. Lynda Mokdad

Mr. Jean-Michel Fourneau

Remerciements

Au terme de ce travail, Je tiens à exprimer ma profonde gratitude et mes sincères remerciements à mes promoteurs, Madame Lynda Mokdad, Monsieur JeanMichel Fourneau, pour tout le temps qu'ils nous ont consacré et pour la qualité de leur suivi durant toute la période de mon stage de Master 1.

A nos frères, sœurs, familles et ami(e)s pour leur encouragement infini tout au long de ce travail.

Mes remerciements vont enfin à toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Introduction

La quatrième révolution industrielle (Internet of Things) a besoin de réseaux performants. Tous les acteurs des télécommunications sont appelés à jouer un rôle dans la mise en place de ces nouvelles technologies. En France, deux réseaux sont particulièrement mis en avant : SigFox et LoRa.

LoRa et son architecture réseau LoRaWAN est actuellement la solution IoT la plus prometteuse sur la bande sans licence avec une procédure de connectivité simplifiée sur des connexions sans fil longue portée. LoRaWAN présente des particularités uniques, notamment la technique de modulation à spectre étalé chirp, les limitations réglementaires sur le cycle de service radio et l'utilisation du protocole ALOHA. La topologie du réseau de départ est utilisée pour la transmission de données, dans laquelle une passerelle relaie les messages entre un serveur de réseau (NS) et des périphériques finaux. La transmission entre les terminaux et la passerelle est possible sur l'un des sous-canaux disponibles et avec l'un des 6 facteurs d'étalement. Une collision se produit lorsque deux ou plusieurs transmissions LoRa se chevauchent au niveau du récepteur. Une collision est provoquée par la sélection du même canal et du même facteur d'étalement (SF) par différents dispositifs, avec chevauchement temporel. De plus, même avec des SF différents, une collision entre des signaux sur le même sous-canal se produira en raison de l'orthogonalité imparfaite des SF, appelée collision inter SF. Inversement, s'il y a des transmissions simultanées sur les mêmes ressources radio (le même SF et le même sous-canal), la passerelle est en mesure de recevoir avec succès l'une d'entre elles si son rapport signal sur interférence et bruit (SINR) est supérieur à un certain seuil (par exemple 6 dB) pour tout SF. Ce dernier est considéré comme un effet de capture (CE) et sera également pris en compte.

Durant toute la période de ce stage, nous avons fait une étude théorique des réseaux LoRa.
Ensuite, nous avons détaillé les simulateurs des réseaux LoRa qui sont déjà déployés : LoRaMAB : A Flexible Simulator for Decentralized Learning Resource Allocation in IoT Networks, IoT scheduling for higher throughput and lower transmission power, Experimental Evaluation of LoRaWAN in NS-3, ainsi que une étude comparative des simulateurs open source. Nous nous attarderons sur les simulateurs LoRaSim, FREE. Finalement, nous introduirons le nouveau simualteur « AkramSim ».

1. Architecture LoRa

1.1. Présentation générale

LoRaWAN appartient à la catégorie des LPWAN (Low Power Wide-Area Network), réseaux basse consommation d'énergie, longue portée, adaptés aux objets connectés dont l'application requiert une autonomie importante. Ils utilisent les bandes de fréquences à usage libre ISM, partagées avec d'autres technologies sans-fil. Ils sont donc contraints au respect de règles d'utilisation définies, notamment en ce qui concerne la puissance d'émission, le rapport cyclique et la bande passante [1].

Les cas d'usage les plus courants des réseaux LPWAN sont les smart cities, les industries connectées et la mesure de données en milieu isolé, par exemple agricoles et météorologiques [1].

1.2. LORA ?

LoRa, pour *Long Range*, est une couche physique développée par Semtech qui permet des communications sans fil longue distance. LoRaWAN définit le protocole de communication et l'architecture du réseau.

LoRa utilise principalement la bande de fréquence 868MHz en Europe et emploie une modulation à étalement de spectre qui offre d'excellentes performances, une portée importante (plus de 15km en moyenne, record établi à 702km!) avec un signal robuste et une sensibilité accrue côté récepteur. Le débit est relativement faible, variant entre 300bps et 50kbps selon le facteur d'étalement.

Cette technologie utilise à la fois les fréquences radio libre 868 MHz et Internet. Peu gourmande en débit et en énergie, elle a l'avantage d'être très économique pour l'utilisateur final et présente, de plus, une excellente capacité de pénétration des bâtiments, caves et sous-sols. LoRa intègre le domaine de l'Internet des Objets (Iot ou IdO) par le champ de communication de Machine to Machine ou M2M.

1.3. Architecture d'un réseau LoRaWAN

Une topologie star-of-stars est utilisée avec au centre le serveur réseau. Les équipements, appelés end-devices, communiquent en modulation LoRa avec des passerelles qui centralisent les données avant de les transmettre via des liens IP (ethernet, 3G/4G) au serveur réseau.

Les end-devices ne communiquent pas exclusivement avec une passerelle définie mais avec l'ensemble des concentrateurs qui les couvrent. Ce mécanisme s'avère très efficace, particulièrement dans le cas d'un réseau mobile, en supprimant la nécessité de réaliser un handover. Inversement lorsque le serveur souhaite transmettre un message à un end-device, celui-ci est envoyé via une seule passerelle [2].

Les transmissions entre end-devices et passerelles doivent respecter les règles suivantes :

- Le device change de canal de manière pseudo-aléatoire à chaque transmission.
- Ne pas dépasser le rapport cyclique autorisé (1% en Europe, soit 36 secondes par heure).
  Ne pas dépasser la puissance maximale d'émission autorisée (+14dBm soit 25mW Europe).

Le serveur réseau assure la gestion de la sécurité, du débit adaptatif, de la redondance et communique avec des serveurs applicatifs qui exploitent les données en provenance des enddevices [2].

La figure suivante présente l'architecture d'un réseau LoRaWAN :

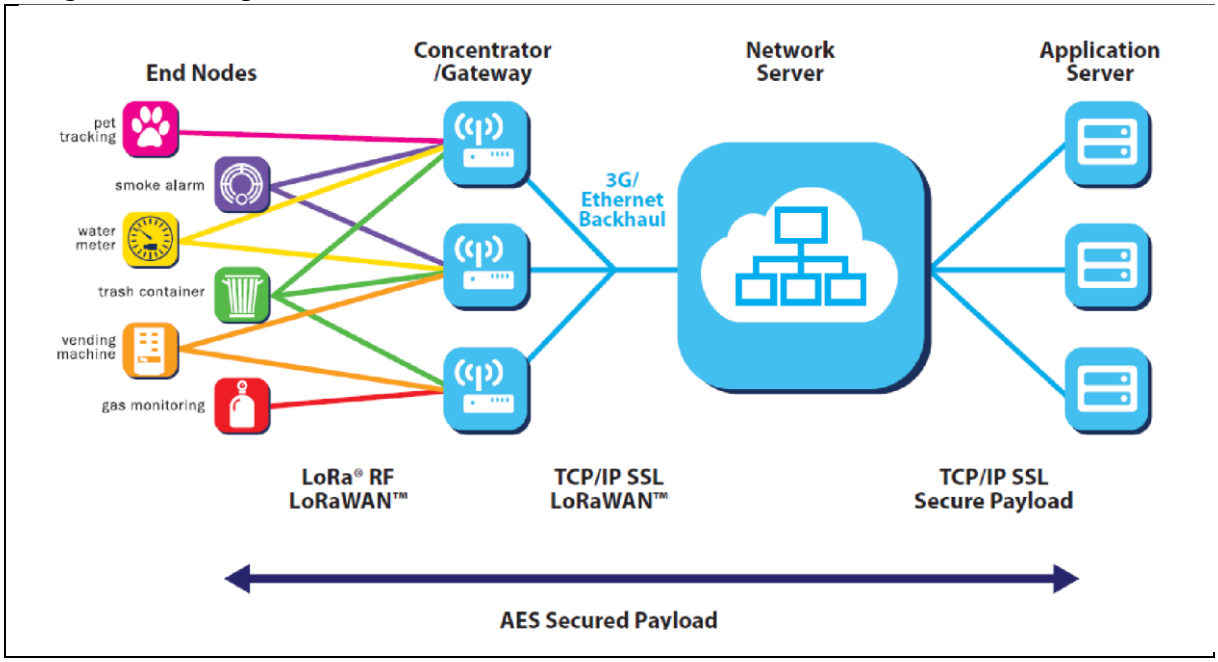


Figure 1 – Architecture d'un réseau LoRaWAN.

1.4. Un peu de théorie

Cette seconde partie explore les principaux concepts de la technologie LoRa, du protocole LoRaWAN et fournit un panorama global de l'écosystème existant.

1.4.1. LoRaWan

A. LoRa Alliance

Créée en 2015, la LoRa Alliance est un consortium d'industriels et d'opérateurs, regroupés autour de LoRa dont l'objectif est la standardisation des réseaux à longue distance LPWAN et notamment de permettre l'interopérabilité de ces réseaux de télécommunication [2].

L'alliance est à l'origine du protocole LoRaWAN qui permet de construire des réseaux utilisant comme couche physique la modulation LoRa. La dernière version en date du protocole est la version 1.02 [2].

La figure suivante présente l'architecture en couche de LoRaWAN en reprenant la terminologie du modèle OSI [2] :

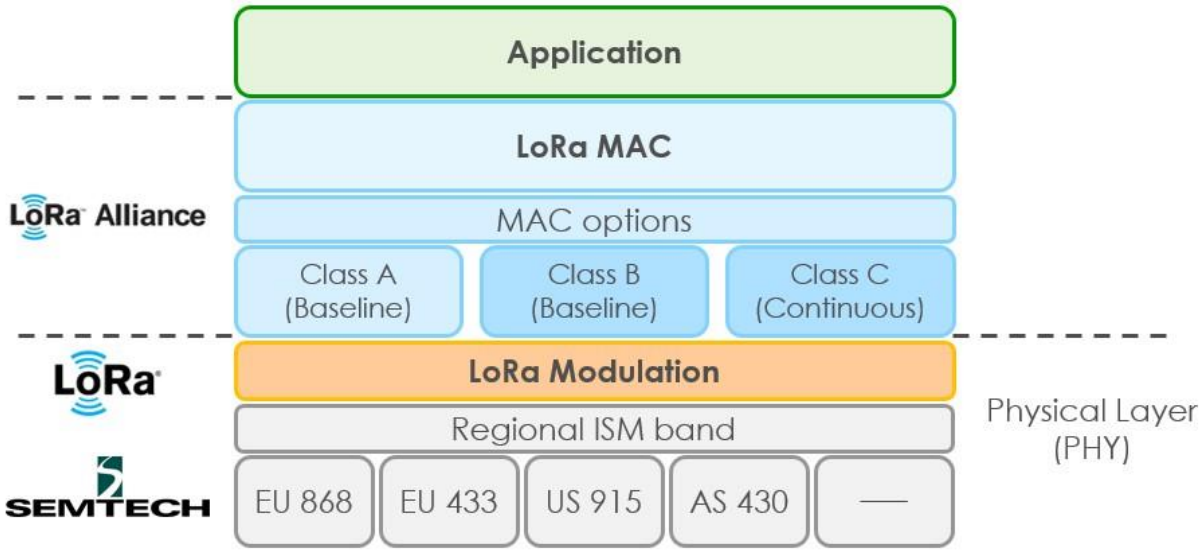


Figure 2 – Architecture en couche d'un réseau LoRaWAN.

A noter qu'il est possible d'effectuer des communications point à point entre deux end-devices, pour cela la spécification LoRaWAN définit le protocole LoRa-MAC. **B. Classes**

La spécification LoRaWAN définit trois classes d'équipements pour des applications aux besoins distincts [2] :

a. Classe A « All »

Les objets de classe A impliquent un envoi de données de l'objet vers le réseau, à l'initiative de l'objet. Lors de l'envoi de données, les objets **restent à l'écoute après l'émission pendant un intervalle régulier** (ex : toutes les 8 heures).

Exemple : Une sonde de température programmée pour envoyer une donnée toutes les 8h00, pourra aussi émettre une donnée si un écart de température important survient (> 2 degrés). La sonde restera joignable entre les 2 intervalles (toutes les 8 heures).

- **Utilisation principale** : l'objet est autonome et programmé pour envoyer des informations.

b. Classe B « Beacon »

Les objets connectés de classe B écoutent selon un planning défini avec le serveur. Régulièrement le périphérique sort de veille, écoute, puis se rendort. Les communications en temps réel avec le périphérique ne sont pas possibles.

Exemple : Prise en compte à heure fixe d'un ordre de démarrage d'une pompe à eau, pour maintenir l'oxygénation d'un bassin …

- **Utilisation principale** : l'objet est autonome et programmé pour envoyer des informations mais peut être programmé pour écouter.

c. Classe C « Continuous »

Un périphérique de classe C écoute en permanence. Il est donc généralement alimenté par une source de courant permanente. Il peut réagir immédiatement à une demande venant du réseau.

Exemple : Dans une smart city, la gestion de l'éclairage (smart city) pourrait être contrôlée par les usagers à l'aide de leurs **smartphones** et de la **géolocalisation**.

 Utilisation principale : communication quasi temps réel pour un pilotage en continu de l'objet.

C. Adresses

Le protocole LoRaWAN utilise plusieurs adresses pour identifier les différents périphériques sur le réseau.

- **DevEUI** : Identifie le end-device, format EUI-64 (unique).
- **AppEUI** : Identifie l'application, EUI-64 (unique).
- **GatewayEUI** : Identifie la passerelle, EUI-64 (unique).
- **DevAddr** : Adresse du device sur le réseau sur 32 bits (non unique).

1.4.2. La modulation LoRa, couche physique

Il existe deux grands types de modulation en LPWAN :

- **L'Ultra Narrow Band** qui consiste à transmettre les signaux dans une gamme de fréquence la plus étroite possible.
- **L'étalement de spectre** qui revient à étaler le spectre sur une large bande de fréquences avec une puissance d'émission très faible.

La modulation LoRa utilise une variante de l'étalement de spectre appelée Chirp Spread Spectrum pour coder l'information. A titre de comparaison SigFox utilise l'Ultra Narrow Band.

Un chirp ou gazouillis est un signal sinusoïdal dont la fréquence augmente ou diminue au cours du temps, on parle respectivement d'upchirp et downchirp. Cette modulation est initialement destinée à des applications militaires et est également utilisée par les radars et les sonars. Elle est robuste contre les perturbations narrowband, broadband et le multipath fading [2].

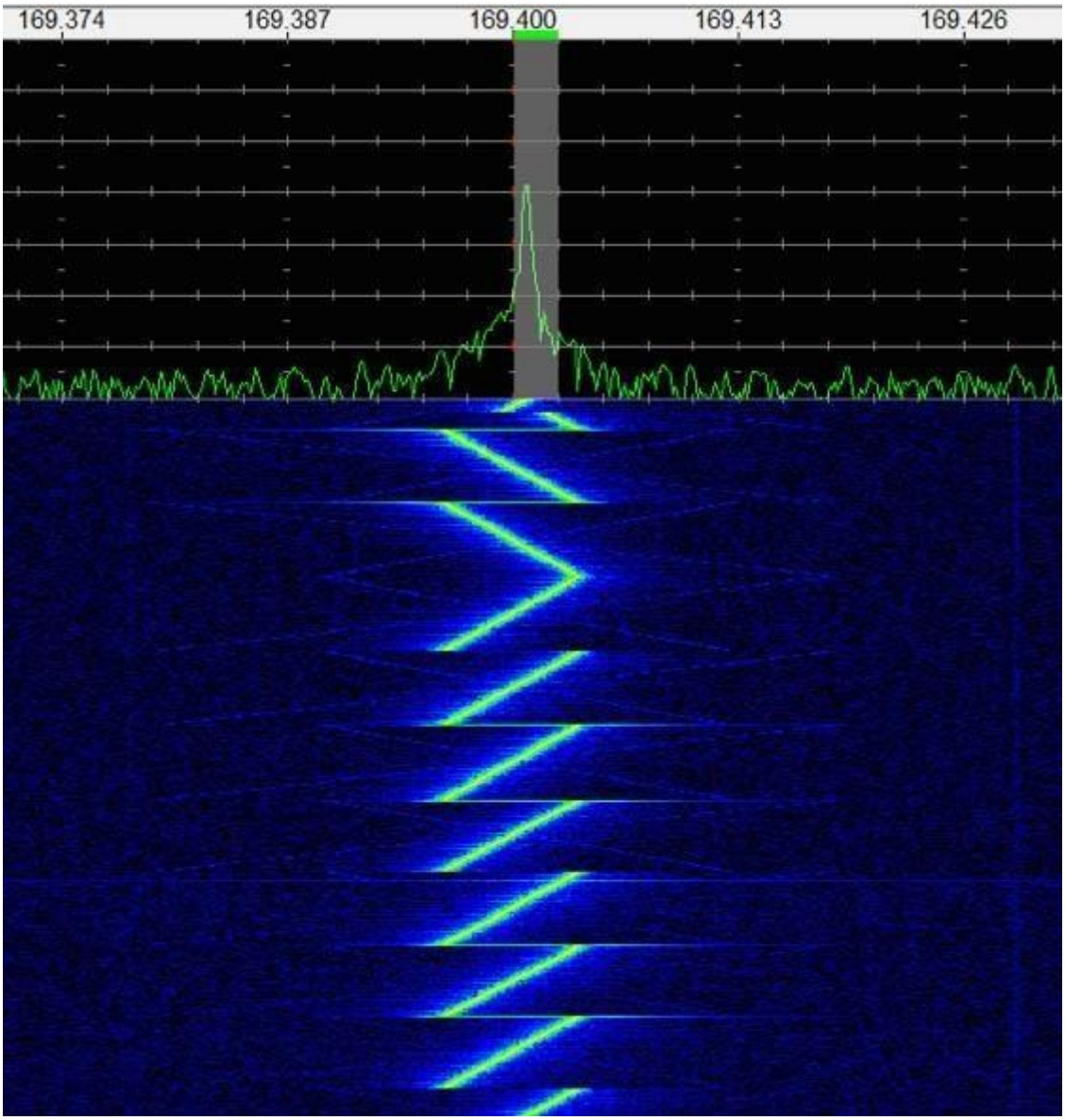


Figure 3 –Modulation LoRa.

L'étalement de spectre est réalisé par la génération d'un signal chirp linéaire dont la fréquence varie continuellement et qui vient ensuite moduler le signal à transmettre. Le facteur d'étalement, ou Spreading Factor est défini par le logarithme base 2 du nombre de chirps par symbole. Autrement dit, en LoRa un symbole est encodé par 2^{SF} chirps.

La documentation Semtech nous fournit la relation suivante entre le débit de modulation Rb, le facteur d'étalement SF, la bande passante BW et le Rate Code, un code de correction de type FEC.

$$R_b = SF \times \frac{Rate\ Code}{2^{SF}} \qquad \text{Bits / sec } [BW]$$

La portée d'une communication LoRa est déterminée par sa bande passante, la puissance d'émission du signal et le facteur d'étalement. En effet, l'étalement du signal augmente sa

portée, mais réduit le débit, la transmission étant plus longue. Une transmission plus longue entraîne une consommation d'énergie supérieure et donc une diminution d'autonomie du device.

LoRaWAN supporte 6 facteurs d'étalements (de SF7 à SF12). La figure suivante présente le débit utile et la portée en fonction du facteur d'étalement.
L'algorithme ADR de gestion du débit adaptatif se base sur des changements de SF [2].

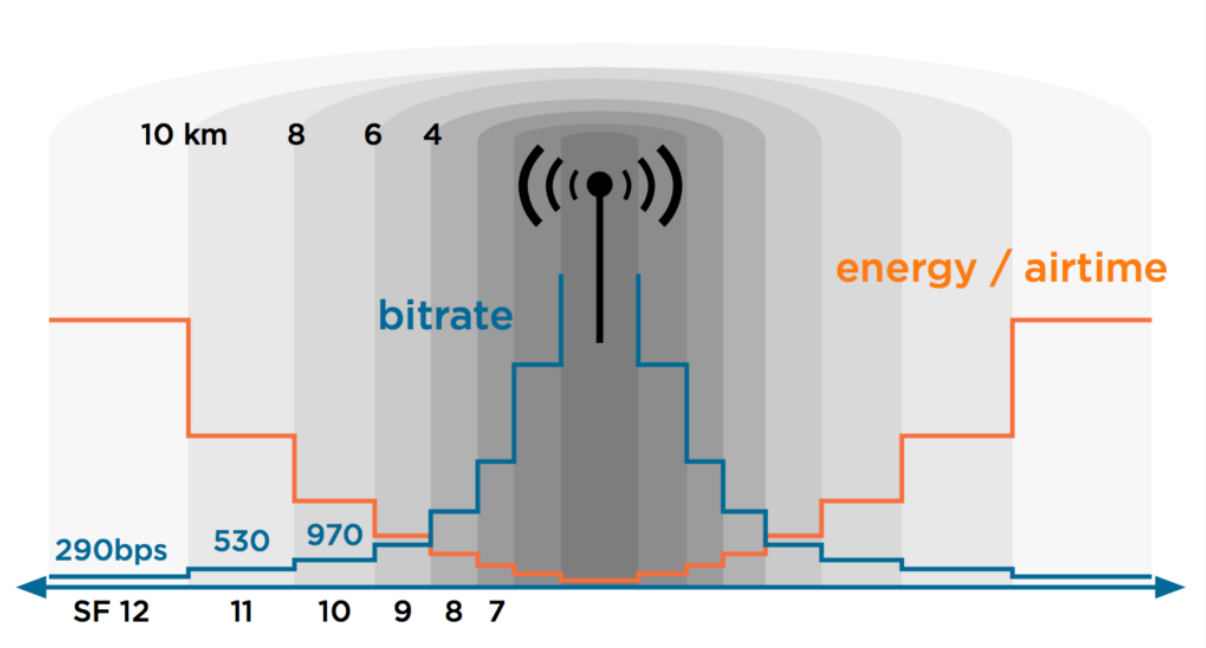


Figure 4 – Débit utile, la portée en fonction du facteur d'étalement.

Une passerelle LoRaWAN doit au minimum proposer trois canaux de fréquence, 868.10, 868.30 et 868.50 avec une bande passante de 125KHz.

Les paramètres régionaux pour l'Europe spécifient :

- 10 canaux de 125kHz/250kHz(SF7).
- Une puissance d'émission TXmax de +14dBm.
- Un débit compris entre 250bps et 11kbps (50 kbps en LoRaWAN avec modulation FSK).
- Un bilan de liaison maximal de 155dB (Semtech garantissant une sensiblité des tranceivers jusqu'à -141dBm).

1.4.3. Mécanisme d'adaptation du débit ADR

Le réseau LoRaWAN permet aux end-devices de changer individuellement leur débit de transmission. Ce mécanisme appelé Adaptive Data Rate optimise les communications des enddevices statiques, en utilisant le débit le plus rapide possible. Le temps dans les airs est réduit, la consommation d'énergie est diminuée et la capacité du réseau augmente [2].

Le schéma suivant décrit avec un exemple le fonctionnement haut niveau de l'algorithme.

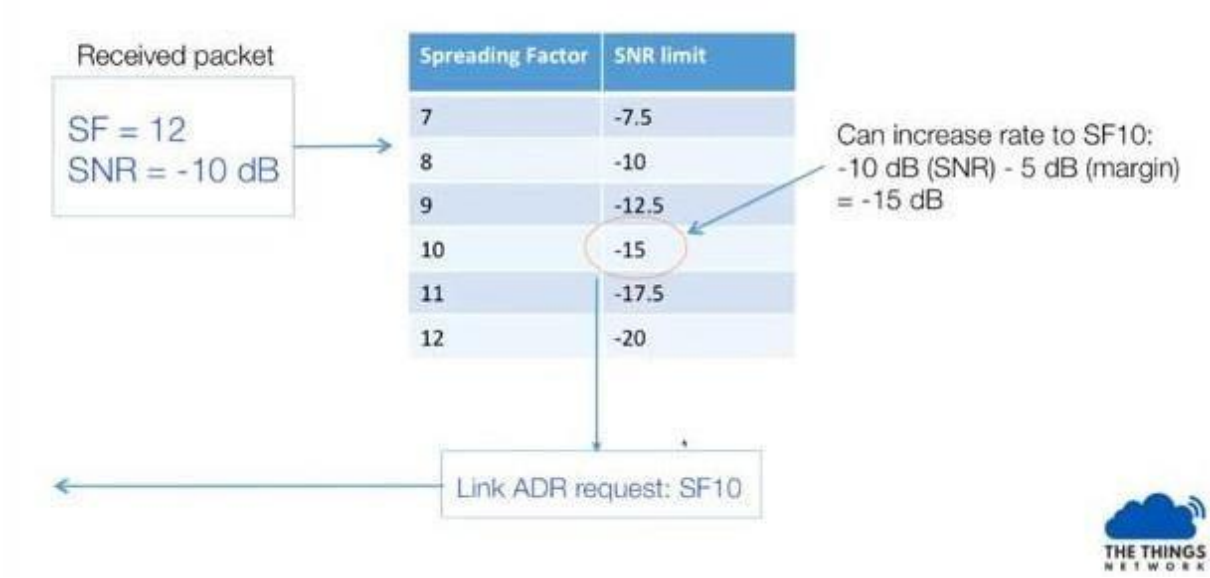


Figure 4 – Exemple de fonctionnement du mécanisme d'adaptation du débit ADR.

- Un paquet est envoyé par un end-device au débit le plus faible, correspondant au facteur d'étalement SF12 et avec un rapport signal sur bruit SNR de -10dB.
- Une marge de -5dB est ajoutée au SNR, on obtient -15dB, ce qui correspond à un SF10.
- Une instruction de changement de facteur d'étalement est envoyée au end-device, qui passe en SF10.

Plus en détail, dans l'implémentation de « The Things Network », le mécanisme est activé quand le end-device positionne le bit ADR à 1 dans ses messages uplink.

Le serveur réseau détecte ce changement et récupère le SNR maximal pour chacun des messages reçus, le nombre de passerelles et les 20 derniers messages uplink qui sont insérés dans un tableau. Si le bit ADR passe à 0, le tableau est supprimé.

Après 20 messages il est possible de calculer la marge de SNR et le nombre d'étapes d'ajustement NStep :

```
SNRmargin = SNRm – RequiredSNR(DR) - margin_db


SNRm : max SNR sur les 20 frames.

 RequiredSNR(DR) : avec le débit DR du dernier message.


NStep = round(SNRmargin/3)
```

- Si NStep > 0 : le débit peut être augmenté ou la puissance d'émission réduite. Le débit augmente par NStep jusqu'à atteindre SF7. Si le nombre d'étapes nécessaires est inférieur à NStep, le « reste » est utilisé pour diminuer la puissance d'émission, de

3dBm, étape par étape jusqu'à atteindre TXMIN = 2dBm pour la région Europe 868MHz.

- Si NStep < 0 : la puissance peut être augmentée de 3dBm par étape jusqu'à atteindre la puissance d'émission TXMAX = 14dBm.

Le débit n'est pas réduit par cet algorithme. Une réduction automatique est appliquée directement par le end-device si les messages uplinks de demande d'acquittement ne sont pas acquittés (contenant le bit ADRACKReq positionné à 1).

2. Les simulateurs

Le simulateur le plus populaire pour LoRaWAN est LoRaSim [4], qui est un simulateur d'événements discrets basé sur Python avec Bibliothèque de simulation SimPy. Il y a d'autres simulateurs qui sont Basés sur LoRaSim, ces derniers étendent LoRaSim pour plusieurs applications IoT en ajoutant le modèle de génération de données d'application avec taille du paquet de données ou pour les spécifications de LoRaWAN dans les États-Unis. Cependant, ces simulateurs sont limités en raison de certains paramètres radio fixes, le réglage (comme les interférences SF) n'est pas pris en compte, surtout, qu'aucune solution d'autogestion n'est implémentée [3].

Dans ce qui suit, nous allons détailler d'autres simulateurs qui sont développés et qui rapportent des améliorations par rapport aux autres :

2.1. LoRa-MAB : A Flexible Simulator for Decentralized Learning Resource Allocation in IoT Networks

2.1.1. Contributions du papier

Les contributions de l'article sont les suivantes [3] :

- **LoRaWAN Simulator** : ils développent un framework de simulation open source flexible, appelé LoRa-MAB, en Python avec la bibliothèque Simpy. Ce cadre capture le comportement de liaison LoRa spécifique pour plusieurs paramètres réseau avec l'impact de l'effet de capture et de la collision inter-SF.
- **Algorithme d'apprentissage distribué** : nous déployons un système distribué algorithme d'apprentissage pour la sélection des ressources radio en LoRaWAN. L'objectif est de piloter de manière autonome la décision de chaque appareil terminal vers les ressources radio la plus adéquate assurant la réactivité aux changements possibles qui peuvent se produire dans l'utilisation commune des ressources.
- **Évaluation LoRaWAN** : nous effectuons une évaluation approfondie des performances du réseau en utilisant le simulateur proposé. Nous montrons que l'approche d'apprentissage distribué est beaucoup plus efficace pour minimiser le nombre de collisions par rapport à une distribution uniforme ou triviale distribution aléatoire sur l'ensemble des ressources.

2.1.2. Simulateur LoRaWAN (LoRa-MAB)

Des travaux récents sur l'apprentissage distribué pour l'allocation des ressources radio dans LoRaWAN ont eu recours au problème du bandit multi-armé (MAB). Chaque terminal est considéré comme un agent intelligent qui choisit un SF et / ou un canal donné pour améliorer le taux de transmission de succès ou le compromis de fiabilité et d'efficacité énergétique, grâce à un processus de récompense adéquat. Dans [3], les auteurs ont supposé que tous les appareils terminaux utilisent le même SF et ont adopté l'algorithme stochastique MAB pour déterminer la sélection de fréquence. Cependant, une telle hypothèse est peu pratique en réalité en raison du couplage mutuel entre plusieurs terminaux intelligents. Un autre travail a présenté des algorithmes d'allocation des ressources stochastiques et contradictoires basés sur l'apprentissage distribué dans un réseau IoT.

Cependant, l'effet de capture et les interférences SF sont ignorés. En outre, les simulateurs engagés, qui sont basés sur Matlab, ne sont pas vraiment un simulateur de réseau et ne sont peut-être pas capables de simuler complètement les opérations réelles de LoRaWAN. Dans ce qui suit nous allons détailler le simulateur développé (LoRa-MAB)

A. Le cadre de travail de la simulation

LoRaWAN est l'architecture de réseau conçue pour connexions sans fil fonctionnant sur la technologie LoRa. Il précise différents types d'appareils, de ressources, de protocoles de transmission, et les méthodes de cryptage pour construire un réseau sans fil sécurisé à longue portée et avec une faible consommation d'énergie. En fonction de la région géographique, les communications avec LoRaWAN en occupant l'un des sous-canaux de la bande ISM et avec une bande passante de 125 kHz, 250 kHz ou 500 kHz. À cette fin, la modulation Chirp Spread Spectrum (CSS) a été utilisée, permettant d'émettre des signaux avec différents facteurs d'étalement distingués et reçus simultanément, même s'ils sont transmis en même temps et sur le même canal. Outre sélectionner un facteur d'étalement et un sous-canal pour la transmission de paquets, chaque périphérique final sélectionne un niveau de puissance de transmission jusqu'à 20 dBm. En outre, des restrictions de cycle de service de 1% ou 0:1% sont imposées pour empêcher tout appareil terminal d'envoyer souvent des données et ce pour permettre aux autres appareils finaux d'envoyer aussi des données [3].

Modèle de simulation

Nous considérons un réseau de type LoRaWAN composé d'un ou plusieurs passerelles, chacune située au centre d'un disque réseau de rayon R, avec N dispositifs terminaux uniformément distribués. Pour simplifier, nous supposons les fonctions de NS sont intégrées dans la passerelle. Ainsi, il y a 3 composants : terminaux, modèle de propagation et passerelles. Cela dépend de la méthode de sélection des ressources radio, nous séparons l'ensemble des terminaux en 2 types :

- Le terminal standard qui sélectionne les ressources en suivant un uniforme ou une triviale distribution
- Le terminal intelligent qui sélectionne les ressources en recourant à un algorithme d'apprentissage.

Le modèle de LoRaWAN-MAB pour simuler le fonctionnement de LoRaWAN est ensuite présenté sur la figure 5 avec les paramètres d'entrée et la structure du simulateur.

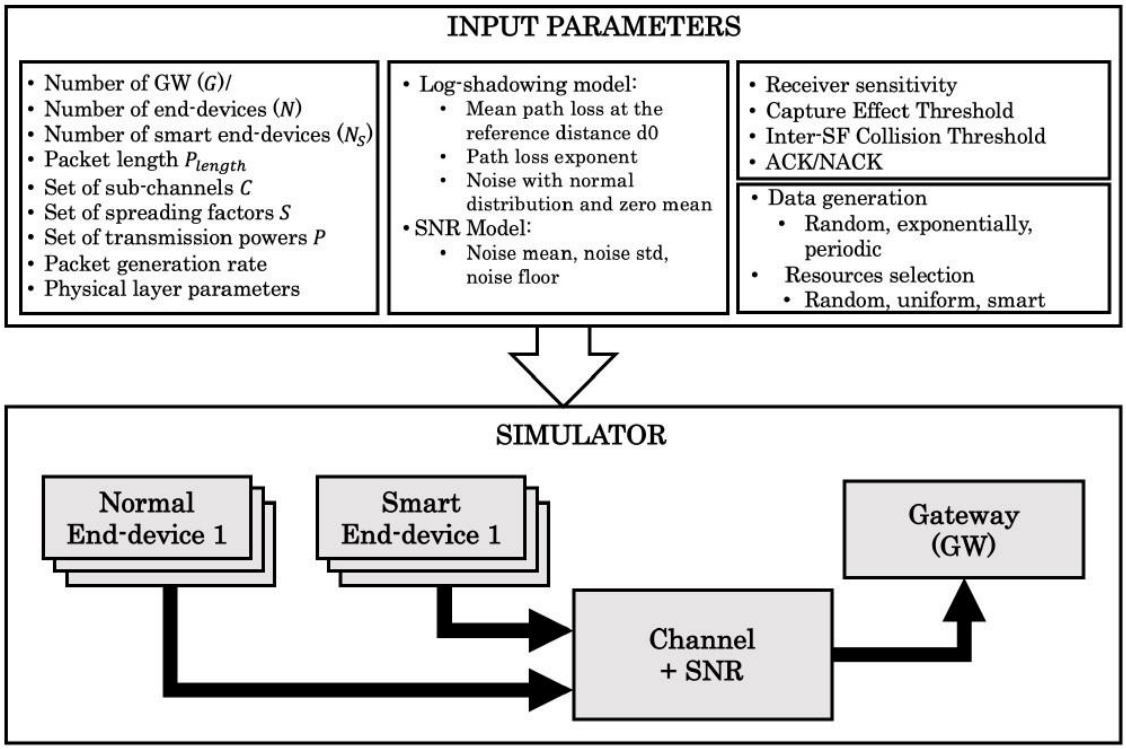


Figure 5 - Modèle de simulation du réseau LoRaWAN.

Paramètres du système a. Les ressources radio

Facteurs d'étalement (SF) : Il peut être sélectionné de 7 à 12. Plus le SF est élevé, plus le rapport signal / bruit, la sensibilité et la portée, mais aussi le temps d'antenne (temps d'occupation du canal) sont élevés [3].

Canaux : dépendent de la région du monde, la communication LoRa peut fonctionner dans la bande fréquence ISM 433 MHz et 868 MHz ou 900 MHz [3].

Niveau de puissance de transmission : en communication LoRa, chaque appareil terminal peut ajuster sa forme de puissance de transmission 4dBm à 20dBm, par pas de 1dBm [3].

Coding rate (CR) : le code de correction d'erreur directe est utilisé pour protéger la communication LoRa contre les interférences. CR peut être réglé sur 4/5, 4/6, 4/7 ou 4/8. Un CR plus élevé offre plus de protection mais augmente le temps d'antenne [3].

b. La bande passante

LoRa utilise une bande passante de 125 kHz, 250 kHz ou 500 kHz pour communiquer. Une bande passante élevée donne un débit de données plus élevé (donc un temps d'antenne plus court), mais une sensibilité plus faible (en raison de l'intégration de bruit supplémentaire). Inversement, une bande passante inférieure donne une sensibilité plus élevée, mais un débit de données inférieur.

c. Taille des paquets de données

La taille du paquet transmis par chaque terminal. Nous définissons la taille du paquet par défaut à 50 octets.

d. Restriction du rapport cyclique et taux de génération de paquets

Soit T_s le temps nécessaire pour transmettre un paquet de l octets sur facteur d'étalement s. Puis, étant donné une limitation du cycle de service notée δ, le taux de génération de paquets pour chaque terminal fonctionnant sur SF s, notée λ_s, doit vérifier $\lambda_s \, . \, T_s > \delta$ [3].

e. Modèle de propagation

Nous implémentons un modèle de propagation radio basé sur le modèle bien connu d'affaiblissement sur le trajet log-distance, qui est présenté dans [4], [15]. En utilisant ce modèle, la perte de chemin L_{pl}(d) qui dépend de la distance de communication d peut être décrite comme suit :

$$L_{pl}(d) = L_{pl}(d_0) + 10\gamma \log\left(\frac{d}{d_0}\right) + X_\sigma \ [dB]$$

Où :

$L_{pl}(d)$ est l'affaiblissement moyen sur le trajet à la distance de référence d_0 est l'exposant de perte de trajet, et $X_\sigma \sim N(0, \sigma^2)$ est la distribution normale qui explique l'ombrage, avec moyenne nulle et σ^2 variance.

f. Sensibilité et comportement de liaison

La sensibilité d'un récepteur radio à température ambiante est calculée avec la formule suivante :

$$S = -174 + 10\log(BW) + NF + SNR \ [dB]$$

Où le premier terme décrit le bruit thermique dans 1 Hz de bande passante et ne peut être influencé qu'en changeant la température du récepteur. BW est la largeur de bande du récepteur. NF est le facteur de bruit du récepteur qui est fixé pour un matériel donné. SNR est le rapport signal / bruit requis par le schéma de modulation sous-jacent et est déterminé par le facteur d'étalement SF. Plus le SF est élevé, plus le SNR est élevé.

Un paquet LoRa transmis sera reçu si sa puissance au récepteur est supérieure ou égale à la sensibilité du récepteur, qui est donnée dans le tableau ci-dessous :

SF	Bit-rate [kbps]	Receiver Sensitivity [dBm]	Reception Thresh [dB]	Inter-SF collision Thresh [dB]
7	5.47	-123	-6	-7.5
8	3.13	-126	-9	-9
9	1.76	-129	-12	-13.5
10	0.98	-132	-15	-15
11	0.54	134.5	-17.5	-18
12	0.29	-137	-20	-22.5

Tableau 1 – Caractéristiques LoRa avec une bande passante = 125 kHz.

B. Les collisions

En raison des ressources limitées, deux transmissions LoRa ou plus peuvent se chevaucher au niveau de la passerelle. Dans un tel cas, il y a certaines conditions qui déterminent si les deux

paquets peuvent être décodés avec succès, un seul ou aucun. Ces conditions dépendent sur le facteur d'étalement, le canal, la puissance et le timing [7]. Premièrement, une collision se produit lorsque deux dispositifs ou plus transmettent en même temps avec le même SF et dans le même canal. Ensuite, une collision temporelle se produira si ces transmissions se chevauchent dans leur section critique (c'est-à-dire la section qui commence aux 5 derniers symboles de préambule du paquet). De plus, s'il y a plusieurs signaux transmettant avec le même SF et sur le même sous-canal simultanément, le NS est toujours en mesure de recevoir avec succès le signal le plus fort si son SINR (Rapport Signal/Bruit en français) est supérieur à un seuil de 6 dB. Ce phénomène est appelé effet de capture [3].

Néanmoins, l'orthogonalité parfaite n'est pas garantie et les interférences entre les communications utilisant différents SF doivent également être prises en compte. En fait, le NS peut recevoir avec succès un signal utilisant SF s si sa puissance est supérieure d'un seuil donné (dans le tableau I) à la multiplication de l'interférence totale des signaux utilisant SF s0, s. Ce phénomène est considéré comme une collision inter SF [3].

C. Processus de simulation

Premièrement, chaque périphérique d'extrémité génère un paquet en suivant une distribution aléatoire, telle que la distribution exponentielle. Le taux de génération de paquets est choisi pour satisfaire la restriction du cycle de service. Le paquet généré est ensuite transmis à la passerelle en utilisant les ressources radio sélectionnées. Après l'envoi d'un paquet, le terminal attend un accusé de réception (ACK) envoyé par le NS via la passerelle. Nous supposons qu'il n'y a pas de collision entre les paquets ACK et de liaison montante. En fait, un ACK peut être livré sur un canal séparé avec un rapport cyclique plus élevé. Par conséquent, si un périphérique d'extrémité reçoit un ACK pour son paquet transmis, alors il n'y a pas eu de collision ou l'effet de capture s'est produit. Inversement, lorsqu'un ACK n'est pas reçu, soit le paquet a été perdu en raison d'une collision avec un autre paquet transmis avec le même SF, soit en raison de la collision inter SF. En fonction de l'ACK ou du NACK reçu, le terminal mettra à jour sa stratégie pour ré sélectionner les ressources radio appropriées afin de minimiser le nombre de collisions [3].

Le processus de simulation du simulateur LoRaWAN proposé est illustré dans la figure suivante :

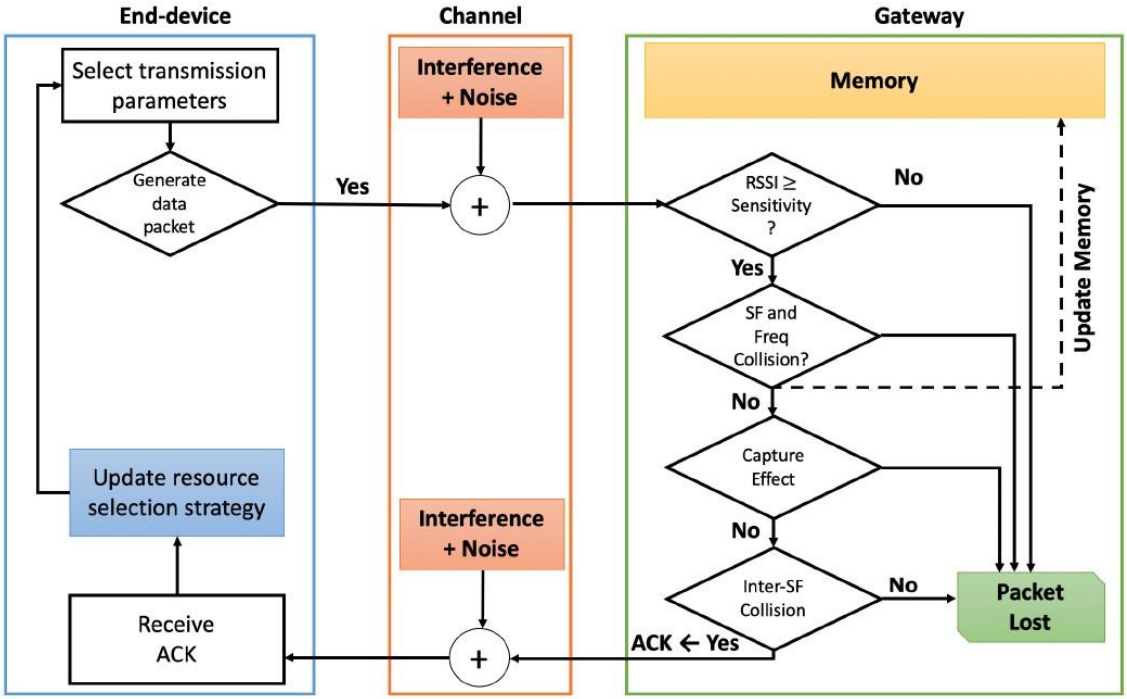


Figure 6 – Comportement de lien et processus de simulation.

D. L'algorithme EXP3

Nous avons recours à l'algorithme EXP3 populaire [4] pour résoudre le problème d'allocation des ressources distribuées. L'objectif est d'orienter la décision de chaque appareil i de choisir de manière autonome (c_i, s_i, p_i) les plus appropriés, avec le minimum de collision tout en assurant la réactivité aux changements possibles qui peuvent se produire dans l'utilisation commune des ressources.

Soit $a(t)$ l'action choisie au temps t, ce dernier donne une récompense $r_a(t)$. A chaque itération t (à l'arrivée des paquets), tout appareil i sélectionne une stratégie avec une distribution $p_a{}^i(t)$ sur A. Le but de tout appareil i est de mettre à jour $p_a{}^i(t)$ afin d'obtenir la plus grande récompense à l'horizon T par rapport à la meilleure stratégie fixe. Nous initialisons l'algorithme avec la distribution uniforme $p_s{}^i(0) = {}_A1$, où A est le cardinal de la stratégie A.

L'algorithme EXP3 pour la sélection du facteur d'étalement dans LoRaWAN est ensuite présenté dans l'algorithme 1. Notez qu'en cas de perte de paquet, $r_s(t) = 0$ et aucune mise à jour ne sera opérée sur la stratégie de distribution, et donc aucun apprentissage non plus.

Initialization:

- Let $a = A_j \in \mathcal{A}$ be the strategy chosen by device j.
- Set the initial weights $\omega_a^j(0) = 1, \forall a \in \mathcal{A}, \forall j \in \mathcal{N}$ and the uniform distribution of strategies per device.
- Set the learning rate $\gamma = \min\left\{1, \sqrt{\frac{K\log(K)}{(e-1)T}}\right\}$ where $e \approx 2.71828\ldots$ is the base of the natural logarithm.

for $t = 1$ *to* T **do**
  initialization ;
  foreach *end-device* j **do**
    At time t, draw strategy $a \in \mathcal{A}$ according to the distribution $p_a^j(t)$;
    if *Transmit* **then**
      Receive reward

$$r_a^j(t) = \begin{cases} 1 & \text{if ACK is received,} \\ 0 & \text{otherwise.} \end{cases}$$

      Update weights and distribution of available strategies:

$$\omega_a^j(t+1) = \omega_a^j(t)\exp\left(\frac{\gamma r_a^j(t)}{K \cdot p_a^j(t)}\right)$$

$$p_a^j(t+1) = (1-\gamma)\frac{\omega_a^j(t+1)}{\sum_{a=1}^{K}\omega_a^j(t+1)} + \frac{\gamma}{K}$$

    end
  end
end

Algorithme 1 - Algorithme EXP3 pour l'allocation SF entièrement distribuée dans le réseau LoRa

2.2. IoT scheduling for higher throughput and lower transmission power

2.2.1. Contributions du papier

Les principales contributions de cet article sont les suivantes [5] :

- **Expression mathématique** : en forme proche de la probabilité de collision et de perte de paquets.
- **Un algorithme de programmation horaire est proposé** : en utilisant des appareils prenant en charge le mode LoRaWAN classe C pour la synchronisation. Par la suite, ces dispositifs passent en classe A pour réduire considérablement la collision et améliorer l'évolutivité en attribuant un temps de garde à chaque nœud d'extrémité.
- **Un algorithme de facteur d'étalement de la distance est proposé** : en fonction de la distance des nœuds d'extrémité de la passerelle et des valeurs de temps d'antenne, nous attribuons la meilleure valeur SF à chaque paquet. Cela entraîne une diminution de la probabilité de collision et un débit plus élevé.

2.2.2. Simulateur LoRa

Nous supposons que le réseau LoRa se compose d'une passerelle et de nombreux nœuds d'extrémité fonctionnant à la fréquence des canaux principaux et de deux canaux de liaison descendante. Tout d'abord, nous considérons une classe C pour la synchronisation entre la passerelle et les nœuds d'extrémité pour spécifier le temps de garde car il est possible que la même procédure soit utilisée pour la FOTA (Firmware Upgrade Over The Air). En cas de collision, nous passons ensuite en classe A. Cela sera pris en compte lorsque les nœuds d'extrémité transmettront les paquets en même temps.

Les nœuds d'extrémité génèrent des paquets aléatoires selon un processus de distribution de Poisson avec une intensité totale k (qui peut être définie) et déterminent la différence de temps moyenne entre deux paquets en millisecondes. Le nombre d'appareils terminaux fonctionnant dans le canal peut être configuré dans la simulation. Le simulateur génère des paquets pour chaque nœud d'extrémité, correspondant au nombre de nœuds d'extrémité. Ces derniers envoient d'abord les données à la passerelle sous forme d'envoi périodique via la classe A, puis ouvrent deux fenêtres de réception (RX1 et RX2), offrant à la passerelle la possibilité d'envoyer un message d'accusé de réception de liaison descendante.

La passerelle a le contrôle sur les nœuds d'extrémité via la concurrence utilisant la classe C, et cela conduit à une latence accrue des communications de liaison descendante car elle doit attendre la prochaine transmission planifiée de liaison montante. Cela a un impact sur plusieurs applications où à la fois une faible latence et une faible consommation d'énergie jouent des problèmes essentiels tels que la santé structurelle ou la surveillance de l'activité sismique. Si aucune collision ne se produit pour la liaison montante, les nœuds d'extrémité sont acquittés par la passerelle. Après chaque transmission de liaison montante, les nœuds d'extrémité attendent un ACK d'une passerelle. Afin d'économiser de l'énergie en diminuant la probabilité de collision et la perte de paquets. Deuxièmement, en proposant un algorithme de facteur d'étalement de distance. Lorsque les nœuds d'extrémité envoient des paquets en utilisant le même facteur d'étalement, cela entraînera une perte de paquets et une forte probabilité de collision. À cet effet, notre algorithme compare les valeurs de distance pour chaque paquet en collision de nœuds finaux de la passerelle, pour donner à chaque nœud final la valeur SF appropriée en fonction de la distance entre les nœuds finaux et la passerelle [5].

2.2.3. Résultats de la simulation

Dans cette section, pour valider notre approche, nous considérons le système composé d'une passerelle qui peut établir des connexions avec N nœuds d'extrémité en utilisant le même facteur d'étalement. Notre implémentation pour les algorithmes d'ordonnancement proposés et la transmission à travers le message de liaison descendante de passerelle pour les nœuds d'extrémité est présentée en deux approches. Les paramètres LoRa utilisés : fc = 868 MHz, BW = 125 kHz et CR = 1.

Tout d'abord, l'algorithme de programmation temporelle est proposé, le temps est divisé en tranches de temps pouvant accueillir la communication dans le réseau. La passerelle est chargée d'ajouter le temps de garde à chaque paquet avec le message d'accusé de réception via la synchronisation de la classe C avec les nœuds d'extrémité. Passez ensuite en classe A s'il n'y a pas eu de collision et pour la périodicité. L'architecture d'implémentation de ce scénario est

représentée sur la figure 6. Sur la base de la tranche de temps dans le message ACK, la passerelle détermine le temps de garde après la collision de paquets survenue à partir de différents nœuds d'extrémité.

À partir de ce moment, après avoir ajouté le temps de garde à chaque paquet, les nœuds d'extrémité commencent à augmenter l'intervalle de temps pour chaque paquet avant de les transmettre et cela est organisé par la passerelle.

Deuxièmement, l'algorithme du facteur d'étalement de la distance est proposé. Chaque nœud d'extrémité sélectionne au hasard la valeur SF du facteur d'étalement autorisé de 7 à 12. Une collision se produit lorsque deux nœuds d'extrémité envoient des paquets en utilisant le même facteur d'étalement en même temps.

Notre algorithme proposé pour les nœuds situés près de la passerelle leur attribue la valeur SF la plus faible. En comparant les distances pour les nœuds d'extrémité, la distance à laquelle il se trouve par rapport à la passerelle, nous attribuons la valeur SF appropriée pour chaque nœud d'extrémité.

La détermination de la distance en fonction de la valeur du temps d'antenne reçu.

Les détails de l'algorithme proposé pour la collision de planification de facteur d'étalement sont donnés dans l'algorithme 2, figure 7.

Algorithm 1 D-SF-Scheduling collision

Input D: Distance , SF: Spreading Factor , T: Time on Air
Begin : Generate random packets from different nodes
For each kk $\in K$ *and* $ff \in F$ **Do**
Start from the first packet from the first node
Search D with value of SF
Compare D
Descirption :
IF $100m \leq D \leq 5000m$
Then $SF = 7$ or 8
Else If $5000m \leq D \leq 10000m$
Then $SF = 9$ or 10
Else $SF = 11$ or 12
END

Algorithme 2 – D-SF-Scheduling collision.

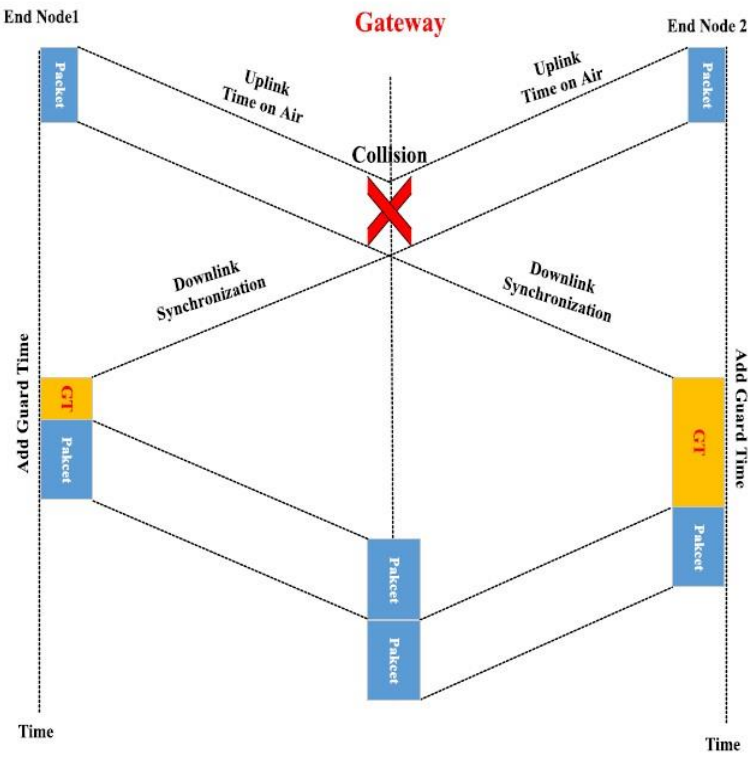


Figure 7 - L'architecture d'implémentation du scénario précédent.

Les nœuds d'extrémité sont dispersés géographiquement dans le réseau. Les simulations sont effectuées par Matlab en utilisant un nombre différent de nœuds d'extrémité (5, 10, 20, 50, 100) pour chaque scénario. Dans le premier scénario, nous avons simulé avec un nombre légèrement accru de nœuds d'extrémité. La figure 8 montre l'évaluation de l'algorithme de programmation horaire. Ainsi, le nombre supposé de paquets entrés en collision est de 2 lorsque nous l'avons appliqué à cinq nœuds d'extrémité. Après avoir utilisé l'algorithme de programmation temporelle, les résultats de la simulation montrent des améliorations en termes de diminution de la probabilité de collision de 18%. Pour pouvoir évaluer la solution proposée dans diverses conditions, la figure 9 présente les paramètres de collision lorsque le nombre de nœuds d'extrémité passe de 5 à 10. Les résultats montrent que l'augmentation est utilisée dans la simulation. Cela montre également que l'amélioration de minimiser la probabilité de collision c'est en augmentant le nombre de nœuds d'extrémité dans le réseau [5].

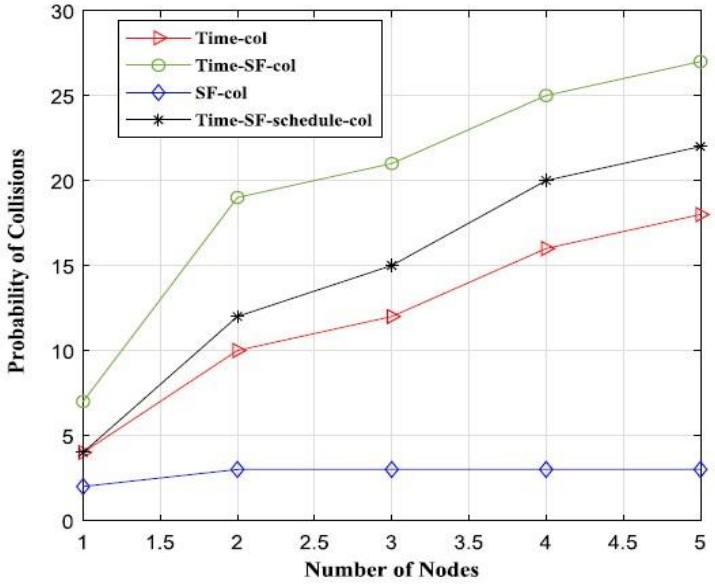


Figure 8 - Probabilité d'au moins une collision pour 5 EN

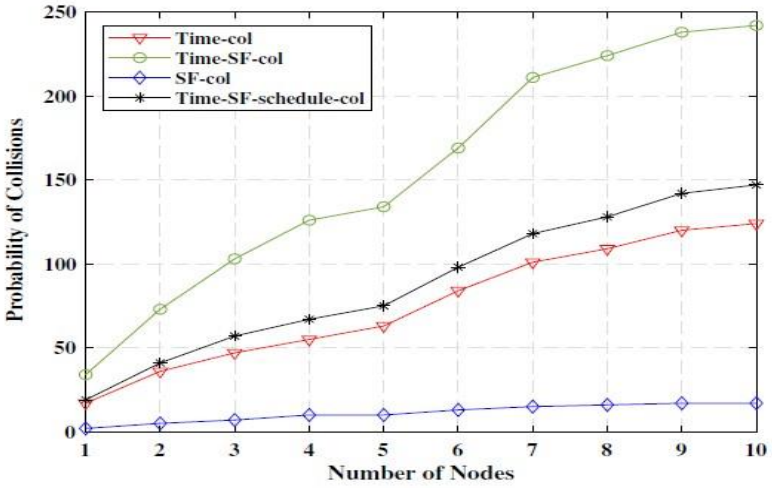


Figure 9 - Probabilité d'au moins une collision pour 5 EN

2.3. Experimental Evaluation of LoRaWAN in NS-3

2.3.1. Contributions du papier

Les contributions de l'article sont les suivantes [6] :

- **La méthodologie expérimentale adoptée** : Cela comprend une description du module LoRaWAN basé sur NS-3, différentes mesures de performance et l'environnement de simulation de leurs expériences.
- **Des détails sur chaque cas d'utilisation et fournit les résultats correspondants de leurs expériences :** utilisant le module LoRaWAN basé sur NS-3.

- **Les résultats de la simulation sont discutés** : montrant le PDR, le débit UL, l'utilisation de la sous-bande ainsi que l'impact de la charge du réseau et des ressources de la sous-bande sur les performances résultantes. Enfin, la section VI conclut cette étude.

2.3.2. MÉTHODOLOGIE EXPÉRIMENTALE

Un réseau LoRaWAN de taille raisonnable nécessite le déploiement de plusieurs dispositifs de détection de faible puissance pouvant atteindre plusieurs milliers de nœuds. Construire et entretenir un réseau aussi vaste n'est pas anodin, coûteux et prend du temps. Par conséquent, une méthode courante consiste à utiliser la simulation pour fournir une approximation d'un environnement réseau réel.

A. Module LoRaWAN dans NS-3

Le module NS-3 LoRaWAN est une extension du module NS-3 pour le réseau étendu à faible puissance (LPWAN) [6]. Les composants LoRaWAN MAC / PHY s'exécutent sur chaque périphérique LoRa et la passerelle. La couche PHY de chaque appareil interagit avec celle de la passerelle respective via le module PH-NS-3 Spectrum, mettant en œuvre l'interface aérienne des appareils et les paramètres spécifiques au canal, comme indiqué sur la figure suivante :

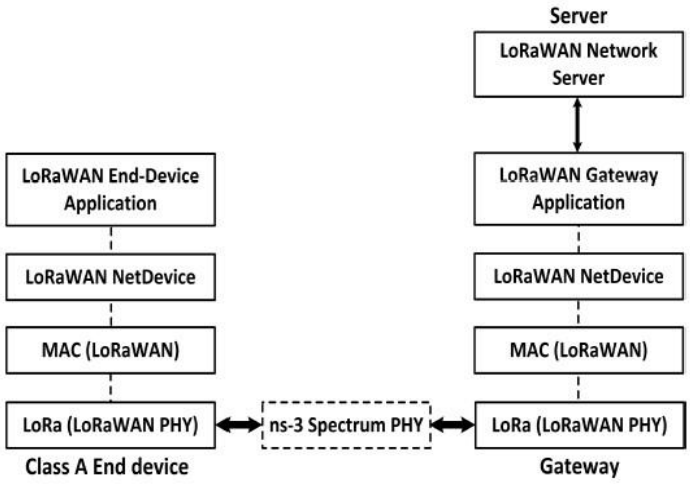


Figure 10 - Configuration du module LoRaWAN dans NS-3 : périphériques LoRaWAN de classe A, passerelle et NS

Comme mentionné précédemment, sur chaque canal dans une sous-bande, une passerelle peut recevoir simultanément des signaux avec différents SF. Notez que dans le banc d'essai, le LoRa PHY utilise le modèle d'erreur de l'implémentation en bande de base de la couche PHY dans MATLAB, les simulations considérant un canal AWGN [6]. Le modèle de collision utilisé dans le module NS-3 est basé sur l'effet de capture. L'effet de capture entre en jeu lorsque, lors de la collision de deux transmissions UL simultanées (avec la même fréquence et SF), le signal le plus fort capture le signal le plus faible. En conséquence, la trame avec une puissance reçue plus élevée est reçue avec succès par la passerelle tandis que la trame avec une puissance de réception plus faible est perdue. Dans le module LoRaWAN, dans le cas de transmissions UL confirmées, le choix des créneaux de réception (RS1 / RS2) par la passerelle pour renvoyer les ACK à l'appareil est basé sur les éléments suivants :

- Restriction RDC sur la sous-bande.
- Sur le canal donné et SF, le MAC LoRaWAN est en état de repos.
- Aucune autre donnée MAC n'est planifiée sur le canal particulier avec la même bande passante et SF.

Notons que l'application génère des paquets suivant la loi de Poisson avec un débit moyen, considéré comme le débit d'arrivée du trafic. Nous supposons que ce taux de génération de paquets est fixe pour tous les périphériques réseau.

B. Les Paramètres de simulation

Les paramètres utilisés dans nos expériences sont répertoriés dans le tableau 2 suivant :

Paramètres de simulatior	Valeur
UL device tx. Power	14dBm
Rayon de couverture de la passerelle	Scenario 1 (7km), 2 (5km)
Spreading factor (SF)	SF12
Longueur du préambule	8 bytes
Frame PHY Payload	21 bytes
ACK Payload	12 bytes
Code rate (CR)	$\frac{4}{7}$
Δ	1% (for UL)
Bande passante du canal	125 kHz
Modèle de perte de chemin	LogDistancePropagationLoss

Tableau 2 – Paramètres de la simulation.

De plus, nos résultats sont basés sur les hypothèses suivantes :

- Seul le trafic UL des appareils de classe A est pris en compte, qui peut être constitué de trames non confirmées ou confirmées.
- Toutes les transmissions UL à partir d'un appareil suivent les spécifications de EU863870 [6] par exemple : Les transmissions UL suivent 1% RDC (comme défini dans le tableau 2), tandis que les transmissions DL adhèrent à un RDC de 10%.
- Chaque appareil $a \in A$ envoie des trames de taille égale sur l'UL avec une intensité de trafic tI.

C. SCENARIOS DE SIMULATION ET RESULTATS

Cette section définit l'environnement de simulation et les scénarios de cas d'utilisation pour l'évaluation de différentes mesures de performances :

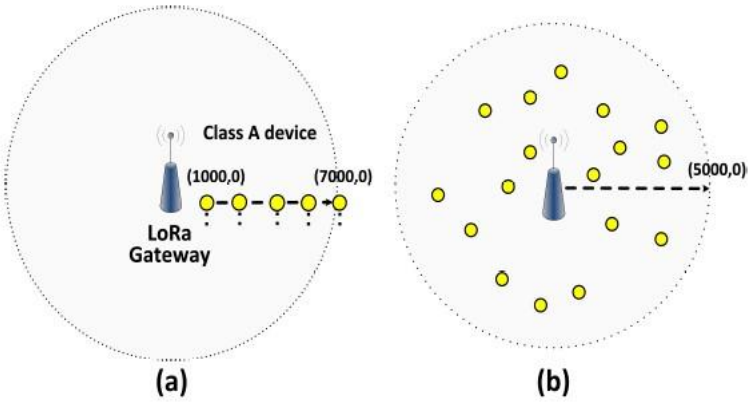


Figure 11 - (a) Scénario-I : Mobilité d'un seul appareil, (b) Scénario-II : Plusieurs appareils répartis uniformément sur un rayon de 5 km.

Scenario I - Mobilité d'un seul appareil

Nous considérons un réseau de passerelle unique reliant les périphériques d'extrémité au NS. Le serveur peut gérer le canal et les SF d'un périphérique via la passerelle pour obtenir de meilleures performances. Initialement, le réseau dessert un seul appareil et une sous-bande (avec un seul canal) est disponible pour la transmission de données UL. Le périphérique final se voit attribuer un SF fixe par le serveur.

- Dispositif unique à passerelle unique : Dans un cas initial, un dispositif terminal se déplace le long d'une ligne droite, c'est-à-dire à une distance de 1 km de la passerelle, à 7 km de la passerelle (comme le montre la figure 4a). Après un certain temps, le nœud envoie 100 paquets sur l'UL. Ici, nous souhaitons évaluer l'impact de la mobilité, ou plus précisément de la distance de la passerelle, sur le PDR et le débit UL. Notez que nous considérons ici uniquement la transmission de trame non confirmée sur l'UL [6].

Résultats pour la mobilité d'un seul appareil

1) Ratio de livraison de paquets (PDR) : la figure 12 trace le PDR résultant pour les différents SF fixes configurés sur le canal donné lorsque l'appareil s'éloigne de la passerelle. On peut voir qu'un réglage de débit UL (liaison montante) de SF = 7 limite la plage de communication de la passerelle à environ 2 km, et après 2,2 km, le PDR de trame UL se réduit à zéro, car toutes les trames sont perdues en raison du faible rapport signal / bruit (SNR).) au niveau du récepteur. Notez qu'un SF plus élevé augmente la sensibilité du récepteur et permet une meilleure couverture et portée sur une plus grande distance. En conséquence, l'UL PDR pour SF = 12 reste proche de 100%, même si l'appareil est à 6 km de la passerelle [6].

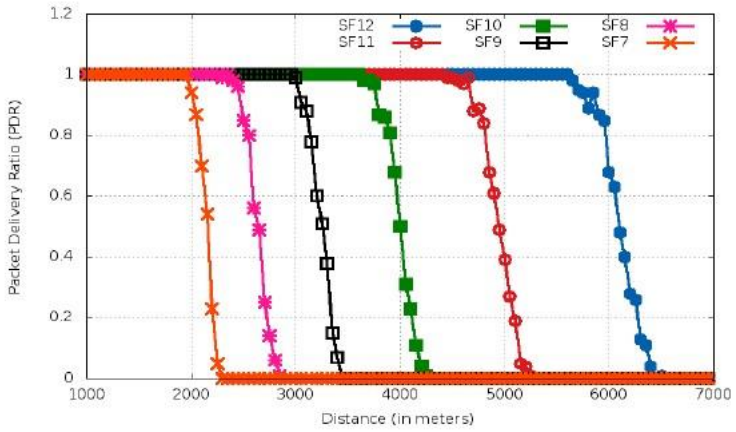


Figure 12 - UL PDR à partir d'un seul appareil mobile avec SF fixe.

2) Débit de liaison montante (UL) : Ensuite sur la figure 13, le résultat montre le débit UL (bits par seconde) pour un SF fixe. On peut voir qu'avec un faible SF (par exemple SF = 7) un débit UL plus élevé peut être atteint. Cependant, il ne durera que sur une petite distance, comme le montre la figure 13, en raison de la faible sensibilité des récepteurs. De même, avec un SF plus élevé (par exemple SF = 12), toutes les trames UL jusqu'à une distance de 6 km peuvent être reçues avec succès. Il est important de noter que sur la base des résultats ci-dessus, un schéma optimal peut être conçu qui choisit de manière adaptative les SF pour différents appareils afin de maximiser leurs taux UL. En utilisant une sélection de taux adaptative, il est prévu qu'un appareil mobile puisse atteindre un meilleur taux UL global par rapport à un schéma SF fixe [6].

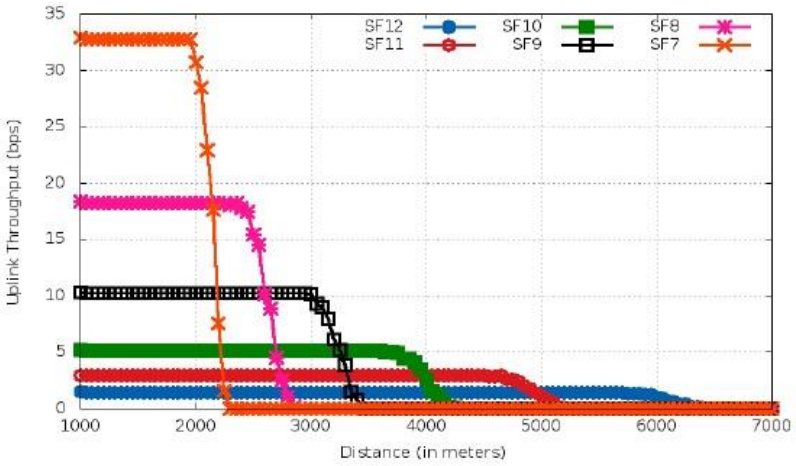


Figure 13 - Débit de liaison montante (UL) pour l'appareil mobile avec différents fixes SF.

Scénario II - Plusieurs périphériques, modèle de trafic de Poisson

Dans ce scénario, nous considérons un réseau de passerelle unique avec plusieurs périphériques répartis uniformément sur une zone circulaire avec un rayon de 5 km, comme le montre la figure 11 (b). Tous les appareils sont configurés pour envoyer des données UL avec SF = 12. Pour les collisions de trames, le modèle implémenté dans le module LoRaWAN NS-3 est basé sur l'effet de capture. De plus, lorsque plusieurs canaux sont disponibles sur la sous-bande (c'est-à-dire mc > 1), nous supposons que tous les canaux ont une probabilité égale d'être sélectionnés par le dispositif pour chaque nouvelle transmission UL.

Dans un premier temps, nous considérons un réseau de passerelle à canal unique (c'est-à-dire une sous-bande avec un seul canal disponible) avec l'arrivée de trafic suivant un processus de Poisson avec un temps moyen d'arrivée inter-réparti ^{1}, comme le montre la Figure 14 [6].

λ

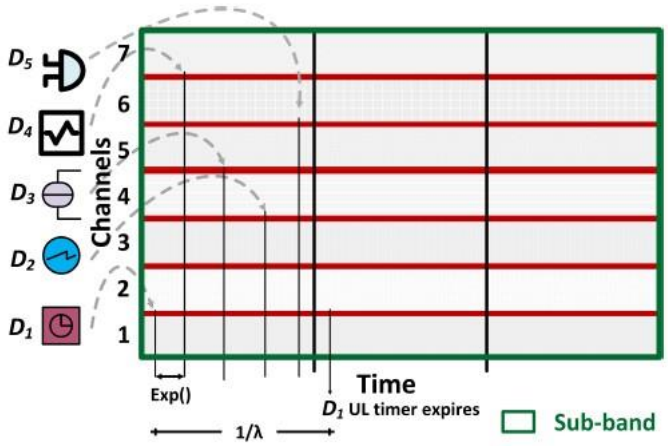


Figure 14 - Arrivées de poisson sur la sous-bande avec taux moyen λ

L'impact de la charge croissante (nombre d'appareils) sur les mesures de performances du réseau telle que l'utilisation de la sous-bande et PDR, est évaluée séparément pour les cas de transmissions UL confirmées et non confirmées.

Résultats pour plusieurs appareils

Dans ce qui suit, nous décrivons les résultats du scénario II.

1) Utilisation de la sous-bande (U) : Compte tenu des différentes intensités de trafic (t_I) ainsi que des transmissions maximales pour les trames UL confirmées et non confirmées (par exemple N_{max-C} et N_{max-UC}), cette section évalue les résultats d'utilisation de la sous-bande.

Transmissions variables (N_{max-C} / N_{max-UC}) Les résultats d'utilisation de la figure 15 sont obtenus dans un environnement de réseau entièrement chargé (par exemple $t_I = 1$). Notez qu'avec $N_{max-UC} = 1$, l'utilisation devient plus élevée sur la figure 15 car les transmissions UL sont capables d'utiliser pleinement la capacité à mesure que le nombre d'appareils augmente. Intuitivement, on peut également voir que les résultats d'utilisation pour les transmissions UL confirmées avec différents N_{max-C} ne changent pas comme prévu. En effet, la transmission ACK de la passerelle sur le même canal augmente les collisions et diminue le nombre de

trames délivrées avec succès. En conséquence, l'utilisation des ressources en cas de trames confirmées est considérablement inférieure à celle des trames non confirmées [6].

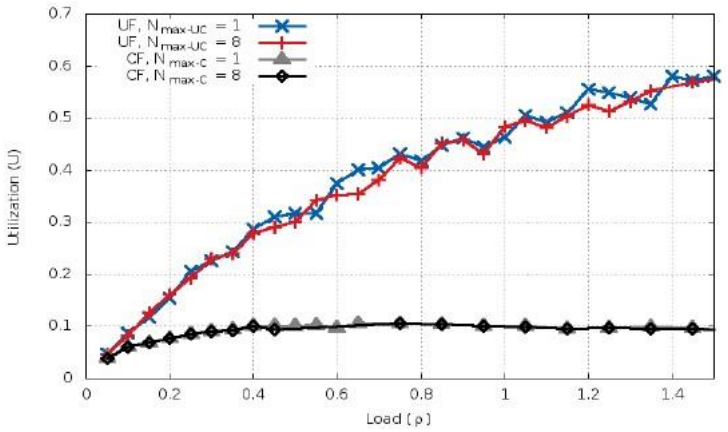


Figure 15 - Utilisation de la sous-bande par rapport à la charge du réseau avec un nombre maximal variable de transmissions de trames UL confirmées / non confirmées autorisées avec SF = 12 fixe, $t_I = 1.0$, $m_c = 1$.

Ensuite, lorsque les transmissions maximales augmentent de 1 à 8 (c'est-à-dire $N_{max-UC} = 8$), l'utilisation reste inchangée. En effet, le canal est utilisé jusqu'à sa limite maximale dans les deux cas et la probabilité de collision ne change donc pas. Du point de vue de l'application, l'utilisation pour le cas non confirmé sera faible lorsque le nombre maximal de transmissions est élevé, même si plusieurs copies de la même trame sont reçues avec succès. Cependant, un seul d'entre eux sera transmis à l'application et compte comme un paquet reçu au niveau de la couche application [6].

Variation de l'intensité du trafic (t_I) Les résultats de la figure 16 montrent l'impact de l'intensité du trafic (t_I) sur l'utilisation de la sous-bande pour les transmissions UL confirmées et non confirmées.

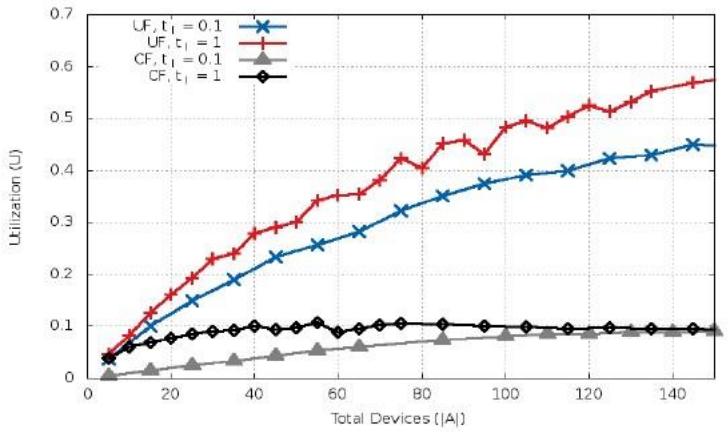
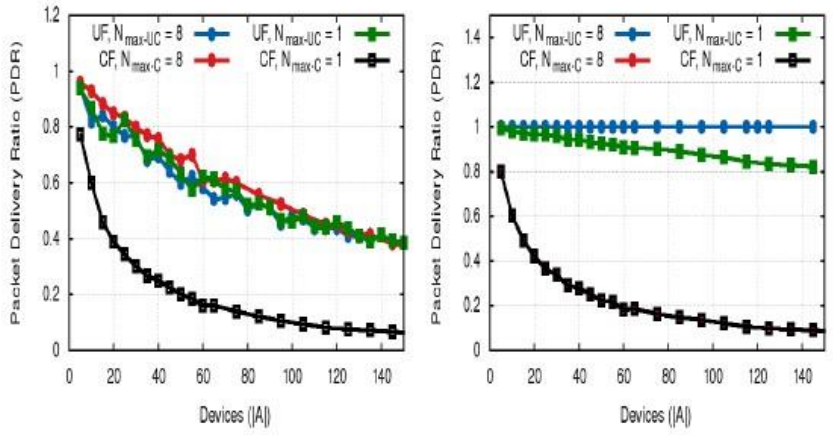


Figure 16 - Utilisation de la sous-bande par rapport au nombre total d'appareils (| A |) avec une intensité de trafic (t_I) variable pour les transmissions UL non confirmées / confirmées avec SF 12 fixe, $N_{max-UC} = 8$, $N_{max-C} = 8$, $m_c = 1$.

Comme prévu, lorsque la charge de trafic est élevée, l'utilisation augmente également pour les deux transmissions UL non confirmées / confirmées. Il est important d'observer sur la figure 9 que dans un environnement moins chargé, la transmission non confirmée permet toujours une meilleure utilisation du canal.

2) Ratio de livraison de paquets (PDR) : Ensuite, nous évaluons le PDR pour les transmissions de trames confirmées et non confirmées en présence de plusieurs canaux disponibles sur la sous-bande.

 Réseau entièrement chargé ($t_I = 1$) **avec des transmissions maximales variables** (N_{max-UC} **ou** N_{max-C}) : Les résultats de la figure 17a, b montrent l'effet sur PDR en raison du nombre de périphériques réseau (| A |), nombre maximal des transmissions et le nombre de canaux disponibles mc. Comme le montre la figure 17a avec une passerelle à canal unique (par exemple $m_c = 1$) et un réseau entièrement chargé, la transmission de trame UL confirmée donne des résultats relativement meilleurs par rapport aux autres lorsque les transmissions maximales sont de 8. Cependant, lorsque les transmissions maximales sont limitées à $N_{max-C} = 1$, le PDR chute pour deux raisons : premièrement, parce que les transmissions confirmées consomment plus de ressources (en raison des transmissions ACK) que les transmissions non confirmées. Deuxièmement, effectuer des transmissions une seule fois entraîne un taux de fin au niveau de la couche application et des trames UL plus infructueuses, en particulier lorsqu'il y a beaucoup de périphériques dans le réseau. Enfin, nous pouvons voir que dans un environnement de réseau entièrement chargé et lorsque la limite de transmission maximale est élevée, le choix de la transmission non confirmée / confirmée devient moins important, car ils atteignent tous les deux des valeurs PDR similaires [6].



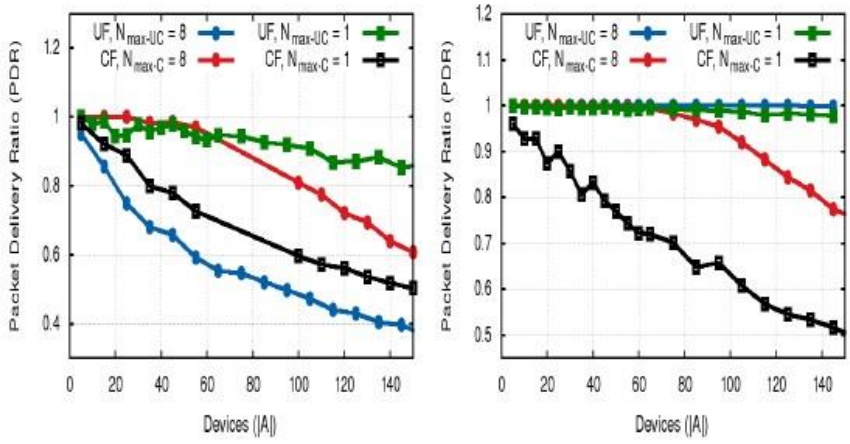
(a) PDR avec $m_c = 1, t_I = 1$ (b) PDR avec $m_c = 7, t_I = 1$

Figure 16 - PDR vs dispositifs réseau (| A |) pour les transmissions UL confirmées et non confirmées avec un nombre différent de transmissions maximales ($N_{max-UC}\ et\ N_{max-C}$) $et\ t_I = 1$.

La figure 17b montre les résultats de transmissions UL non confirmées et confirmées lorsque le nombre de canaux disponibles est augmenté à 7 (m_c = 7). Notez qu'avec des transmissions non confirmées, lorsque N_{max-UC} est 1 et 8, la couche d'application PDR est élevée par rapport au cas des transmissions UL confirmées. Surtout pour $N_{max-UC} = 8$, le nombre de transmissions de trames augmente les chances de réussite de la réception de trames, et donc le PDR reste proche de 100% pour jusqu'à 150 périphériques réseau. Dans le cas des seules transmissions UL confirmées avec une transmission maximale ($N_{\max-C}$) définie sur 8, nous remarquons que la disponibilité de 7 canaux diminue les risques de collisions et permet d'envoyer plus de nouvelles trames confirmées via UL avec succès. Cependant, en raison d'un nombre plus élevé de canaux disponibles pour les transmissions confirmées, le rapport des paquets reçus avec succès est également élevé, ce qui donne un PDR qui est le même que dans le cas des transmissions confirmées avec des transmissions maximales définies à 1 [6].

- **Charge réduite ($t_I = 0,10$) avec des transmissions maximales** variables (N_{max-UC} **ou** N_{max-C}) : De plus, nous avons exploré le cas d'une charge réseau modérée, avec une intensité de trafic de t_I= 0,10. Sur la figure 18a, les résultats d'un réseau de passerelle à canal unique montrent que lorsque la charge est élevée, le PDR des transmissions confirmées pour les deux cas de N_{max-C} 1 et 8 est faible par rapport à celui non confirmé lorsque N_{max-UC} est égal à 1. Un autre point important à noter est que lorsque la charge du réseau est faible (par exemple $t_I = 0,10$), même une seule transmission non confirmée est suffisante et permet d'obtenir un meilleur PDR par rapport aux transmissions répétitives UL non confirmées de la même trame (par exemple lorsque $N_{max-UC} = 8$). En effet, les transmissions répétitives non confirmées augmentent la charge du réseau et entraînent plus de collisions, réduisant ainsi efficacement le PDR, comme le montre la figure 17a.

  De la tendance intéressante dans des conditions de faible charge, nous pouvons conclure que lorsque les périphériques réseau sont inférieurs à 40, les transmissions UL confirmées avec une limite de transmission maximale élevée peuvent être utilisées pour obtenir une meilleure fiabilité. En comparant cela aux résultats antérieurs d'un réseau de passerelle à canal unique sur la figure 16a, un point important à noter est que lorsque le réseau est légèrement chargé, les transmissions répétitives des mêmes trames non confirmées augmentent la charge et entraînent plus de collisions. Sur la figure 17b, les résultats PDR basés sur une faible intensité de trafic ($t_I = 0,10$) avec tous les canaux disponibles ($m_c = 7$) montrent que lorsque la charge sur le canal est relativement faible, les résultats PDR suivent les modèles pour différents $N_{max-UC} = N_{max-C}$ et périphériques réseau, comme prévu. En d'autres termes, avec une limite de transmission maximale plus élevée et davantage de canaux disponibles, le PDR augmente pour les transmissions confirmées et non confirmées.



(a) PDR avec $m_c = 1, t_I = 1$ (b) PDR avec $m_c = 7, t_I = 1$

Figure 17 - PDR vs dispositifs réseau (| A |) pour les transmissions UL confirmées et non confirmées avec un nombre différent de transmissions maximales $(N_{max-UC}\ et\ N_{max-C})\ et\ t_I = 0{,}10$.

3. Les simulateurs open source

3.1. Tableau comparatif des simulateurs open source

Reference	Description	Imperfect SF	Capture effect	Downlink traffic	Duty cycle limitations	Energy consumption
[8, 15]	LoRaSim	No	Yes	No	No	Yes
[2]	FREE	Yes	Yes	Yes	Yes	Yes
[11]	Java, 915 MHz	No	Yes	No	Implements North American requirements	No
[20]	ns-3 module PHY focus	No	Yes	No	Yes	Yes
[13]	ns-3 module MAC focus	Yes	Yes	Yes	Yes	No
[22, 23]	C++ simulator	Yes	Yes	Yes	Yes	No
[24]	LoRaEnergySim	No	Presumably, is based on [15]	Yes	Yes	Yes
[25]	FLoRa	No	Yes	Yes	Yes	Yes

Tableau 3 - COMPARAISON ENTRE LES SIMULATEURS OPEN-SOURCE [7].

3.2. Récapitulatif

Différents cas d'utilisation de l'IoT ont des exigences différentes des réseaux sans fil qui les activent, ce qui à son tour influence les exigences lors du choix d'un simulateur approprié. Le tableau 4 montre comment les différents cas d'utilisation peuvent imposer des exigences différentes aux simulateurs LoRaWAN [7].

Use case	Description	Design requirements	Simulator requirements	Recommended simulators
Smart city traffic monitoring	Monitoring traffic requires a large amount of sensors over a large geographical urban area.	Network must be scalable, gateway locations must be planned and communication must be able to withstand a city's noisy radio conditions	Simulate large node and gateway numbers, use data traffic and city radio interference models.	[13, 23]
Smart cold chain monitoring	Monitor storage temperature and humidity of items such as food as well as tracking of shipments.	Network must be scalable, support mobile communication and provide a device's location.	Simulator must be able to accurately model large networks, LoRa's behaviour when moving (see [27]) and give location accuracy estimates.	[13, 23]
Environmental monitoring	Large scale continuous monitoring of the environment such as water quality monitoring or gas level monitoring in mines.	Long range communication, low power consumption and the ability for confirmed uplinks if required.	Simulate large geographically spread networks, accurate energy consumption capabilities and simulating the impact of downlink traffic.	[2]

Tableau 4 - CAS D'UTILISATION IoT ET LEURS EXIGENCES DE SIMULATEUR [7].

Le but de ce tableau n'est pas de décrire tous les cas d'utilisation possibles de l'IoT, mais plutôt d'illustrer qu'ils entraînent des exigences différentes pour le simulateur. Le tableau fait également quelques recommandations pour choisir le simulateur qui conviendra le mieux à chaque cas d'utilisation. Pour le premier cas d'utilisation, le simulateur présenté dans [13] permet de simuler plusieurs passerelles à des emplacements fixes et différents types de trafic. Ce simulateur est l'un des simulateurs les plus complets disponibles, avec l'inconvénient de ne pas calculer la consommation d'énergie. Un autre simulateur à considérer car il autorise plusieurs passerelles serait [11], en supposant que le réseau simulé fonctionnera en Amérique du Nord. Bien que **FREE** soit également un simulateur complet, il ne se concentre que sur un seul réseau de passerelle et ne conviendrait donc pas. Le deuxième cas d'utilisation a une exigence (suivi) qu'aucun simulateur ne prend actuellement en charge. L'impact de l'effet Doppler sur LoRa a été documenté dans [27], mais ces résultats n'ont pas été intégrés dans les simulateurs LoRaWAN. [13] ou [23] peuvent cependant être utilisés pour simuler le réseau pour voir comment il fonctionnerait lorsque tous les nœuds sont stationnaires. Pour le cas d'utilisation final, la surveillance environnementale, le simulateur **FREE** est recommandé. Ce simulateur prend en charge le trafic sur la liaison descendante et le schéma d'ordonnancement proposé mérite d'être étudié dans les situations de surveillance environnementale où la passerelle n'est présente que périodiquement.
Dans le cadre de nos recherches et à partir du tableau précédent, nous allons mieux parler sur LoRaSim et FREE qui est plus complet que LoRaSim, compte tenu de la capacité de liaison descendante et d'un modèle théorique d'erreur de paquet, etc [8].

3.2.1. Simulateur LoRaSim et LoRaWANSim

Le simulateur le plus populaire pour LoRaWAN est LoRaSim [4], qui est un simulateur d'événements discrets basé sur Python avec Bibliothèque de simulation SimPy.

De plus, le besoin de connectivité à faible puissance, longue portée et à faible coût pour répondre aux exigences des applications IoT a conduit à l'émergence de technologies de mise en réseau LPWA (Low Power Wide Area). La promesse de ces technologies de connecter sans fil un

grand nombre d'appareils géographiquement dispersés à faible coût continue d'attirer beaucoup d'attention dans les communautés académiques et commerciales. Plusieurs déploiements sont déjà en cours, même si les performances de ces technologies restent à comprendre. À la lumière de ces développements, des outils pour effectuer des «analyses de simulation» et des études de pré-déploiement sont nécessaires pour comprendre les implications des choix qui sont faits au moment de la conception. Bien qu'il existe plusieurs technologies prometteuses dans l'espace LPWA, ce document se concentre spécifiquement sur la technologie LoRa / LoRaWAN. En particulier, nous présentons LoRaWANSim, un simulateur qui étend l'outil LoRaSim pour ajouter la prise en charge du protocole MAC LoRaWAN, qui utilise une communication bidirectionnelle [9].

3.2.2. Le simulateur LoRaFREE

FREE est une approche de programmation à granularité fine (C'est une extension de LoRaSim) pour synchroniser les transmissions afin d'obtenir une collecte de données fiable et économe en énergie. La collecte des données a lieu à des intervalles périodiques / apériodiques connus de la passerelle et des terminaux. Afin de calculer le calendrier de chaque période de collecte de données, la passerelle doit connaître le nombre de terminaux qui ont des données à transmettre, la quantité de leurs données en mémoire tampon et estimer leur perte de chemin.

La passerelle diffuse ensuite le planning et synchronise tous les périphériques finaux avec la même référence de temps. Le calcul et la diffusion du programme et le maintien de la synchronisation se font en deux étapes, comme illustré au § II B.LIBRE prend en charge trois canaux de rapport cyclique de 1% comme dans LoRaWAN plus un canal de rapport cyclique de 10% dans la bande ISM de l'UE 868. Le rapport cyclique définit le temps maximum et le pourcentage maximum avec lesquels un appareil peut occuper un canal. Par exemple, un rapport cyclique de 1% entraîne un temps de transmission maximal de 36 s / h pour chaque appareil. De plus, un temps de silence minimum de 99T, où T est le temps de transmission des paquets, est requis après chaque transmission. Les trois canaux à 1% sont utilisés pour la liaison montante et la liaison descendante et le canal à 10% est uniquement dédié au trafic sur la liaison descendante. Avant que FREE n'ait construit un programme, les canaux sont accessibles de manière asynchrone, mais une fois que le programme est calculé et diffusé aux terminaux, l'accès aux canaux s'effectue de manière synchrone.

Les transmissions en FREE sont programmées de telle manière que les appareils avec le même facteur d'étalement transmettent séquentiellement et les appareils avec différents facteurs d'étalement transmettent simultanément. À cette fin, le temps est divisé en trames, chaque trame ayant un certain nombre de tranches de liaison montante et une tranche de liaison descendante à la fin d'une trame. Il y a six trames parallèles, correspondant aux six facteurs d'étalement de LoRa (7 - 12). Les périphériques par facteur d'étalement se voient attribuer des créneaux séparés dans la trame correspondante, où la planification reste la même dans les trames consécutives. Le terme arrondi par facteur d'étalement est utilisé pour indiquer les trames consécutives qui sont nécessaires pour collecter toutes les données. En plus des tours parallèles, FREE peut également prendre en charge plusieurs canaux par facteur d'étalement. Le tour, dans ce cas, commence par un retard de slot sur le deuxième canal, deux slots sur le troisième canal et ainsi de suite. En effet, les appareils ne peuvent pas transmettre simultanément sur plusieurs canaux.

Les appareils (utilisant ce facteur d'étalement particulier) transmettent dans chaque trame sur tous les canaux pris en charge, ce qui accélère le cycle de collecte de données correspondant.

La fente et les longueurs de trame sont égales pour le même facteur d'étalement mais peuvent ne pas être égales pour différents facteurs d'étalement. La longueur de créneau et le nombre de créneaux de liaison montante par trame dépendent de plusieurs facteurs, par ex. longueur des paquets, nombre d'appareils par facteur d'étalement correspondant, etc. Tous les facteurs qui affectent la conception de FREE sont examinés au § III. Avant de commencer la collecte de données, les entités du réseau (c'est-à-dire la passerelle et les terminaux) doivent convenir d'un calendrier unifié et d'une référence temporelle afin de synchroniser les transmissions.

4. Simulation avec LoRaSim

4.1. Syntaxe de LoRaSim

Tous les simulateurs fonctionnent principalement de la même manière et prennent les mêmes paramètres. La principale différence est que loraDirMulBS.py simule jusqu'à 24 stations de base :

./loraDir.py <NODES> <AVGSEND> <EXPERIMENT> <SIMTIME> [COLLISION]

./loraDirMulBS.py <NODES> <AVGSEND> <EXPERIMENT> <SIMTIME> <BASESTATIONS> [COLLISION]

./directionalLoraIntf.py <NODES> <AVGSEND> <EXPERIMENT> <BASESTATIONS> <SIMTIME> <COLLISION> <DIRECTIONALITY> <NETWORKS> <BASEDIST>

./oneDirectionalLoraIntf.py

<NODES> <AVGSEND> <EXPERIMENT> <BASESTATIONS> <SIMTIME> <COLLISION> <DIRECTIONALITY> <NETWORKS> <BASEDIST>

Description

NODES

Nombre de nœuds à simuler.

AVGSEND

Intervalle d'envoi moyen en millisecondes.

EXPERIMENT

L'expérience est un entier qui détermine avec quels paramètres radio la simulation est exécutée. Tous les nœuds sont configurés avec une puissance d'émission fixe et une fréquence d'émission unique, sauf indication contraire.

- **0** : utilisez les paramètres avec le débit de données le plus bas (SF12, BW125, CR4 / 8).
- **1** : similaire à l'expérience 0, mais utilise un choix aléatoire de 3 fréquences de transmission.
- **2** : utilisez les paramètres avec le débit de données le plus rapide (SF6, BW500, CR4 / 5).
- **3** : optimiser le paramètre par nœud en fonction de la distance à la passerelle.
- **4** : utilisez les paramètres définis dans LoRaWAN (SF 12, BW125, CR4 / 5).

- **5** : similaire à l'expérience 3, mais optimise également la puissance de transmission.

SIMTIME

Durée totale en millisecondes.

BASESTATIONS

Nombre de stations de base à simuler. Peut-être 1, 2, 3, 4,6, 8 ou 24.

COLLISION

Mis à 1 pour activer le contrôle de collision complet, 0 pour utiliser un contrôle simplifié (par défaut). Avec la vérification simplifiée, deux messages entrent en collision lorsqu'ils arrivent en même temps, sur la même fréquence et facteur d'étalement. Le contrôle de collision complet prend en compte «l'effet de capture», selon lequel l'un des deux messages entrant en collision peut toujours passer en fonction de la synchronisation relative et de la différence de puissance de réception.

DIRECTIONALITY

Mis à 1 pour activer l'antenne directionnelle pour les nœuds.

NETWORKS

Nombre de réseaux LoRa.

BASEDIST

Distance X entre deux stations de base.

4.2. Avantages de LoRaSim

- Le simulateur le plus populaire pour LoRaWAN.
- C'est un simulateur d'évènements discrets basé sur python avec la bibliothèque de simulation SimPy.
- Tous les autres simulateurs pour LoRaWAN sont basés sur LoRaSim, ces derniers étendent LoRaSim pour plusieurs applications IoT (Internet Of Things).

4.3. Inconvénients de LoRaSim

- Il ne prend pas en considération les acquittements (ACK).
- Les paramètres (SF, BW, CR…) ne sont pas paramétrés.
- Les calculs (énergie…) ne sont pas précis.
- Il ne prend pas en considération la taille des paquets. Les paramètres sont relatifs à chaque nœud.

5. Simulation avec FREE

FREE est l'un des simulateurs de LORAWAN basé sur LoRaSim.

La syntaxe de FREE est proche de celle de LoRaSim, le choix de FREE a été fait car c'est le simulateur le plus approprié selon les statistiques (voir tableau 3)

5.1. Avantages FREE

- Il prend en considération les acquittements (ACK).
- Les calculs (énergies, …) sont bien précis.
- Il prend en considération la taille des paquets.
- Les paramètres sont relatifs à chaque paquet.

5.2. Inconvénients de FREE

- Les paramètres (SF, BW, CR …) ne sont pas paramétrés.
- Il ne prend pas en considération le cas de multiples stations de base.

6. Simulateur AkramSim

J'ai eu l'idée de réaliser le nouveau simulateur (AkramSim), en se basant sur les avantages et les inconvénients des deux simulateurs étudiés (cités précédemment) et en faisant plusieurs simulations.

Les points essentiels que j'ai pris en considération pour la réalisation du nouveau simulateur sont les suivants :

- Il prend en considération les acquittements (ACK), les collisions. Les paramètres sont relatifs à chaque paquet.
- Les paramètres (SF, BW, CR…) sont paramétrés.
- Il prend en considération le cas de multiples stations de base.
- Les calculs (énergie…) sont plus précis (en tenant compte de tous les paramètres).
- La récupération des données peut se faire à partir des fichiers.

6.1. Les exécutables

6.1.1. OneBS

C'est un exécutable pour la simulation avec une seule station de base **Usage**

:

Nous avons deux modes d'exécution :

- Mode compatible LoRaSim
- Mode AkramSim

Mode compatible LoRaSim

./OneBS.py <nodes> <avgsend> <experiment> <simtime> <datasize> [collision]

D'où :

Nodes : nombre de nœuds.

Avgsend : le temps moyen de transmission

Experiment : les experiments comme définis dans LoRaSim.

Simtime : durée totale de la simulation en millisecondes.

dataSize : c'est la taille maximale du paquet à transmettre.

Mode AkramSim:

./OneBS.py <nodes> <avgsend> <simtime> <datasize> <SF> <BW> [collision]

D'où :

Facteur d'étalement et bande passante choisis par l'utilisateur (SF, BW) Les

autres paramètres sont les mêmes que ceux dans LoRaSim.

6.1.2. MulBS

C'est un exécutable pour la simulation avec multiples stations de base.

Nous avons aussi deux modes d'exécution :

Mode compatibles LoRaSim
Mode AkramSim.

Usage :

./MulBS.py <nodes> <avgsend> <experiment> <simtime> <nrBS> <datasize> <SF> <BW> [collision] D'où

:

Nodes : nombre de nœuds.

Avgsend : le temps moyen de transmission

Experiment : les experiments comme définis dans LoRaSim.

Simtime : durée totale de la simulation en millisecondes.

dataSize : c'est la taille maximale du paquet à transmettre. nrBS

: nombre de stations de base.

SF : facteur d'étalement. BW

: bande passante

6.1.3. OneBSFichier et MulBSFichier

OneBSFichier est un exécutable pour la simulation avec une seule station de base sauf que les données sont récupérées à partir d'un fichier que j'ai nommé « nœuds.txt » (la première ligne du fichier représente les informations concernant la simulation tel que le nombre de nœuds, durée de la simulation… et les autres lignes représentent les informations qui concernant chaque nœud).

MulBSFichier est un exécutable pour la simulation avec une seule station de base sauf que les données sont récupérées à partir d'un fichier que j'ai nommé « MulNœuds.txt » (la première ligne du fichier représente les informations concernant la simulation tel que le nombre de nœuds, durée de la simulation… et les autres lignes représentent les informations qui concernant chaque nœud).

6.1.4. OneBSA

C'est un exécutable pour la simulation avec une seule station de base avec une contrainte sur le taux de collision (i.e, si le nombre de collisions dépasse un certain seuil, 3 versions sont possibles) et nous allons les expliquer dans ce qui suit :

- **Version 1 :** si le nombre de collisions dépasse un certain seuil, alors je vais tirer au hasard un autre SF et c'est le même cas pour les autres paramètres (BW, CR).

```
# 1ère version ok
print "J'ai changé de SF : node", node.nodeid
node.packet.sf = random.randint(6,12)
node.packet.cr = random.randint(1,4)
node.packet.bw = random.choice([125, 250, 500])
```

- **Version 2 :** si le nombre de collisions dépasse un certain seuil, alors je vais tirer au hasard un nombre qui varie entre [0,1] et si la probabilité de ce nombre est supérieure ou égale à 0,40 alors je vais changer les paramètres de la même manière que la version 1 sinon je ne fais rien.

```
# 2ème version ok

p = random.randint(0,1)
if p >= 0.40:
     print "J'ai changé de SF : node", node.nodeid
     node.packet.sf = random.randint(6,12)
     node.packet.cr = random.randint(1,4)
     node.packet.bw = random.choice([125, 250, 500])
else:
     print "J'ai pas changé de SF je retante avec le meme SF : node", node.nodeid
```

- **Version 3 :** si le nombre de collisions dépasse un certain seuil (*) :
  - Si le paquet rentre en collision, alors je diminue la valeur du SF utilisé.
  - Si le paquet est bien reçu par la station de base, alors j'augmente la valeur du SF utilisé

  Si (*) est vérifié alors la valeur du SF sera changé par le celui qui a la plus grande note.

```
# 3ème version
taille = len(node.SFs)
max = 0
for i in range(1,taille):
    if node.SFs[max] <= node.SFs[i]:
       max = i
node.packet.sf = node.SFs[max] + 7
print "J'ai changé le SF du noeud n° ",node.nodeid
print "Nouvelle SF ", node.packet.sf
```

6.2. Explication du code

Dans cette section, nous allons expliquer les différents bouts de code du nouveau simulateur AkramSim afin de mieux comprendre la procédure de simulation :

6.2.1. Les collisions

Frequency collision

Une collision de tel type est détectée, si l'un des cas suivants est réalisé

- Si la différence de fréquences des deux paquets est inférieure ou égale à 120KHz et la bande passante des deux paquets est égale à 500, sinon ;
- Si la différence de fréquences des deux paquets est inférieure ou égale à 60KHz et la bande passante des deux paquets est égale à 250, sinon ;
- Si la différence de fréquences des deux paquets est inférieure ou égale à 30KHz.

```
def frequencyCollision(p1,p2):
    if (abs(p1.freq-p2.freq)<=120 and (p1.bw==500 and p2.bw==500)):
        print "frequency coll 500"
        return True
    elif (abs(p1.freq-p2.freq)<=60 and (p1.bw==250 and p2.bw==250)):
        print "frequency coll 250"
        return True
    else:
        if (abs(p1.freq-p2.freq)<=30):
            print "frequency coll 125"
            return True
        #else:
    print "no frequency coll"
    return False
```

Figure 18 _ La fonction frequencyCollision.

Sf collision

Si les valeurs des Sfs des deux paquets sont égaux alors une collision de type sf collision est détectée

```
def sfCollision(p1, p2):
    if p1.sf == p2.sf:
        print "collision sf node {} and node {}".format(p1.nodeid, p2.nodeid)
        # p2 may have been lost too, will be marked by other checks
        return True
    print "no sf collision"
    return False
```

Figure 19 _ La fonction sfCollision.

PowerCollision

Une collision de tel type détectée, si l'un des cas suivants est réalisé :

- La valeur absolue de la différence entre le rssi des deux paquets est inférieure à un seuil de 6 dB (powerThreshold), dans ce cas les deux paquets sont trop proches l'un de l'autre, les deux entrent en collision, renvoyer les deux paquets comme victimes, Sinon ;
- Si, la différence entre le rssi des deux paquets est strictement inférieure à un seuil de 6 dB (p1.rssi – p2.rssi < powerThreshold), dans ce cas p2 surpuissant p1, renvoyer p1 comme victime.
- Sinon ; p2 était le paquet le plus faible, renvoyer p2 comme victime.

```
def powerCollision(p1, p2):
    powerThreshold = 6 # dB
    if abs(p1.rssi - p2.rssi) < powerThreshold:
        # packets are too close to each other, both collide
        # return both packets as casualties
        return (p1, p2)
    elif p1.rssi - p2.rssi < powerThreshold:
        # p2 overpowered p1, return p1 as casualty
        return (p1,)
    # p2 was the weaker packet, return it as a casualty
    return (p2,)
```

Figure 20 _ La fonction powerCollision.

TimingCollision

En supposant que p1 est le paquet fraîchement arrivé et que c'est la dernière vérification, nous avons déjà déterminé que p1 est un paquet faible, donc la seule façon de gagner est d'être suffisamment en retard (seuls les n-5 premiers symboles de préambule se chevauchent).

En supposant que 8 symboles de préambule le rendent comme une victime.

```
def timingCollision(p1, p2):
    # assuming p1 is the freshly arrived packet and this is the last check
    # we've already determined that p1 is a weak packet, so the only
    # way we can win is by being late enough (only the first n - 5 preamble symbols overlap)

    # assuming 8 preamble symbols
    Npream = 8

    # we can lose at most (Npream - 5) * Tsym of our preamble
    Tpreamb = 2**p1.sf/(1.0*p1.bw) * (Npream - 5)

    # check whether p2 ends in p1's critical section
    p2_end = p2.addTime + p2.rectime
    p1_cs = env.now + Tpreamb
    print "collision timing node {} ({},{},{}) node {} ({},{})".format(
        p1.nodeid, env.now - env.now, p1_cs - env.now, p1.rectime,
        p2.nodeid, p2.addTime - env.now, p2_end - env.now
    )
    if p1_cs < p2_end:
        # p1 collided with p2 and lost
        print "not late enough"
        return True
    print "saved by the preamble"
    return False
```

Figure 21 – La fonction timingCollision.

6.2.2. CheckCollision

Cette fonction permet de vérifier si le paquet est rentré en collision avec tous les autres paquets qui sont déjà à la station de base.

```
# check for collisions at base station
# Note: called before a packet (or rather node) is inserted into the list
def checkcollision(packet):
    col = 0 # flag needed since there might be several collisions for packet
    processing = 0
    for i in range(0,len(packetsAtBS)):
        if packetsAtBS[i].packet.processed == 1:
            processing = processing + 1
    if (processing > maxBSReceives):
        print "too long:", len(packetsAtBS)
        packet.processed = 0
    else:
        packet.processed = 1

    if packetsAtBS:
        print "CHECK node {} (sf:{} bw:{} freq:{:.6e}) others: {}".format(
             packet.nodeid, packet.sf, packet.bw, packet.freq,
             len(packetsAtBS))
        for other in packetsAtBS:
            if other.nodeid != packet.nodeid:
               print ">> node {} (sf:{} bw:{} freq:{:.6e})".format(
                   other.nodeid, other.packet.sf, other.packet.bw, other.packet.freq)
               # simple collision
               if frequencyCollision(packet, other.packet) \
                   and sfCollision(packet, other.packet):
                   if full_collision:
                       if timingCollision(packet, other.packet):
                           # check who collides in the power domain
                           c = powerCollision(packet, other.packet)
                           # mark all the collided packets
                           # either this one, the other one, or both
                           for p in c:
                               p.collided = 1
                               if p == packet:
                                   col = 1
                       else:
                           # no timing collision, all fine
                           pass
                   else:
                       packet.collided = 1
                       other.packet.collided = 1  # other also got lost, if it wasn't lost already
                       col = 1
        return col
    return 0
```

Figure 22 _ La fonction checkCollision.

6.2.3. Transmit

La méthode transmit permet pour chaque nœud la transmission du paquet et vérifier si le paquet n'est pas perdu, si ce n'est pas le cas alors vérification si y a pas eu de collision (avec la méthode checkCollision) avec tous les autres nœuds.

Ensuite, si le paquet n'est pas perdu et n'est pas rentré en collision et de plus, il est acquitté (checkACK), alors ajouter ce paquet comme un paquet reçu par la station de base.

```
def transmit(env,node):
    while True:
        yield env.timeout(random.expovariate(1.0/float(node.period)))

        # time sending and receiving
        # packet arrives -> add to base station

        node.sent = node.sent + 1

        global packetSeq
        packetSeq = packetSeq + 1

        global nrBS
        for bs in range(0, nrBS):
            if (node in packetsAtBS):
                print "ERROR: packet already in"
            else:
                sensitivity = sensi[node.packet[bs].sf - 7, [125,250,500].index(node.packet[bs].bw) + 1]
            if node.packet[bs].rssi < sensitivity:
                print "node {}: packet will be lost".format(node.nodeid)
                node.packet[bs].lost = True
            else:
                node.packet[bs].lost = False
                if (per(node.packet[bs].sf,node.packet[bs].bw,node.packet[bs].cr,node.packet[bs].rssi,node.packet[bs].pl) < random.uniform(0,1)):
                    # OK CRC
                    node.packet[bs].perror = False
                else:
                    # Bad CRC
                    node.packet[bs].perror = True
                # adding packet if no collision

                if (checkcollision(node.packet[bs])==1):
                    node.packet[bs].collided = 1

                else:
                    node.packet[bs].collided = 0
            yield env.timeout(node.packet[0].rectime)

            if (node.packet[bs].lost == 0\
                and node.packet[bs].perror == False\
                and node.packet[bs].collided == False\
                and checkACK(node.packet[bs])):
                 node.packet[bs].acked = 1
            # the packet can be acked
            # check if the ack is lost or not
                 if((14 - Lpld0 - 10*gamma*math.log10(node.distances[bs]/d0) - np.random.normal(-var, var)) > sensi[node.packet[bs].sf-7, [125,250,500].index(node.packet[bs].bw) + 1]):
                  # the ack is not lost
                    node.packet[bs].acklost = 0
                 else:
            # ack is lost
                    node.packet[bs].acklost = 1
            else:
                node.packet[bs].acked = 0
```

Figure 23 _ La fonction transmit.

6.3. Résultats de simulations

Dans cette partie, nous allons faire une petite comparaison des trois différentes versions afin de critiquer et de tirer la meilleure version :

6.3.1. Durée de simulation

La figure 24 montre que plus la durée de simulation est grande plus le nombre de collisions sera plus grand pour les trois versions. Par contre, avec les mêmes données fournies avant le début de la simulation, on remarque que la version 3 est meilleure que les deux autres versions du coté de nombre de collisions.

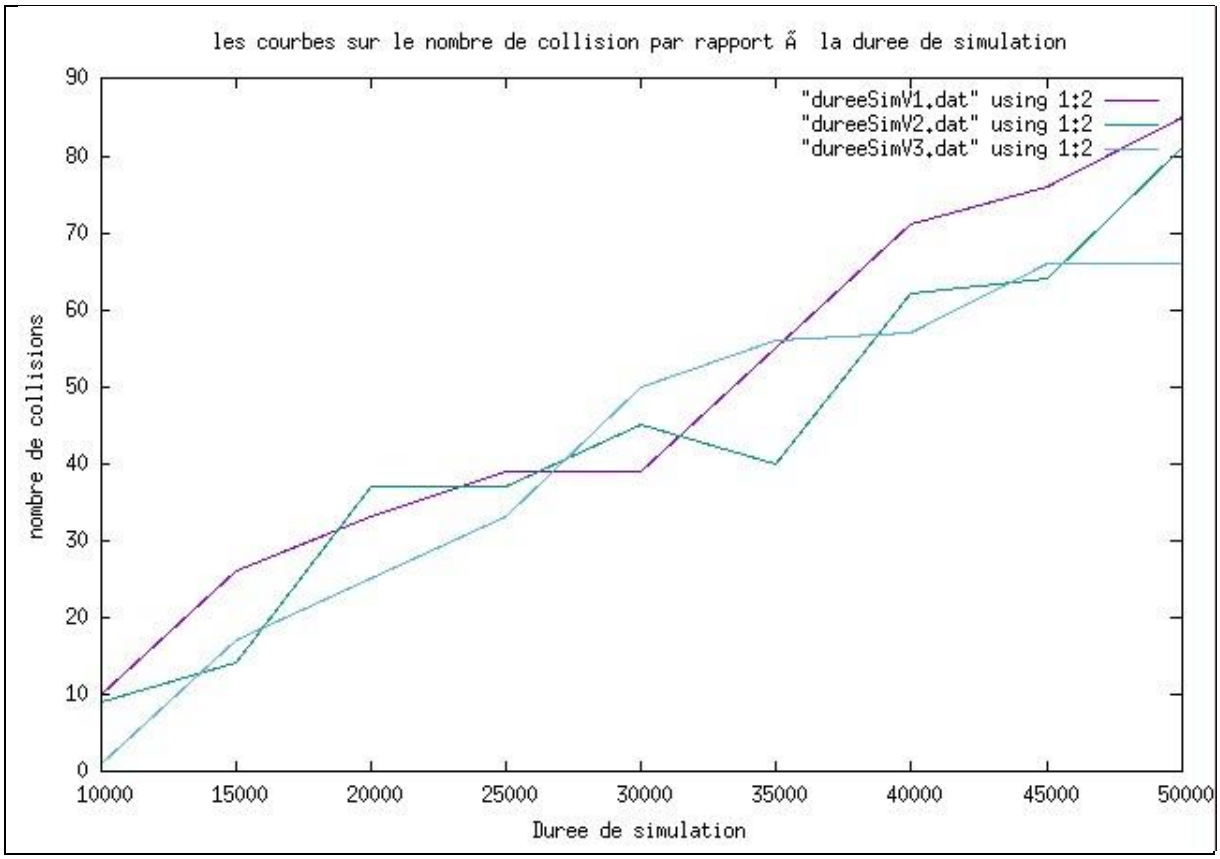


Figure 24 – nombre de collisions par rapport à la durée de simulation.

Conclusion

On a donc une preuve de concept de réseau IOT LoRaWAN Open Source simple mais fonctionnelle qui peut facilement être étendue par l'ajout d'autres capteurs et l'amélioration du

connecteur MQTT. Cependant pour un véritable environnement de production, les passerelles mono canal sont à proscrire au profit de passerelles entièrement compatibles avec la spécification LoRaWAN, permettant notamment de recevoir et émettre simultanément sur plusieurs canaux et avec des facteurs d'étalement différents.

La technologie LoRa et le protocole LoRaWAN bien que encore relativement jeunes constituent une alternative viable au leader actuel du marché des réseaux IOT : SigFox. Si les performances radios des deux concurrents sont relativement proches, leurs modèles économiques sont diamétralement opposés. L'approche plus ouverte de LoRaWAN séduit, mais les multiples réseaux publics n'égalent pas encore la couverture de SigFox. De plus la démocratisation prochaine des standards LTE-M et NB-IOT risque de transformer significativement le marché.

Dans le cadre de ce projet, nous étions amenés à concevoir un simulateur pour le contrôle des performances des réseaux LoRaWAN. Durant la réalisation de ce projet, nous avons effectué dans un premier temps une étude théorique des réseaux LoRaWAN. Ensuite, nous avons détaillé les simulateurs des réseaux LoRa qui sont déjà déployés. Finalement, nous avons introduit le nouveau simulateur « AkramSim » ainsi que les résultats de simulations avec ce dernier.

Ce projet nous a permet de nous familiariser avec les nouvelles technologies, surtout les réseaux LoRaWAN, d'étudier les performances des réseaux LoRaWAN (énergie…) et de faire des simulations en utilisant le simulateur le plus approprié tout en réalisant un nouveau simulateur.

Bibliographie

[1] https://www.objetconnecte.com/tout-savoir-reseau-lora-bouygues.

[2] https://www.linuxembedded.fr/2017/12/introduction-a-lora.

[3] LoRa-MAB: A Flexible Simulator for Decentralized Learning Resource Allocation in IoT Networks.

[4] Un Algorithme pour le Problème des Bandits Manchots avec Stationnarité par Parties, Robin Allesiardo et Raphael Féraud, Orange Labs, 2 av. Pierre Marzin, 22300 Lannion 2TAO - INRIA, LRI, Université Paris-Sud, CNRS, 91405 Orsay.

[5] IoT scheduling for higher throughput and lower transmission power, Husam Rajab, Taoufik Bouguera et Tibor Cinkler, publié en mars 2020.

[6] Experimental Evaluation of LoRaWAN in NS-3, Furqan Hameed Khan et Marius Portmann.

[7] A Review of LoRaWAN Simulators: Design Requirements and Limitations, Jaco M. Marais, Adnan M. Abu-Mahfouz, Gerhard P. Hancke

[8]https://www.researchgate.net/post/Im_looking_for_a_good_Simulator_compatible_with_LoRaWAN_protocol_please.

[9]https://www.researchgate.net/publication/316089079_Does_Bidirectional_Traffic_Do_More_Harm_Than_Good_in_LoRaWAN_Based_LPWA_Networks.