



UNIVERSITÉ
SAVOIE
MONT BLANC

LoRa[®] - LoRaWAN[®] et l'Internet des Objets



Sylvain MONTAGNY

**UNE TECHNOLOGIE SANS FIL, BASSE
CONSOMMATION ET LONGUE PORTÉE**



"This book is a tremendous resource for anyone interested in LoRaWAN technology. You will simply discover why LoRaWAN is the premier leading solution for large scale LPWAN deployments. Many thanks to the Savoie Mont Blanc University team on behalf of the LoRa Alliance."

Mme Donna Moore, PDG et Présidente de l'**Alliance LoRa**

Relecteurs

SEMTECH : Olivier Seller, Damian Gabino Rascon, Joseph Knapp

Université Savoie Mont Blanc : Florent Lorne, Antoine Augagneur, Marie-Line Fournier

À propos de ce livre

Ce livre est le fruit d'un travail réalisé par l'équipe pédagogique des Masters ESET ([Electronique des Systèmes Embarqués](#)) et Master TRI ([Télécoms et Réseaux Informatiques](#)) à l'[université Savoie Mont Blanc](#). Toutes remarques, modifications, améliorations, corrections peuvent être proposées via la page de contact de notre site web : www.univ-smb.fr/lorawan/contact/

Ce livre cite de nombreuses marques (Device, LoRaWAN Server, IoT Platform...) et l'université a travaillé avec des partenaires industriels pour la réalisation de ces ressources pédagogiques. Néanmoins, aucune entreprise n'a financé ces documents/vidéos/formations et c'est en toute objectivité que ce travail a été réalisé.

En janvier 2021, l'Université Savoie Mont Blanc a présenté la meilleure façon d'apprendre LoRaWAN dans un Webinar. Vous pouvez voir le [Replay](#) ici (version française uniquement) ou sur le site de [Chipselect](#).

Ce livre sur l'Internet des Objets et les protocoles LoRa / LoRaWAN est mis à jour régulièrement. Sur notre site web, vous pouvez souscrire pour recevoir une notification par email lorsque que de nouveaux éléments ont été ajoutés. De plus le site web regroupe de nombreux outils, informations et ressources à votre disposition.

@siteweb

A propos de l'auteur

Sylvain MONTAGNY



Je suis professeur agrégé à l'Université Savoie Mont Blanc depuis 2006 et je suis spécialisé dans les systèmes à microprocesseurs et l'Internet des objets. Je suis responsable du Master en Electronique et Systèmes Embarqués dans lequel la plupart des cours ont une relation étroite avec l'industrie. Les nouvelles tendances éducatives nous incitent à proposer des contenus en ligne de qualité. J'ai relevé le défi d'écrire un livre avec des informations claires et simples et j'espère que ce travail vous donnera entière satisfaction.

L'Université Savoie Mont Blanc est membre institutionnel de la [LoRa Alliance](#) depuis 2021.



Cours LoRa-LoRaWAN en vidéo

Malgré le contenu complet de ce livre, une version plus interactive est disponible en vidéo sur une plateforme de [e-Learning \(UDEMY\)](#).

LoRa et LoRaWAN pour l'Internet des Objets

Maîtrisez la transmission LoRa / LoRaWAN, du Device jusqu'à l'application utilisateur

Meilleure vente 4,9 ★★★★★ (83 notes) 276 participants

Créé par [Sylvain MONTAGNY](#), [Florent LORNE](#)

Ce cours est composé de 132 vidéos d'environ 4 minutes chacune. Il comprend des informations complémentaires et des détails sur la configuration des Devices, des Gateways et des serveurs. Vous pouvez poser vos questions sur la plateforme, les formateurs sont là pour y répondre.



Une heure de ces [vidéos LoRaWAN](#) est disponible gratuitement.



Vous pouvez demander un bon gratuit à des fins éducatives uniquement.



Pour l'instant, ces vidéos ne sont disponibles qu'en français. La version anglaise sera disponible sous peu.

Formation LoRaWAN (en distanciel)

Une formation de 1 + 2 jours 100% en ligne avec des formateurs est également proposée (uniquement en français - disponible en anglais sur demande).

- ➔ Équipement fourni : Nous vous envoyons un Device LoRaWAN et une Gateway pour créer votre propre réseau LoRaWAN. Vous conservez tout l'équipement à la fin de la formation.
- ➔ Nombre réduit de participants : Nous souhaitons personnaliser au maximum le contenu afin de répondre au mieux à vos besoins (maximum de 10 participants).
- ➔ Suivi personnalisé : Nous sommes à votre disposition tout au long de la formation, et nous restons disponibles même après pour répondre à toutes vos questions.

Avant la formation

- Vous avez accès à la plateforme pédagogique e-Learning
- Vous recevez le kit pédagogique et avez accès aux serveurs



Pendant la formation

- Suivi de tous les participants.
- Mise en place d'une transmission LoRa.
- Test des paramètres de transmission (bande passante, canaux, Spreading Facteur).
- Configuration de votre Gateway, de votre serveur LoRaWAN. Enregistrement des Devices.
- Test des deux modes d'activation (ABP et OTAA).
- Spécification des Devices de classe A, transmission uplink, downlink, confirmed ou non.
- ADR (Adaptive Data Rate).
- Exportation des données avec les protocoles HTTP et MQTT.
- Création de votre propre serveur LoRaWAN (The Things Stack v3 et Chirpstack).
- Création de votre propre plateforme IoT.



Toutes les informations sur cette formation sont disponibles sur notre site web.

Légendes



→ Liste d'informations importantes



→ A lire très attentivement



Notes



Exercices



Validation d'un résultat



Vidéos disponibles www.univ-smb.fr/lorawan/

Abréviations et acronymes

ABP	Activation By Personalization
ADR	Adaptive Data Rate
AS	Application Server
AppEUI	Application Extended Unique Identifier
AppKey	Application Key
AppSKey	Application Session Key
BW	Bandwidth
CDMA	Code Division Multiple Access
CHIRP	Compressed High Intensity Radar Pulse
CID	Command Identifier (MAC Command)
CR	Coding Rate
CRC	Check Redundancy Cycle
DevAddr	Device Address
DevEUI	Device Extended Unique Identifier
FDM	Frequency Division Multiplexing
FFT	Fast Fourier Transform
HTTP	HyperText Transfer Protocol
HSM	Hardware Security Module
IoT	Internet of Things
JSON	JavaScript Object Notation
JoinEUI	Join Extended Unique Identifier
JS	Join Server
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPWAN	Low Power Wide Area Network.
LTE-M	Long Term Evolution Cat M1
MIC	Message Integrity Control
MQTT	Message Queuing Telemetry Transport
NB-IoT	NarrowBand Internet of Things
NS	Network Server
NwkSKey	Network Session Key
OTAA	Over The Air Activation
QoS	Quality of Service
RSSI	Received Signal Strength Indication.
SDR	Software Digital Radio
SE	Secure Element
SF	Spreading Factor
SNR	Signal Over Noise Ratio
TDM	Time Division Multiplexing
TOA	Time One Air
TTI	The Things Industries
TTN	The Things Network
TTS	The Things Stack

Sommaire

1	LES SYSTEMES EMBARQUES ET L'IOT	9
1.1	L'INTERNET DES OBJETS (IOT)	9
1.2	LES MODES DE PARTAGE DU SUPPORT	11
1.3	L'ETALEMENT DE SPECTRE EN UTILISANT DES CODES.....	12
2	TRANSMISSION RADIO ET PROPAGATION	16
2.1	UNITES ET DEFINITIONS.....	16
2.2	DISTANCE DE TRANSMISSION EN LoRA.....	19
2.3	DOCUMENTATION D'UN TRANSCIVEUR LoRA	19
3	LA MODULATION LoRA (COUCHE PHYSIQUE).....	21
3.1	LA MODULATION LoRA	21
3.2	DEBIT BINAIRE EN LoRA ET LoRAWAN	25
3.3	LA SIMULATION D'UNE TRANSMISSION LoRA	31
3.4	TEST REEL D'UNE TRANSMISSION LoRA	35
3.5	CONSOMMATION D'ENERGIE	36
4	LE PROTOCOLE LoRAWAN	37
4.1	LoRA - LoRAWAN - LoRA ALLIANCE	37
4.2	LA STRUCTURE D'UN RESEAU LoRAWAN	38
4.3	LES CLASSES DE DEVICES LoRAWAN	45
4.4	ACTIVATION DES DEVICES LoRAWAN : ABP ET OTAA	49
4.5	AVANTAGES ET INCONVENIENTS DE L'ABP ET DE L'OTAA.....	55
4.6	TYPES DE TRAMES LoRAWAN	59
4.7	LES COMMANDES MAC.....	64
4.8	DATA RATE, CANAUX ET PUISSANCE.....	66
5	LES RESEaux LoRAWAN ET LES SERVEURS LoRAWAN.....	74
5.1	LES DIFFERENTS TYPES DE RESEaux.....	74
5.2	LA CONFIGURATION D'UN RESEAU LoRAWAN	78
6	LA TRAME LoRA / LoRAWAN	82
6.1	LES COUCHES DE PROTOCOLE LoRAWAN	82
6.2	LA COMMUNICATION ENTRE LES GATEWAYS ET LE NETWORK SERVER	85
6.3	L'ANALYSE DES TRAMES IP.....	89
7	EXPORTER NOS DONNEES DU SERVEUR LoRAWAN	95
7.1	LES SERVICES FOURNIS PAR LA PLATEFORME IOT	95
7.2	EXPORTATION DES DONNEES AVEC LE PROTOCOLE HTTP GET	97
7.3	EXPORTATION DE DONNEES AVEC LE PROTOCOLE HTTP POST	101
7.4	PRESENTATION DU PROTOCOLE MQTT.....	104
7.5	EXPORTATION DES DONNEES AVEC LE PROTOCOLE MQTT	110
7.6	UTILISATION D'UNE PLATEFORME IOT.....	113
8	CONCEVOIR VOTRE PROPRE DEVICE LoRAWAN	116
8.1	LES STACKS LoRAWAN DISPONIBLES.....	116
8.2	ARCHITECTURE "MICROCONTROLEUR + TRANSCIVEUR"	116
8.3	ARCHITECTURE "MODULE LoRAWAN AUTONOME".....	118
8.4	ARCHITECTURE "MICROCONTROLEUR + LoRAWAN MODULE"	120
8.5	ARCHITECTURE "MICROCONTROLEUR LoRAWAN"	121

8.6	RESUME DES ARCHITECTURES.....	122
9	CONFIGURER VOTRE PROPRE SERVEUR LORAWAN	123
9.1	INFORMATIONS PRELIMINAIRES	123
9.2	SERVEUR LORAWAN CHIRPSTACK	126
9.3	CONFIGURATION DE CHIRPSTACK	128
10	INSTALLER VOTRE PROPRE APPLICATION UTILISATEUR- PLATEFORME IOT.....	131
10.1	APERÇU DES CHOIX	131
10.2	CONSTRUIRE UNE PLATEFORME IOT	133
11	LORAWAN AVANCE	135
11.1	LE JOIN SERVEUR.....	135

1 Les systèmes embarqués et l'IoT

Le terme IoT (Internet of Things) est récent, mais fait référence à un usage ancien appelé Machine to Machine. Le **Machine to Machine** (M2M) est un ensemble de technologies de réseaux câblés ou sans fil permettant l'échange automatique d'informations entre systèmes sans intervention humaine. L'IoT est simplement une vision élargie du M2M où les dispositifs ne sont pas seulement utilisés dans le monde industriel, mais également dans la sphère publique.

Le marché de l'IoT augmente de manière très significative dans le monde et cette évolution rapide encourage de nouveaux acteurs à proposer de nouvelles technologies dans de nombreux domaines : matériels informatiques, couverture réseau et connectivité cloud pour le stockage et la disponibilité des données.

L'IoT est souvent présenté comme la nouvelle révolution industrielle, et le marketing fait énormément de promesses à ce sujet en indiquant que tous les cas d'usage seront intelligents ou "smart" : smart building, smart city, smart agriculture... Mais rendre les choses "smart" n'est pas toujours simple et de nombreux protocoles existent. Dans ce livre, nous vous permettrons de comprendre l'un des principaux protocoles du monde de l'IoT : le LoRaWAN.

1.1 L'internet des objets (IoT)

1.1.1 Les systèmes embarqués dans l'IoT

De manière générale, les systèmes électroniques peuvent être caractérisés par leur **consommation** (électrique), leur **puissance de calcul**, leur **taille** et leur **prix**. Dans le cas précis des systèmes embarqués utilisés dans l'IoT, nous pouvons attribuer l'importance suivante à chacune de ces caractéristiques :

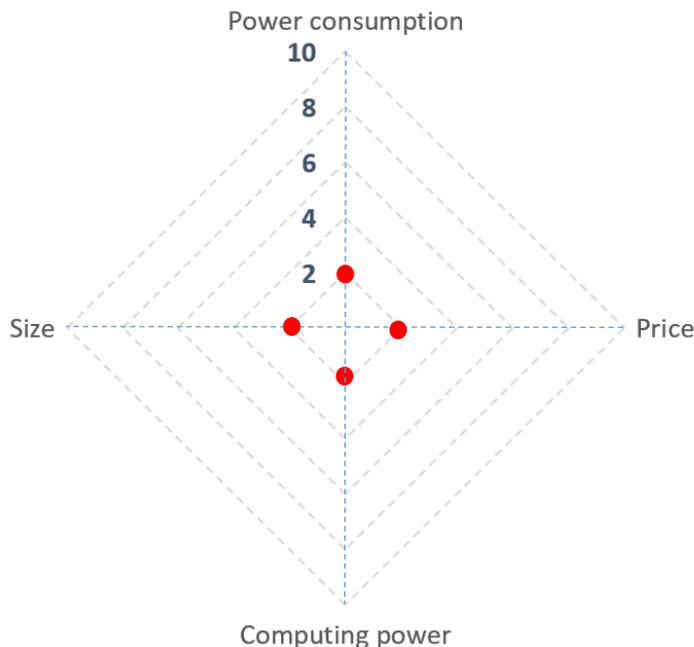


Figure 1: Caractéristiques des Devices IoT

Même si vous pouvez évidemment trouver des exceptions à cette définition simple, nous partons du principe que, par rapport à d'autres systèmes électroniques, les systèmes embarqués utilisés dans l'IoT ont :

- Une faible consommation d'énergie
- Une faible puissance de calcul
- Une petite taille
- Un prix bas

La deuxième caractéristique d'un dispositif IoT est sa capacité à transmettre des données sur un réseau sans fil. De nombreux protocoles existent et le concepteur aura un large choix en fonction de la portée, de la bande passante et de la consommation d'énergie qu'il souhaite atteindre.

1.1.2 Les protocoles sans fil de l'IoT

Dans le monde de l'IoT, nous pouvons trouver de nombreux protocoles tels que Bluetooth, Zigbee, WiFi, 2G, 3G, 4G, 5G, NFC... Nous les classons généralement en fonction de leur débit et de leur portée, comme nous pouvons le voir sur la Figure 2. En tant que concepteur, nous sommes toujours satisfaits d'obtenir une plus grande portée et un plus grand débit. Si l'on jette un coup d'œil rapide à l'illustration, on peut penser que les protocoles de la zone supérieure droite sont bien meilleurs (meilleure portée et meilleur débit). Le protocole en bas à gauche devrait donc être le plus mauvais (portée quasi nulle et débit très faible). Ce qui nous échappe ici, c'est que ce graphique ne représente pas la consommation d'énergie induite par le protocole. En effet, le NFC est un protocole à très faible consommation d'énergie : Il fonctionne même sans aucune batterie ou autre stockage puisqu'il est alimenté par le champ magnétique utilisé pour la transmission.

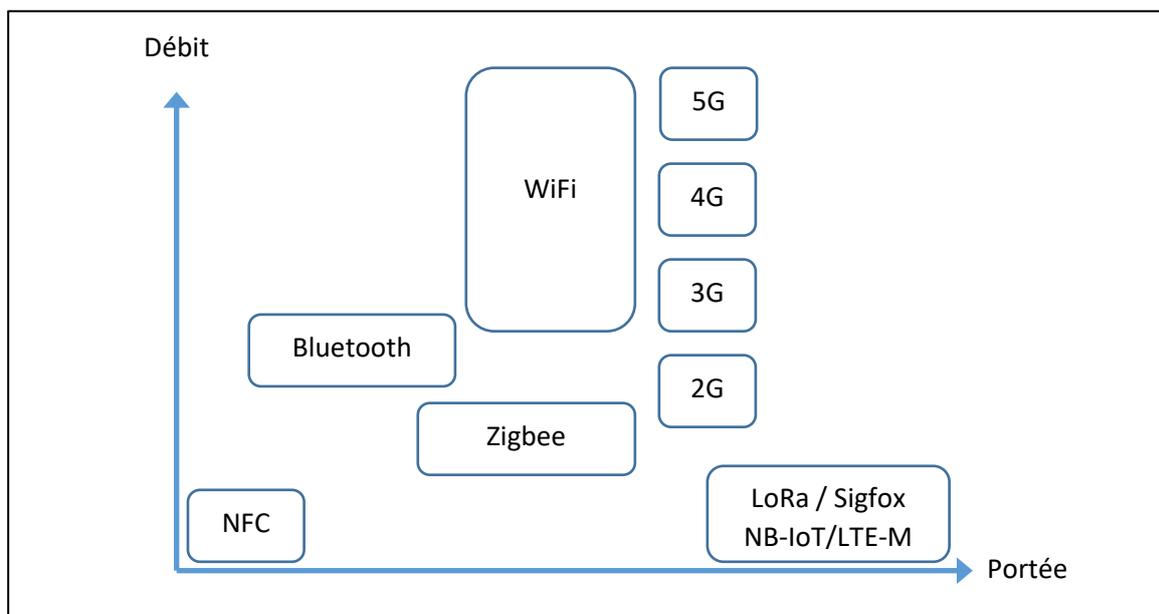


Figure 2: Protocoles utilisés dans l'IoT

En bas à droite, nous trouverons des protocoles à faible débit, à grande portée, mais globalement à faible consommation comme le NB-IoT ou LTE-M. Sigfox et LoRaWAN sont considérés comme des réseaux à très longue portée et à très faible consommation. Tous ces réseaux sont appelés LPWAN : **Low Power Wide Area Network**.

Dans ce cours, nous nous concentrerons uniquement sur **LoRaWAN (Long Range Wide Area Network)** qui est un protocole à longue portée, à faible débit et à très faible consommation.

1.1.3 Les bandes de fréquence

En Europe, certaines bandes de fréquence sont libres d'utilisation. Cela signifie :

- Pas de demande d'autorisation
- Gratuité d'utilisation

Le Tableau 1 présente quelques-unes de ces bandes libres en Europe.

Bande	Quelques protocoles
13,56 MHz	RFID, NFC
433 MHz	Talkie-walkie, télécommande, LoRa
868 MHz	Sigfox, LoRa, LoRaWAN
2,4 GHz	WiFi, Bluetooth, Zigbee, LoRa
5 GHz	WiFi

Tableau 1 : Bandes de fréquences libre d'utilisation

On constate qu'en Europe, le LoRa peut utiliser la bande 433 MHz, 868 MHz ou 2,4GHz. En revanche seule la bande 868 MHz est utilisée pour LoRaWAN.

1.2 Les modes de partage du support

Quel que soit le protocole utilisé, le support de transfert de l'information est l'air, car tous les protocoles de l'IoT sont sans fil. Le support doit être partagé entre tous les utilisateurs de telle façon que chaque dispositif ne perturbe pas les autres. Pour cela une bande de fréquence est allouée. Par exemple pour la radio FM (Modulation de Fréquence) en Europe, la bande de fréquence va de 87,5 MHz à 108 MHz.

1.2.1 Le FDM (Frequency Division Multiplexing) :

Les Devices utilisent des canaux de fréquence pour séparer leurs transmissions. Le **LoRa utilise ce mode de partage**, c'est-à-dire que la bande libre de 868 MHz est divisée en plusieurs canaux qui peuvent être utilisés pour transmettre de l'information.

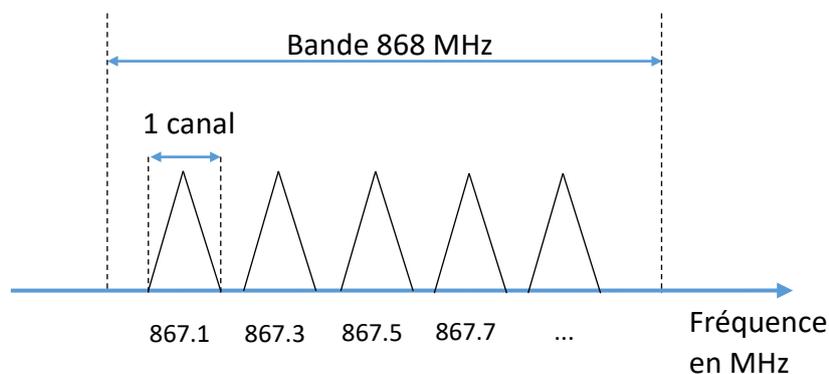


Figure 3: Utilisation du FDM par le LoRa

1.2.2 Le TDM (Time Division Multiplexing).

Dans ce mode de transmission, les dispositifs émettent par intermittence afin de laisser le canal libre pour les autres utilisateurs. Le **LoRa utilise ce mode de partage**, c'est-à-dire que LoRa ne permet pas une transmission continue et il utilisera la bande seulement pendant un temps donné. Cependant, les Devices ne sont pas synchronisés entre eux et donc aucun ne sait si un autre Device est déjà en train de transmettre. Cette caractéristique sera à l'origine d'éventuelles collisions.

1.2.3 L'Étalement de spectre

Dans ce mode de transmission, les Devices transmettent en **même temps, sur le même canal**, mais avec une structure de signal spécifique (avec des codes ou des symboles) qui permet au récepteur de retrouver le signal enfoui dans le bruit. Le **LoRa utilise ce mode de partage**.

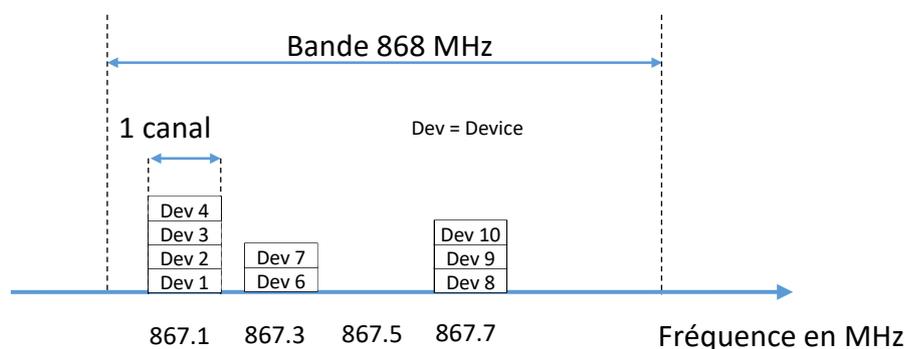


Figure 4: Utilisation de l'étalement de spectre dans le LoRa

Les Devices peuvent choisir entre plusieurs canaux pour transmettre mais l'avantage du LoRa est que même si plusieurs Devices choisissent le même canal, ils pourront transmettre en même temps sans que la communication soit perturbée.

Bien que les méthodes de modulation par étalement du spectre les plus connues utilisent des "codes" pour atteindre ce résultat, Le LoRa utilise à la place, des symboles (appelés Chirp), d'où son nom : "**Chirp Spread Spectrum (CSS) modulation**".

Afin de comprendre la pertinence de cette méthode de partage, nous allons valider le concept en utilisant des codes dans la section suivante, puis, dans le chapitre 3 nous expliquerons en détail la véritable modulation LoRa "Chirp Spread Spectrum".

1.3 L'étalement de spectre en utilisant des codes

Nous allons voir au travers d'un exemple concret comment il est possible pour trois Devices de transmettre en même temps sur le même canal. Nous nous concentrerons sur la validité de la communication : les éléments binaires reçus doivent correspondre à ceux transmis.

La méthode consiste à utiliser des codes qui ont des propriétés mathématiques adaptées à notre objectif : transmettre au même moment sur le même canal. La matrice ci-dessous donne quatre codes d'étalement (1 par ligne).

Code orthogonal Utilisateur 0	1	1	1	1
Code orthogonal Utilisateur 1	1	-1	1	-1
Code orthogonal Utilisateur 2	1	1	-1	-1
Code orthogonal Utilisateur 3	1	-1	-1	1

Tableau 2 : Matrice Hadamart (mathématicien) d'ordre 4

La propriété de ces codes (**1 1 1 1 ; 1 -1 1 -1 ; 1 1 -1 -1 ; 1 -1 -1 1**) est leur "orthogonalité" les uns par rapport aux autres. Nous ne l'expliquons pas ici mais il faut garder à l'esprit que cela ne fonctionne que si cette propriété "d'orthogonalité" est respectée.

1.3.1 La validation d'une transmission unique

Chaque tableau ci-dessous représente une transmission, qui est indépendante des autres. Dans cette première étape, nous ne nous soucions pas encore de savoir si elles ont lieu en même temps, mais nous vérifions simplement que chaque transmission arrive à destination correctement.

Chaque utilisateur possède son propre code orthogonal provenant du Tableau 2 et appelés "code d'étalement". Deux utilisateurs ne peuvent pas avoir le même code. La plupart des tableaux suivants sont déjà remplis à titre d'exemple. La méthode est la suivante :

Transmission :

Chaque bit du message est multiplié par son "code d'étalement" (1) x (2), et le résultat de cette multiplication est transmis. (3)

A la réception :

Chaque symbole reçu (4) est multiplié à nouveau par le même "code d'étalement" (2). Le message reçu (6) est égal à la somme des symboles décodés (5), divisée par le nombre de symboles (ici nous avons quatre symboles).

Chaque tableau ci-dessous représente un utilisateur.

1	Message Utilisateur 1 (bits)	1	0	1	0
2	Code d'étalement Utilisateur 1	1 -1 1 -1	1 -1 1 -1	1 -1 1 -1	1 -1 1 -1
3	Symboles transmis = (1) x (2)	1 -1 1 -1	0 0 0 0	1 -1 1 -1	0 0 0 0
... transmission ...					
4	Symboles reçus	1 -1 1 -1	0 0 0 0	1 -1 1 -1	0 0 0 0
5	Symboles décodés = (4) x (2)	1 1 1 1	0 0 0 0	1 1 1 1	0 0 0 0
6	Message reçu = $\sum (5) / \text{nbr_Symb}$	1	0	1	0

Tableau 3 : Transmission de l'utilisateur 1



Les dernières colonnes des tableaux suivants (Utilisateur 2 et Utilisateur 3) sont laissées en blanc afin que vous puissiez vérifier la démarche par vous-même.



Une explication étape par étape de cette méthode est présentée dans le [cours LoRaWAN en vidéo](#).

1'	Message Utilisateur 2 (bits)	0	1	0	1
2'	Code d'étalement Utilisateur 2	1 1 -1 -1	1 1 -1 -1	1 1 -1 -1	1 1 -1 -1
3'	Symboles transmis = (1') x (2')	0 0 0 0	1 1 -1 -1		
... transmission ...					
4'	Symboles reçus	0 0 0 0	1 1 -1 -1		
5'	Symboles décodés = (4') x (2')	0 0 0 0	1 1 1 1		
6'	Message reçu = $\sum (5') / \text{nbr_Symb}$	0	1		

Tableau 4 : Transmission de l'utilisateur 2

1''	Message Utilisateur 3 (bits)	1	1	0	0
2''	Code d'étalement Utilisateur 3	1 -1 -1 1	1 -1 -1 1	1 -1 -1 1	1 -1 -1 1
3''	Symboles transmis = (1'') x (2'')	1 -1 -1 1	1 -1 -1 1		
... transmission ...					
4''	Symboles reçus	1 -1 -1 1	1 -1 -1 1		
5''	Symboles décodés = (4'') x (2'')	1 1 1 1	1 1 1 1		
6''	Message reçu = $\sum (5'') / \text{nbr_Symb}$	1	1		

Tableau 5 : Transmission de l'utilisateur 3

1.3.2 Transmissions simultanées

Les transmissions ont maintenant lieu simultanément et les messages de l'utilisateur 1, de l'utilisateur 2 et de l'utilisateur 3 sont envoyés en même temps sur le même canal. Les deux premières colonnes sont déjà remplies à titre d'exemple. La méthode est la suivante :

Dans l'air :

- Le signal reçu est l'addition de tous les symboles transmis par tous les utilisateurs (1, 2, 3); nous ajoutons donc les lignes suivantes (3) + (3') + (3'') = (3''') :

3	Symboles transmis Utilisateur 1	1 -1 1 -1	0 0 0 0	1 -1 1 -1	0 0 0 0
3'	Symboles transmis Utilisateur 2	0 0 0 0	1 1 -1 -1		
3''	Symboles transmis Utilisateur 3	1 -1 -1 1	1 -1 -1 1		
3'''	\sum des symboles transmis (3 + 3' + 3'')	2 -2 0 0	2 0 -2 0		

Tableau 6 : Transmission simultanée des utilisateurs 1, 2 et 3.

Ensuite, sur chaque récepteur :

- Le signal reçu sur chaque récepteur (3''') est multiplié par le "code d'étalement" de chaque utilisateur (2, 2' ou 2''), le résultat est (7), (7') et (7'').
- Chaque message (6, 6', 6'') est égal à la somme des symboles décodés (7, 7', 7''), divisée par le nombre de symboles (ici nous avons quatre symboles).

3'''	Symboles reçus (3 + 3' + 3'')	2 -2 0 0	2 0 -2 0		
2	Code d'étalement Utilisateur 1	1 -1 1 -1	1 -1 1 -1	1 -1 1 -1	1 -1 1 -1
7	Symboles décodés (3''') x (2)	2 2 0 0	2 0 -2 0		
6	Message reçu Utilisateur 1 = $\sum (7) / \text{nbr_Symb}$	1	0		
2'	Code d'étalement Utilisateur 2	1 1 -1 -1	1 1 -1 -1	1 1 -1 -1	1 1 -1 -1
7'	Symboles décodés (3''') x (2')	2 -2 0 0	2 0 2 0		
6'	Message reçu Utilisateur 2 = $\sum (7') / \text{nbr_Symb}$	0	1		
2''	Code d'étalement Utilisateur 3	1 -1 -1 1	1 -1 -1 1	1 -1 -1 1	1 -1 -1 1
7''	Symboles décodés (3''') x (2'')	2 2 0 0	2 0 2 0		
6''	Message reçu Utilisateur 3 = $\sum (7'') / \text{nbr_Symb}$	1	1		

Tableau 7 : Réception des messages des utilisateurs 1, 2, 3.



On note que les messages (6, 6', 6'') sont équivalents à ceux envoyés (1, 1', 1'').

1.3.3 Le protocole LoRa

Le protocole LoRa utilise une méthode d'étalement de spectre différente de celle étudiée dans le paragraphe précédent. Cependant, l'objectif est le même : pouvoir émettre **au même moment, sur**

le même canal. Par exemple, Le transceiver LoRa SX1261 peut utiliser huit "codes d'étalement" appelés "Spreading Factor" [SF5, SF6, SF7, SF8, SF9, SF10, SF11 et SF12]. On peut donc avoir huit transmissions simultanées sur le même canal. Le paramètre SF (**S**Spreading **F**actor) sera expliqué en détails dans le chapitre 3. Dans le protocole LoRaWAN, nous utilisons seulement six SF [de SF7 à SF12].

2 Transmission radio et propagation

2.1 Unités et définitions

2.1.1 Le décibel (dB)

Lorsqu'un signal se propage, le rapport entre la puissance reçue et la puissance émise peut être très différent suivant les situations. Alors que ce rapport est presque égal à 1 si on utilise du cuivre, il peut être très élevé pour un amplificateur ou à l'inverse, extrêmement faible lorsque nous transmettons dans l'air. Dans ce dernier cas, la puissance reçue peut être aussi faible que quelques milliardièmes de milliardièmes de la puissance transmise. Traiter des nombres aussi grands et aussi petits ne permet pas de caractériser rapidement une amplification ou une atténuation. De plus, les opérations sur les gains (multiplication) et les atténuations (division) ne simplifient pas l'utilisation de ces valeurs.

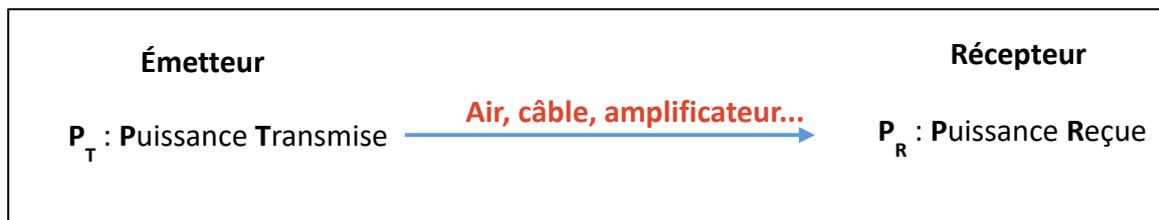


Figure 5 : La transmission de puissance entre l'émetteur et le récepteur

Le **dB** est le rapport entre deux puissances : la puissance du récepteur P_R et la puissance de l'émetteur P_T . Pour mémoire, la formule pour le rapport en dB est :

$$\text{Power ratio (dB)} = 10 \cdot \log_{10} \left(\frac{P_R}{P_T} \right)$$

Si le résultat est un nombre négatif (-), il s'agit d'une atténuation. Si le résultat est un nombre positif (+), il s'agit d'une amplification.

Si vous avez la puissance en dB et que vous souhaitez l'exprimer en rapport, alors la formule est la suivante :

$$\frac{P_R}{P_T} = 10^{\frac{\text{Power ratio (dB)}}{10}}$$

Si le résultat est inférieur à 1, il s'agit d'une atténuation. Si le résultat est supérieur à 1, il s'agit d'une amplification.

Avec ces deux formules, vous pouvez facilement vérifier les valeurs du tableau suivant.

Rapport de puissance en dB	Rapport de puissance
+ 10 dB	Multiplication par 10
+ 3 dB	Multiplication par environ 2
0 dB	Égalité
-3 dB	Division par 2 environ
- 10 dB	Division par 10

Tableau 8 : Calcul du rapport de puissance

L'intérêt de l'utilisation du ratio en dB est que non seulement nous gérons maintenant des nombres raisonnablement grands, mais nous n'utilisons plus que l'opération + et - pour le calcul global.



Exercice :

Sur la Figure 5, le câble qui transmet le signal a une amplification de -6 dB (ou une atténuation de 6 dB). Quel est le rapport de puissance entre P_R et P_T ?

Réponse :

$$\frac{P_R}{P_T} = 10^{\frac{-6}{10}} = 0.25$$



- -6 dB est négatif, il s'agit bien d'une atténuation.
- 0.25 est inférieur à 1, il s'agit bien d'une atténuation.

2.1.2 Puissance en dBm

Le **dBm** est la puissance par rapport à 1 mW : 0 dBm correspond à 1 mW. En utilisant les mêmes ratios que dans Tableau 8, nous pouvons remplir le Tableau 9 en ce qui concerne la puissance en dBm.

Puissance en dBm	Puissance en mW
10 dBm	10 mW
+ 3 dBm	2 mW
0 dBm	1 mW
- 3 dBm	0,5 mW
- 10 dBm	0,1 mW

Tableau 9 : Comparaison de la puissance en dB et mW

Exercice : Le talkie-walkie a une puissance d'émission de 2 W. A l'aide du Tableau 9, trouvez la puissance d'émission en dBm.



Réponse :

$$1 \text{ mW} \times 10 \times 10 \times 10 \times 2 = \mathbf{2 \text{ W}}$$

$$0 \text{ dBm} + 10 + 10 + 10 + 3 = \mathbf{33 \text{ dBm}}$$

Le talkie-walkie a une puissance de transmission de 33 dBm.

Pour mémoire, les formules pour retrouver la puissance en dBm ou la puissance en Watt sont :

$$Power \text{ (dBm)} = 10 \cdot \log_{10} \left(\frac{Power \text{ (Watt)}}{0,001} \right)$$

$$Power \text{ (Watt)} = 0.001 \times 10^{\frac{Power \text{ (dBm)}}{10}}$$

2.1.3 RSSI, Sensibilité, SNR, Bilan de liaison

Un émetteur transmet un signal avec une puissance P_T . Le récepteur récupère une fraction de cette puissance (P_R), ainsi que du bruit (P_B).

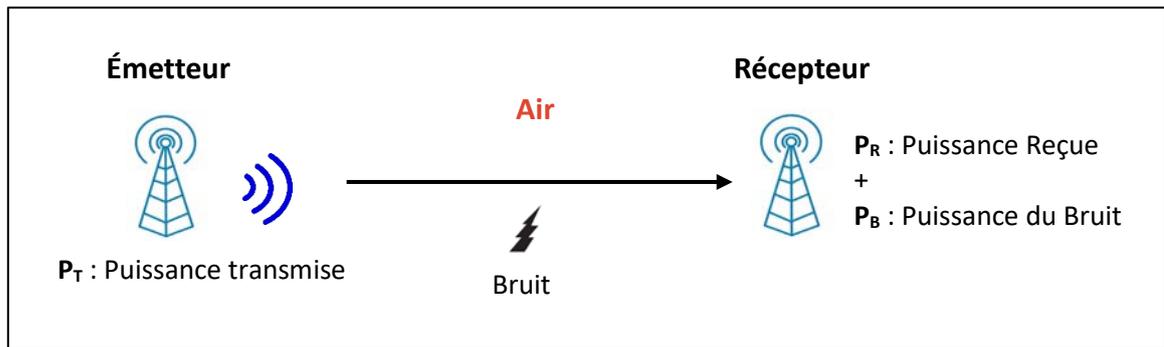


Figure 6: Bilan d'une transmission radio

- Le RSSI (**R**eceived **S**ignal **S**trength **I**ndication) est la puissance reçue P_R .
- La **sensibilité** est la puissance P_R minimale (ou RSSI minimal) qui doit être présente au niveau du récepteur afin de pouvoir réceptionner le signal. Si le RSSI reçu est inférieur à la sensibilité, alors le signal est indétectable.
- Le **SNR** (**S**ignal over **N**oise **R**atio) est le rapport entre la puissance reçue (P_R) et la puissance du bruit (P_B).

Toutes ces valeurs (RSSI, Sensibilité, SNR, ...) sont données en décibel. Un signal peut être correctement reçu si les deux conditions suivantes sont remplies :

1. Le RSSI est supérieur à la sensibilité du récepteur.
2. Le rapport signal/bruit (SNR) ne descend pas en dessous d'un certain seuil qui rendrait le signal impossible à détecter du côté du récepteur.



Exercice : Un émetteur fournit une puissance de 13 dBm en utilisant une antenne avec un gain de 2 dB. La perte dans l'air est de 60 dB. L'antenne de réception a un gain de 2 dB et est connectée à un récepteur avec une sensibilité de -80 dBm. Le signal sera-t-il reçu ?

Réponse :

$$13 + 2 - 60 + 2 = -43$$

-43 dB est supérieur à -80 dB (sensibilité)

La puissance reçue est de -43 dBm

Oui, le signal peut être reçu

Les logs ci-dessous proviennent d'une Gateway LoRaWAN. Ils donnent un exemple des valeurs RSSI et SNR mesurés lors d'une transmission de données. Les valeurs "rssi" : -13 et "snr" : 9,5 montrent que dans cet exemple, le signal reçu a une puissance élevée et a un très bon SNR. En effet, le Device LoRaWAN n'était qu'à quelques mètres de la Gateway pendant ce test.

```
"Gateways" :
  {
    "time" : "2020-04-29T12:09:45.563621044Z",
    "channel" : 0,
    "rssi" : -13,
    "snr" : 9.5
  }
```

Que pouvons-nous faire si la puissance reçue (P_R) est inférieure à la sensibilité ? La première idée serait d'augmenter P_T . C'est possible dans une certaine mesure car la puissance de transmission est limitée par la réglementation. La puissance maximale P_T sur la bande 868 MHz est de 14 dBm (25 mW). La deuxième possibilité est d'améliorer la sensibilité du récepteur. C'est à l'évidence ce sur

quoi travaillent les concepteurs de modules LoRa. Au final, c'est la différence entre la puissance émise P_T et la sensibilité du récepteur qui importe. C'est ce qu'on appelle le bilan de liaison. Dans l'exercice précédent, le bilan de liaison est de 93 dB (13 + 80).



- ➔ En LoRa, nous avons un bilan de liaison d'environ 157 dB.
- ➔ En LTE (4G), le bilan de liaison est d'environ 130 dB.

2.2 Distance de transmission en LoRa

La puissance transmise (P_T) est atténuée dans l'air selon la formule simplifiée suivante :

$$Loss = 10 \cdot \log_{10}(Distance^2 \cdot Frequency^2 \cdot 1755)$$

- Perte : en **dB**.
- Distance : en **km**.
- Fréquence : en **MHz**.

On peut aussi estimer la distance maximale par la formule suivante :

$$Distance = \sqrt{\frac{10^{\frac{Loss}{10}}}{1755 \cdot Frequency^2}}$$

Le bilan de liaison correspond aux pertes maximales qu'une transmission peut supporter et nous allons estimer grossièrement ici que toutes ces pertes se font dans l'air.

$$distance = \sqrt{\frac{10^{\frac{Link\ Budget}{10}}}{1755 \cdot frequency^2}}$$

- L'émetteur-récepteur LoRa SX1272 (Link Budget de 157 dB) donne une distance théorique de 1946 km.
- L'émetteur-récepteur LoRa SX1262 (Link Budget de 170 dB) donne une distance théorique de 8696 km.

En avril 2020, le record du monde pour une transmission LoRa a été battu. On a atteint 832 km dans la bande EU868 en utilisant une puissance de 25 mW / 14 dBm (puissance maximale autorisée).

2.3 Documentation d'un Transceiver LoRa

La Figure 7 est un extrait de la documentation du transceiver SX1262.

Features

- LoRa and FSK Modem
- 170 dB maximum link budget (SX1262 / 68)
- +22 dBm or +15 dBm high efficiency PA
- Low RX current of 4.6 mA
- Integrated DC-DC converter and LDO
- Programmable bit rate up to 62.5 kbps LoRa and 300 kbps FSK
- High sensitivity: down to -148 dBm

Figure 7: Principales caractéristiques du Transceiver LoRa SX1262

En utilisant la définition du bilan de liaison (P_T moins la sensibilité du récepteur), nous trouvons les 170 dB indiqués dans cette documentation (22 dBm + 148 dBm). Néanmoins, nous rappelons que la puissance d'émission maximale en Europe est de 14 dBm, ce qui réduit le bilan de liaison à 162 dB (14 dBm + 148 dBm).

En LoRa, plus le Spreading Factor est grand, plus nous sommes capables de transmettre dans un environnement perturbé. Le Tableau 10 montre les rapports signal/bruit (SNR) avec lesquels nous pourrions effectuer une transmission, en fonction du Spreading Factor utilisé.

SpreadingFactor (RegModemConfig2)	Spreading Factor (Chips / symbol)	LoRa Demodulator SNR
6	64	-5 dB
7	128	-7.5 dB
8	256	-10 dB
9	512	-12.5 dB
10	1024	-15 dB
11	2048	-17.5 dB
12	4096	-20 dB

Tableau 10 : Influence du Spreading Factor sur le SNR

On remarque que pour un SF8, on est capable de recevoir un signal avec un SNR de -10 dB : on pourra donc recevoir un signal noyé dans du bruit dont la puissance est 10 fois supérieure au signal.

On remarque que pour un SF12, on est capable de recevoir un signal avec un SNR de -20 dB : on pourra donc recevoir un signal noyé dans du bruit dont la puissance est 100 fois supérieure au signal.

Cependant, nous remarquons également qu'en utilisant un Spreading Factor plus élevé, le nombre de "chips" transmis augmente (2^{ème} colonne du tableau). Nous verrons plus tard ce que cela signifie exactement, mais nous pouvons d'ores et déjà dire que cela affectera de toute évidence le temps de transmission et donc le débit binaire.

3 La modulation LoRa (couche physique)

3.1 La modulation LoRa

Comme nous l'avons expliqué ci-dessus, la modulation LoRa utilise l'étalement de spectre pour transmettre ses informations. Mais au lieu d'utiliser des codes d'étalement (CDMA), elle utilise une méthode appelée Chirp Spread Spectrum. La finalité est toujours la même : avoir plusieurs transmissions dans le même canal. La conséquence sur le spectre est aussi la même : cela provoque un étalement du spectre.

3.1.1 Le Chirp

Le signal émis par la modulation LoRa est un symbole dont la forme de base est représentée ci-dessous. Son nom (Chirp) vient du fait que ce symbole est utilisé dans la technologie Radar (**Chirp** : Compressed **H**igh **I**ntensity **R**adar **P**ulse).

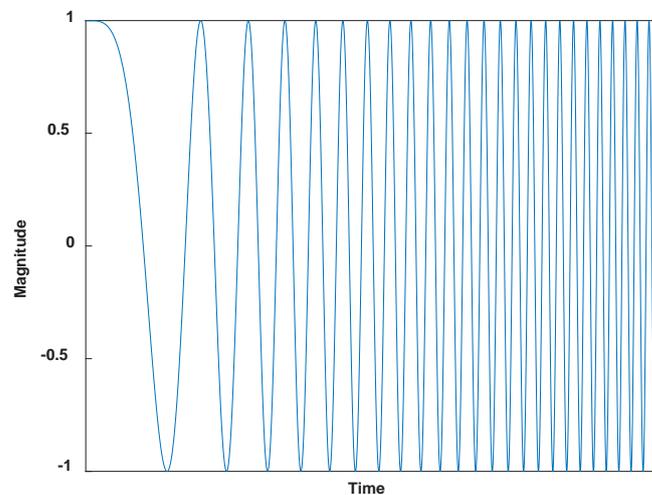


Figure 8: Un Chirp (simulation Matlab)

La fréquence de départ est la fréquence centrale du canal **moins** la Bande Passante divisée par deux. La fréquence de fin est la fréquence centrale du canal **plus** la Bande Passante divisée par deux. La Figure 9 représente un Chirp LoRa dans le domaine fréquentiel.

- Le canal ($F_{channel}$) est la fréquence centrale.
- La bande occupée autour de $F_{channel}$ est la bande passante (BW).

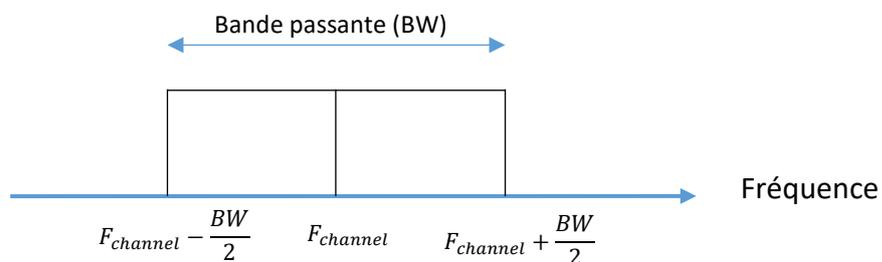


Figure 9: Occupation spectrale d'une transmission LoRa



Exercice : Une transmission LoRa utilise le canal 868,1 MHz avec une bande passante de 125 kHz. Donner la fréquence de début et de fin du Chirp.

Réponse :

- Fréquence de départ : $868\,037\,500\text{ Hz} = 868\,100\,000 - 125\,000/2$
- Fréquence finale : $868\,162\,500\text{ Hz} = 868\,100\,000 + 125\,000/2$

Pour faciliter la représentation de ce symbole, nous utilisons un graphique Temps-Fréquence (Spectrogramme).

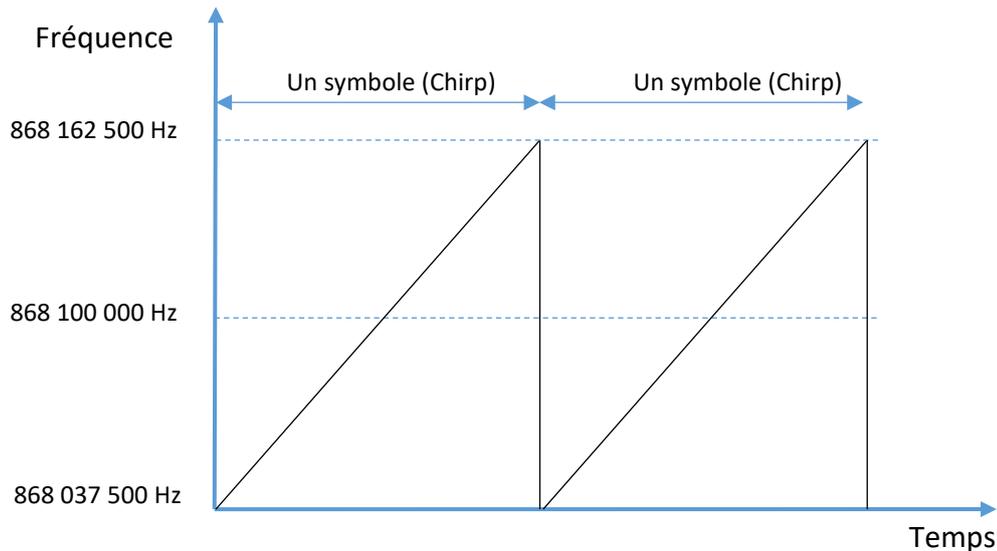


Figure 10: Spectrogramme d'une transmission LoRa

En LoRa, chaque symbole représente un nombre de bits transmis. La règle est la suivante :

Nombre de bits transmis dans un symbole = Spreading Factor

Par exemple, si la transmission utilise un Spreading Factor de 10 (SF10), alors un symbole (Chirp) représente 10 bits.

Lors de l'émission, les bits sont regroupés en paquets de **SF** bits. Ensuite, chaque paquet est représenté par un symbole particulier parmi les 2^{SF} formes possibles. Entre tous ces symboles, la seule différence est qu'ils partent tous d'une fréquence particulière, désignant ainsi le paquet de bits.

La Figure 11 montre un exemple théorique de modulation SF2 à 868,1 MHz, avec une bande passante de 125 kHz. Chaque symbole représente 2 bits.

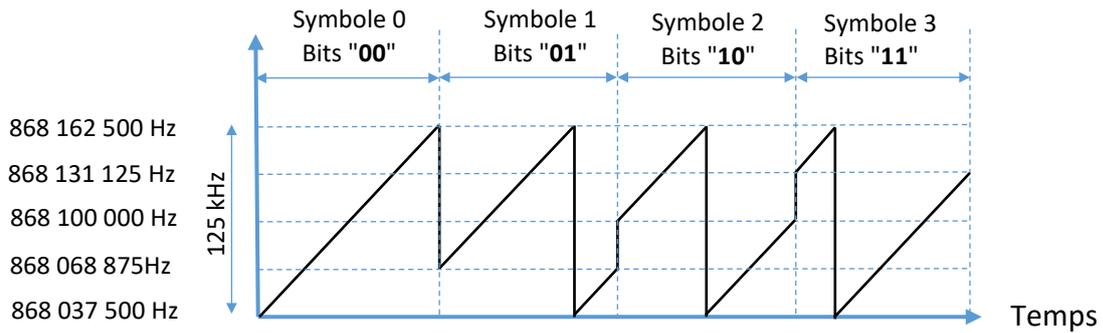


Figure 11: Symboles transmis en modulation LoRa en SF2 (cas théorique)

Exemple :

- On considère la séquence binaire suivante : 0 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 1 0 1
- Nous utilisons SF10

On regroupe les bits par paquets de 10. Chaque paquet de 10 bits est représenté par un symbole particulier. Il existe 1024 symboles différents pour coder les 1024 combinaisons binaires possibles (2^{10}).

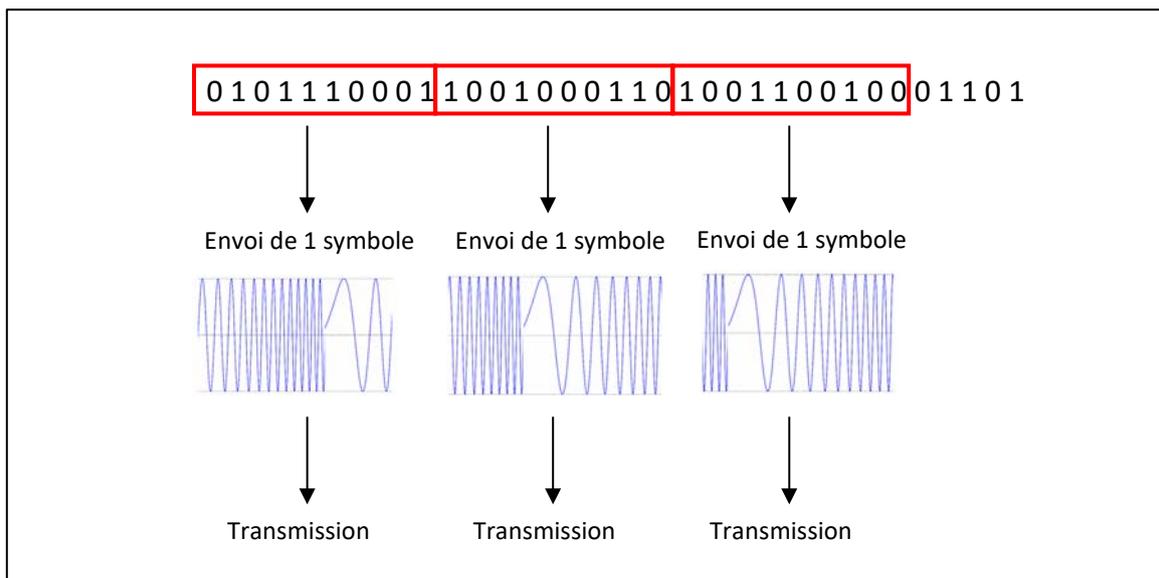


Figure 12: Emission des Chirps en LoRa

La Figure 13 est le spectrogramme d'une transmission LoRa mesuré grâce à un récepteur ADALM Pluto.



Figure 13: Spectrogram d'une transmission LoRa

3.1.2 Durée d'émission d'un symbole

En LoRa, le temps d'émission de chaque symbole (T_{symbole}) dépend du **Spreading Factor** utilisé. Plus le SF est élevé, plus le temps d'émission sera long. Pour une même bande passante, le temps de transmission d'un symbole en SF8 est deux fois plus long que le temps de transmission d'un symbole en SF7. Ainsi de suite jusqu'à SF12.

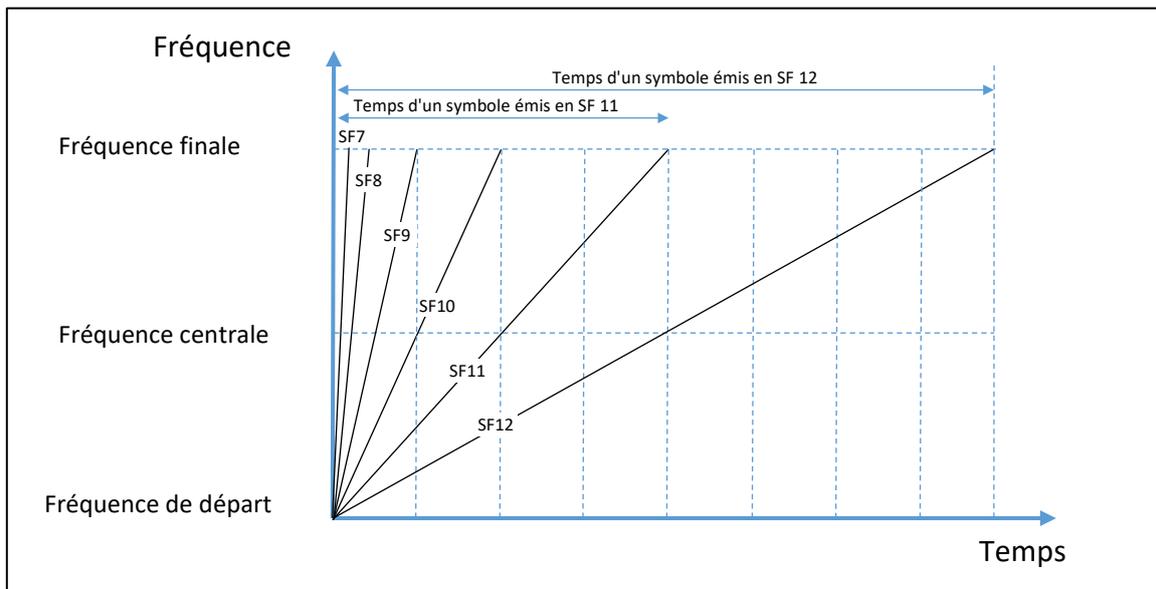


Figure 14: Temps d'émission d'un symbole

Le temps d'émission de chaque symbole (T_{symbole}) est aussi inversement proportionnel à la bande passante. Si nous prenons maintenant en compte le **SF** et la **bande passante**, nous obtenons la formule suivante :

$$T_{\text{symbole}} = \frac{2^{SF}}{\text{Bandwidth}}$$

A titre d'exemple, le Tableau 11 montre le temps d'émission T_{symbole} en fonction du Spreading Factor pour une bande passante de 125 KHz.

Spreading Factor	Temps d'émission d'un symbole
SF7	1,024 ms
SF8	2,048 ms
SF9	4,096 ms
SF10	8,192 ms
SF11	16,384 ms
SF12	32,768 ms

Tableau 11 : Temps d'émission d'un symbole à 125 kHz

Le débit de symboles est $\frac{1}{T_{symbol}} = F_{symbol} = \frac{Bandwidth}{2^{SF}}$. Évidemment, plus la bande passante est élevée, plus le débit de symboles est élevé.

3.1.3 Le Time on Air

Le "Time on Air" est la durée globale d'une trame LoRa. Il dépend du nombre de symboles présents dans la trame en prenant en compte le préambule, l'entête, le payload et le CRC.

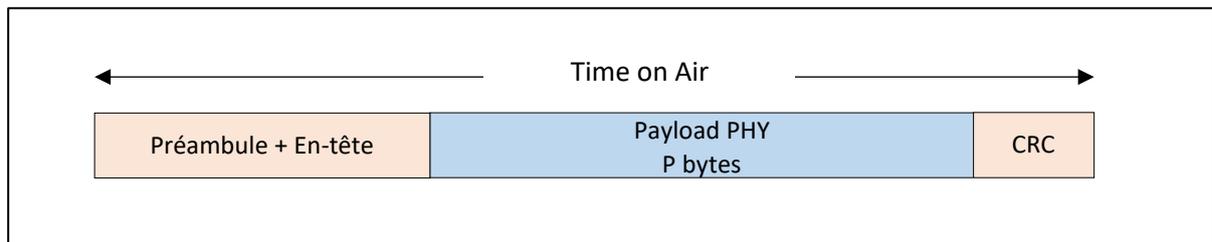


Figure 15: Time on Air d'une trame LoRa

$Time\ on\ Air = n_{symbole} \cdot T_{symbole}$ où $n_{symbole}$ est le nombre de symboles présents dans la trame LoRa.

Nous verrons plus tard dans le cours une façon de calculer $n_{symbole}$ afin de trouver le Time on Air d'une transmission LoRa.

3.2 Débit binaire en LoRa et LoRaWAN

3.2.1 Débit binaire en LoRa

Comme chaque symbole est composé de SF bits, le débit binaire est de :

$$Bit\ Rate = SF \cdot \frac{Bandwidth}{2^{SF}}$$



- ➔ Plus le Spreading Factor est élevé, plus le débit binaire est faible.
- ➔ Plus la bande passante est élevée, plus le débit binaire est élevé.



Exercice : On considère les deux cas suivants : cas 1 (SF7, 125 kHz) et cas 2 (SF12, 125 kHz). Donnez le débit binaire correspondant.

Réponse :

- Cas 1 : Pour SF7, 125 kHz > Débit binaire = **6 836 bps**
- Cas 2 : Pour SF12, 125 kHz > Débit binaire = **366 bps**

3.2.2 Influence du Coding Rate sur le débit

Le Coding Rate est un ratio qui augmente le nombre de bits à transmettre afin de réaliser de la détection / correction d'erreurs. Dans le cas d'un CR = 4 / 8, il y aura 8 bits transmis réellement à chaque fois que nous souhaitons transmettre 4 bits. Dans cet exemple, cela provoque un doublement du nombre de bits envoyés.

CodingRate (RegModemConfig1)	Cyclic Coding Rate	Overhead Ratio
1	4/5	1.25
2	4/6	1.5
3	4/7	1.75
4	4/8	2

Tableau 12: Influence du Coding Rate sur le nombre de bits transmis



Exercice : En reprenant les deux cas précédents (SF7, 125 kHz) et (SF12, 125 kHz) avec un CR de 4/5, donnez le débit binaire correspondant.

Réponse :

- Cas 1 : Pour SF7, 125 kHz et CR4/5 > Débit binaire = 6.836 kbps / 1.25 = **5469 bps**
- Cas 2 : Pour SF12, 125 kHz et CR4/5 > Débit binaire = 366 bps / 1,25 = **293 bps**



La documentation d'un Transceiver LoRa donne les débits en fonction du Spreading Factor, de la bande passante et du Coding Rate. Nous pouvons vérifier la cohérence du résultat avec votre calcul précédent : le cas 2 a bien un débit binaire de 293 bps.

Bandwidth (kHz)	Spreading Factor	Coding rate	Nominal Rb (bps)	Sensitivity (dBm)
125	12	4/5	293	-136

Tableau 13: Débit binaire en fonction des paramètres de transmission LoRa

3.2.3 Simulation du débit binaire avec le LoRa Modem Calculator

Le LoRa Modem Calculator est un petit logiciel fourni par Semtech permettant de simuler une transmission LoRa en fonction des paramètres configurables : Canal, SF, CR, etc...

- Le LoRa Modem Calculator pour transceiver SX1272 est disponible [ici](#).
- Le LoRa Modem Calculator pour le transceiver SX1261 / SX1262 est disponible [ici](#).
- Un simulateur LoRa en ligne équivalent est également disponible [ici](#).

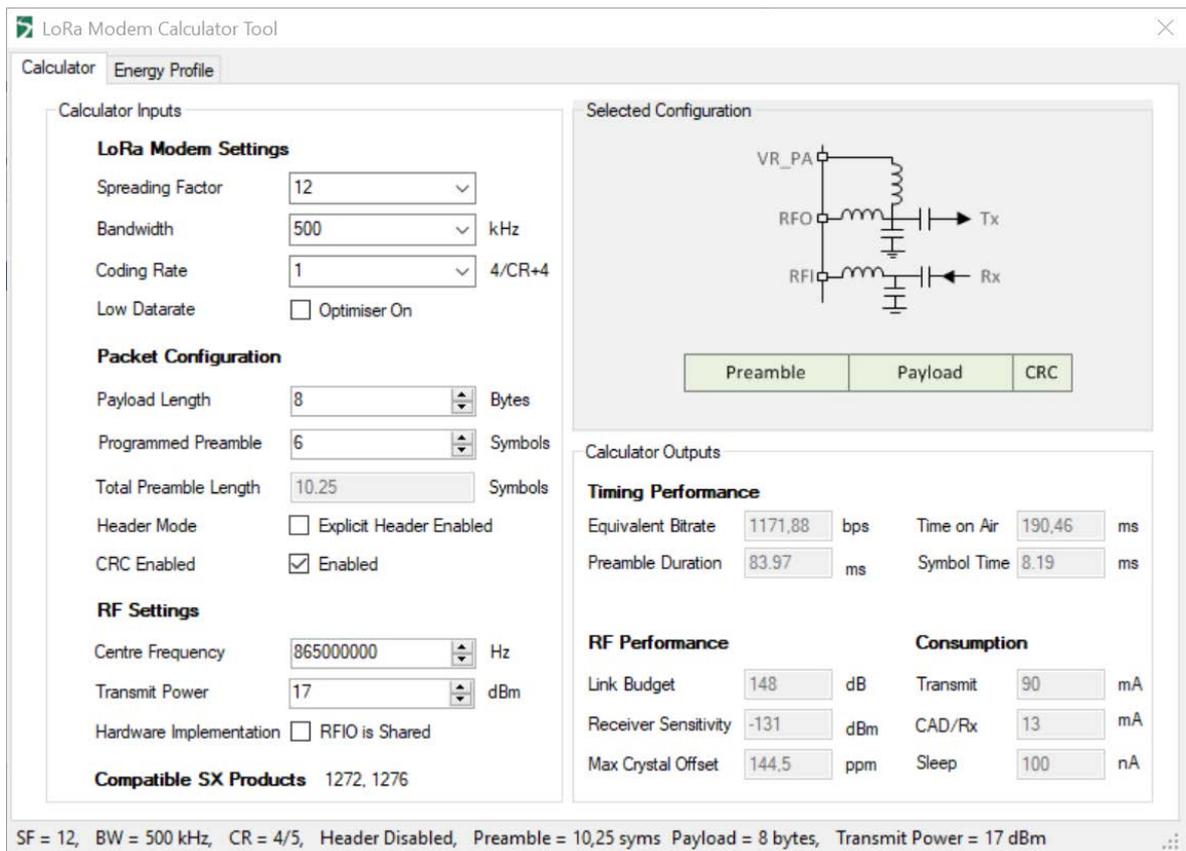


Figure 16: Le logiciel SX1272 LoRa Calculator

Exercice : En utilisant l'exemple des deux cas précédents [SF7, 125 kHz, CR 4/5] et [SF12, 125 kHz, CR 4/5], vérifiez les calculs "Equivalent Bitrate" avec le LoRa Modem Calculator.



Réponse :

- **Cas 1** : pour SF7, 125 kHz et CR4/5>. Débit binaire = **5468,75 bps**
- **Cas 2** : Pour SF12, 125 kHz et CR4/5 > Débit binaire = **292,97 bps**

3.2.4 Influence de l'entête LoRa sur le débit

Nous nous sommes focalisés pour l'instant uniquement sur le débit instantané. En réalité, il n'y a pas seulement les données qui sont envoyées lors d'une transmission LoRa, mais il faut transmettre en plus :

- Un préambule permettant au récepteur de se synchroniser.
- Un entête.
- Un CRC (vérification de l'intégrité de la trame).

Les données LoRa sont appelées **PHY Payload**.

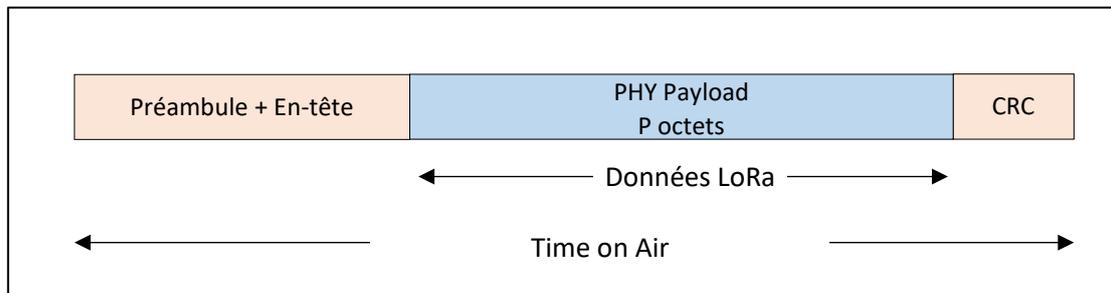


Figure 17: Trame LoRa

Nous nous intéressons maintenant au débit du PHY Payload : combien un utilisateur peut-il envoyer de bits de données utiles par seconde? Nous devons connaître le temps de transmission de la trame entière (Time on Air) et pour cela, nous utilisons le "LoRa Modem Calculator". La Figure 18 simule le Time on Air pour une transmission SF7, BW125, CR4/5 avec 1 octet comme PHY Payload.

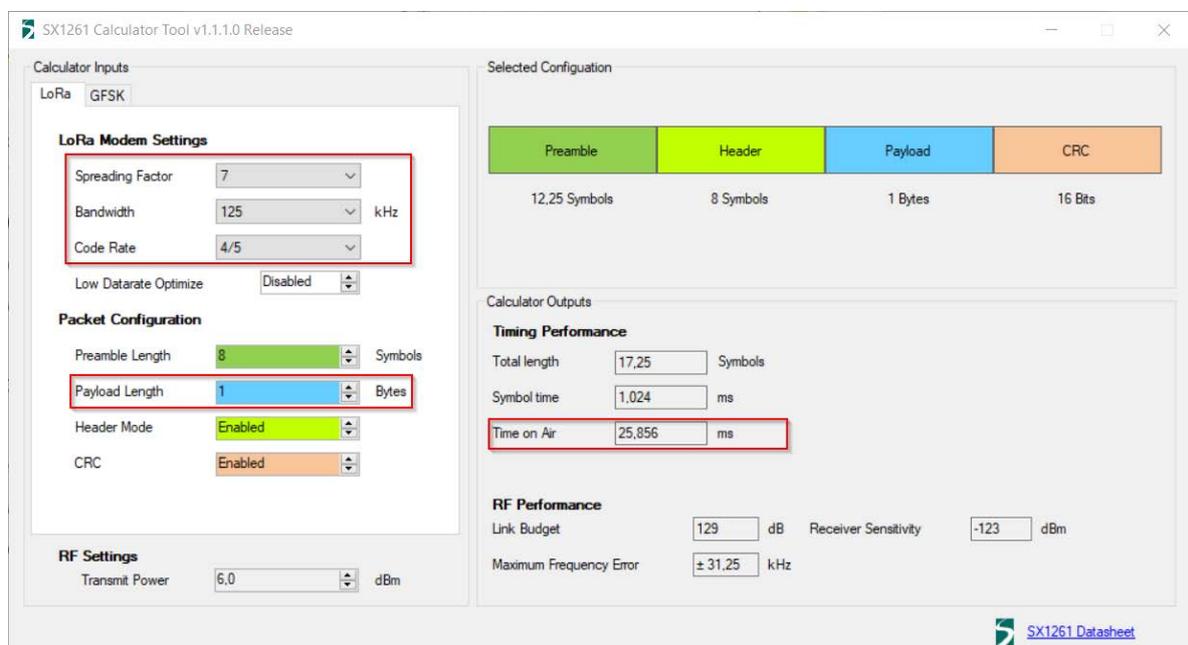


Figure 18: Simulation du Time on Air

Exercice : Vérifier le Time on Air des deux cas précédents [SF7, 125 kHz, CR 4/5] et [SF12, 125 kHz, CR 4/5] avec 1 octet de PHY Payload. Déduisez le débit réel du PHY Payload dans les deux cas.



Réponse :

- L'envoi d'un octet (PHY Payload) en SF7 donne un Time on Air de **25.85 ms**.
 - L'envoi d'un octet (PHY Payload) en SF12 donne un Time on Air de **827.39 ms**.
-
- **Cas 1 :** Pour SF7, 125 kHz et CR4/5 > Débit binaire_{LoRa payload} = 8 / 25,85 ms = **309,3 bps**
 - **Cas 2 :** Pour SF12, 125 kHz et CR4/5 > Débit binaire_{LoRa payload} = 8 / 827.39 ms = **9.6 bps**

3.2.5 Influence de l'entête LoRaWAN sur le débit

Le protocole LoRaWAN doit fournir des informations supplémentaires. Nous verrons dans le paragraphe 4.1.4 les différences entre le protocole LoRa et LoRaWAN. Nous verrons également au paragraphe 6.1 le détail de chaque champ de la trame LoRaWAN. Pour l'instant, nous pouvons simplement affirmer que le protocole LoRaWAN fournit un service supplémentaire au protocole LoRa.

La Figure 19 représente une trame LoRaWAN simplifiée. Un entête LoRaWAN a été ajouté et sa transmission augmentera le Time on Air. On considère que les données de l'utilisateur sont toujours de 1 octet.

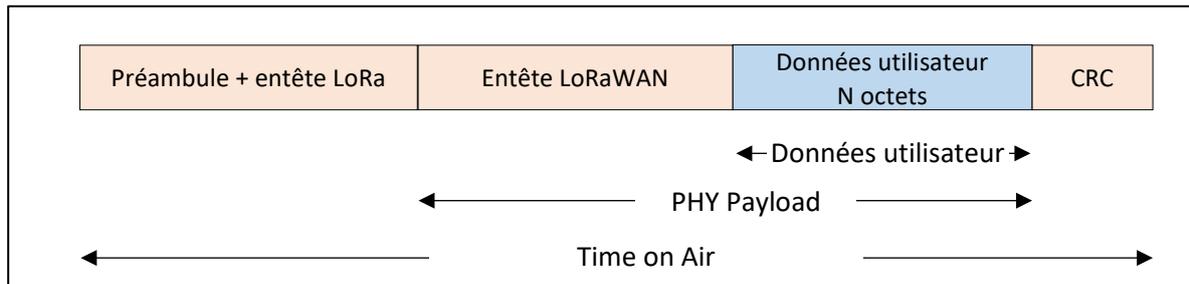


Figure 19: Trame LoRaWAN

Au niveau de l'utilisateur, seule compte la quantité de données utiles transmises. Par exemple, si l'utilisateur veut transmettre une température (1 octet), sa seule préoccupation est de savoir combien de données concernant la température il pourra transmettre par secondes / minutes / heures.

Nous pouvons simuler ce débit réel avec le LoRa Modem Calculator avec la configuration suivante :

- Spreading Factor : SF7
- Bande passante : 125 kHz
- Taux de codage : 4/5
- Taille du Payload : un entête LoRaWAN (habituellement de 13 octets) + les données de l'utilisateur (1 octet dans notre exemple). Le PHY Payload est donc de 14 octets.

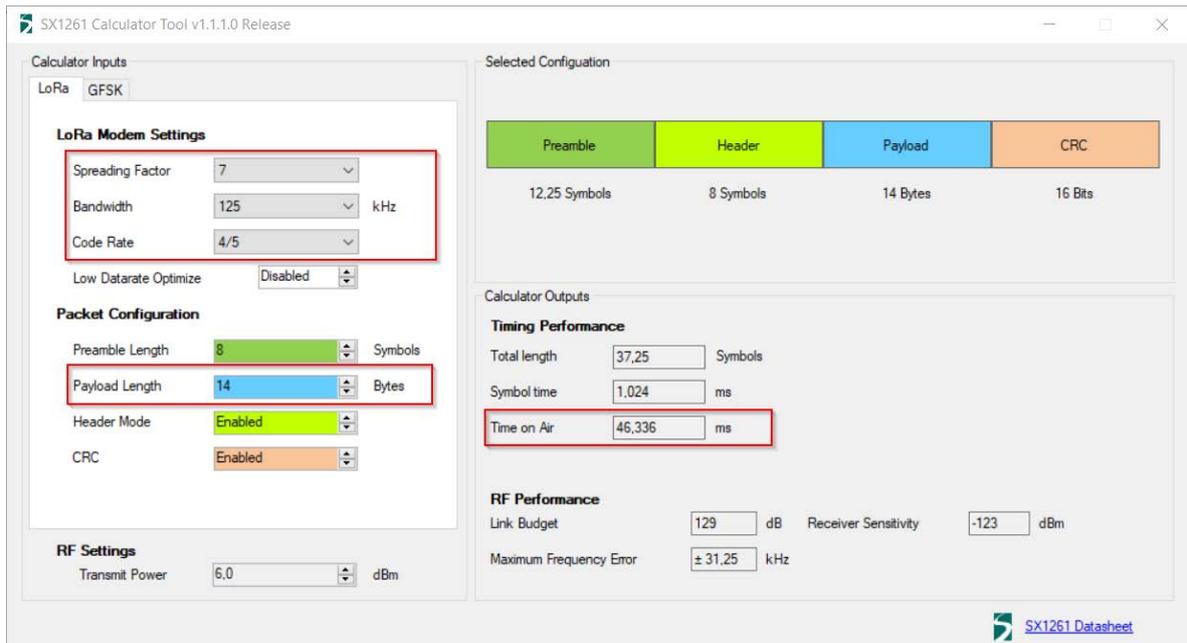


Figure 20: Time on Air d'une trame LoRaWAN

Le LoRa Calculator donne un Time on Air de **46,3 ms** pour SF7, 125 kHz et CR4/5.

Nous pouvons également vérifier ce Time on Air par une transmission réelle d'un Device LoRaWAN. Dans cette application, nous transmettons un seul octet (une température). La Gateway enregistre les informations de transmission de chaque paquet LoRa reçu. Nous collectons les valeurs suivantes sur la Gateway pour une transmission en SF7, puis en SF12. Nous nous intéressons à la colonne "airtime (ms)".

time	frequency	mod.	CR	data rate	airtime (ms)	cnt
▲ 09:54:22	868.5	lora	4/5	SF 7 BW 125	46.3	3783 dev addr: 26 01 16 8E payload size: 14 bytes
time	frequency	mod.	CR	data rate	airtime (ms)	cnt
▲ 11:07:48	868.3	lora	4/5	SF 12 BW 125	1155.1	1 dev addr: 26 01 16 8E payload size: 14 bytes

Figure 21: Time on Air pour un octet transmis en LoRaWAN en SF7 et SF12

On note que :

- L'envoi d'un octet (données utilisateur) en SF7 donne un Time on Air de **46,3 ms**.
- L'envoi d'un octet (données utilisateur) en SF12 donne un Time on Air de **1155.1 ms**.
- Le "payload size" indiquée (14 octets) est bien supérieure à 1 octet, ce qui montre qu'un entête supplémentaire (13 octets d'en-tête LoRaWAN) a été ajouté, ainsi que l'octet de température, soit 14 octets au total.



Exercice : Calculez le débit réel de cette transmission pour les deux cas [SF7, 125 kHz, CR 4/5] et [SF12, 125 kHz, CR 4/5].

Réponse :

- **Cas 1** : Pour SF7, 125 kHz et CR4/5 > Data Rate $_{\text{LoRaWAN Payload}} = 8 / 46.3 \text{ ms} = \mathbf{172.7 \text{ bps}}$
- **Cas 2** : Pour SF12, 125 kHz et CR4/5 > Data Rate $_{\text{LoRaWAN payload}} = 8 / 1155,1 \text{ ms} = \mathbf{6,9 \text{ bps}}$

3.2.6 Influence du Duty-cycle sur le débit (bande EU868)

La norme européenne exige qu'un dispositif radiofréquence n'émette pas plus de 1 % du temps dans la bande 868 MHz. C'est ce qu'on appelle le Duty Cycle. Par exemple, un Duty Cycle de 1% signifie que si un appareil émet pendant 1 (aucune unité), il doit rester silencieux pendant 99, quelle que soit l'unité de temps utilisée.

Exemple : Sur la Figure 21 en utilisant SF7, le Time on Air est de 46,3 ms. Le Device LoRa ne doit donc pas émettre pendant $99 \times 46,3 \text{ ms} = 4,58 \text{ secondes}$.

Exercice : En utilisant les exemples précédents [SF7, 125 kHz, CR 4/5] et [SF12, 125 kHz, CR 4/5], quel est le débit binaire moyen si l'on tient compte du Time on Air, et du duty-cycle de 1% de la norme LoRaWAN ?



Réponse :

- **Cas 1** : Pour SF7, 125 kHz et CR4/5 > Débit binaire_{LoRaWAN Payload 1%} = $172,7 \text{ bps} / 100 = \mathbf{1,73 \text{ bps}}$
- **Cas 2** : Pour SF12, 125 kHz et CR4/5 > Débit binaire_{LoRaWAN Payload 1%} = $6,9 \text{ bps} / 100 = \mathbf{0,07 \text{ bps}}$

Si nous lisons attentivement les spécifications, les choses sont en réalité un peu plus compliquées car le duty-cycle de 1% s'applique sur chaque bande :

- 863,0 - 868,0 MHz : 1 %.
- 868,0 - 868,6 MHz : 1 %.

Cela signifie que si un Device LoRa envoie une trame sur le canal 867,1 MHz (bande entre 863,0 - 868,0 MHz), ce Device est toujours autorisé à envoyer une autre trame sur le canal 868,1 MHz (bande entre 868,0 - 868,6 MHz).

3.2.7 Influence de la politique d'utilisation du serveur LoRaWAN

Le duty-cycle de 1% est un paramètre spécifique qui s'applique sur la bande européenne de 868 MHz (EU868). En outre, le serveur LoRaWAN peut limiter le nombre de messages que vous êtes autorisé à transmettre et cela dépend de votre abonnement.

Exemple : Sur la version communautaire de "The Things Network", il existe une limite qui empêche un Device de surcharger le réseau.

" Le Time on Air en uplink est limité à 30 secondes par jour (24 heures) et par Device. Les messages downlink sont limités à 10 messages par jour (24 heures) et par Device. Si vous utilisez un réseau privé, ces limites ne s'appliquent pas."

3.3 La simulation d'une transmission LoRa

3.3.1 Calcul du Time on Air

Le Time on Air dépend du nombre de symboles envoyés dans la trame LoRa, ainsi que de la durée d'un symbole.

$Time\ on\ Air = n_{symbole} \cdot T_{symbole}$ où

- $n_{symbole}$ est le nombre de symboles présents dans la trame LoRa.
- $T_{symbole} = \frac{2^{SF}}{Bandwidth}$ est le temps d'un symbole.

$n_{symbole}$ dépend de nombreux paramètres LoRa et peut être résumé à l'aide de la formule suivante fournie par SEMTECH :

$$n_{symbol} = (n_{preamble} + 4,25) + 8 + \max \left(\text{ceil} \left(\frac{8 \cdot \text{Payload} - 4 \cdot SF + 28 + 16 - 20 \cdot H}{4(SF - 2 \cdot DE)} \right) (CR + 4), 0 \right)$$

où

- Payload est le PHY Payload
- SF est le Spreading Factor
- H=0 lorsque l'en-tête est activé et H=1 si elle n'est pas activée.
- DE = 1 lorsque l'option "low data rate optimization" est activée, 0 sinon.
- CR est le Coding Rate de 1 à 4



Exercice : Vérifier la valeur du Time on Air trouvé dans la Figure 20 (46,3 ms)

Réponse : $n_{preamble} = 8$, Payload = 14, SF = 7, H = 0, DE = 0, CR = 1, donc

$$n_{symbol} = (8 + 4,25) + 8 + \text{ceil} \left(\frac{8 * 14 - 4 * 7 + 28 + 16}{4 * 7} \right) (1 + 4) = 45.25 \text{ symbols}$$

$$T_{symbol} = \frac{2^{SF}}{\text{Bandwidth}} = 1.024 \text{ ms}$$

$$\text{Time on Air} = 45.25 * 1.024 = 46.3 \text{ ms}$$



Nous avons donc bien retrouvé par le calcul le Time on Air simulé, ainsi que le Time on Air de la transmission réelle.

3.3.2 Modulation LoRa (Matlab)

La modulation LoRa génère un chirp nommé $\text{symbol}(t)$ mélangé à un oscillateur local nommé $\text{channel}(t)$ afin de décaler le chirp autour de la fréquence du canal (f_{channel}) et donc de générer le signal transmis $\text{lora}(t)$.

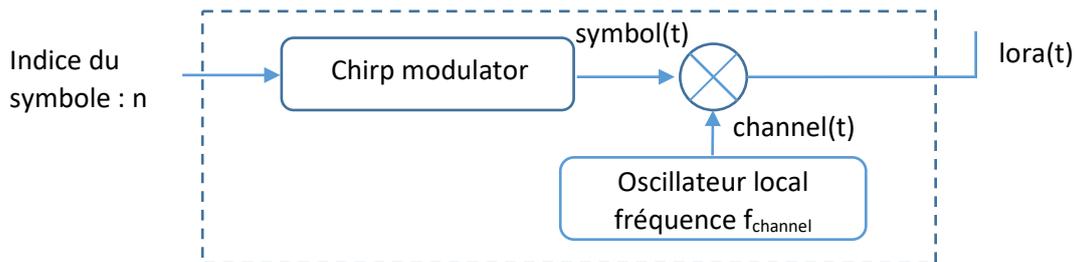


Figure 22: Schéma bloc de la modulation LoRa

Un tel modulateur peut être simulé à l'aide de Matlab pour une transmission en SF12 et une bande passante de 125 kHz. Dans ce cas, il y a $2^{12} = 4096$ symboles différents ($n \in [0; 4096]$) et chaque symbole représente 12 bits.

$$T_{symbole} = \frac{2^{SF}}{\text{Bandwidth}} = 32,768 \text{ ms}$$

La Figure 23 représente le spectrogramme de $\text{symbol}(t)$ pour quatre indices différents : $n=0, 1024, 2048$ et 3072 .

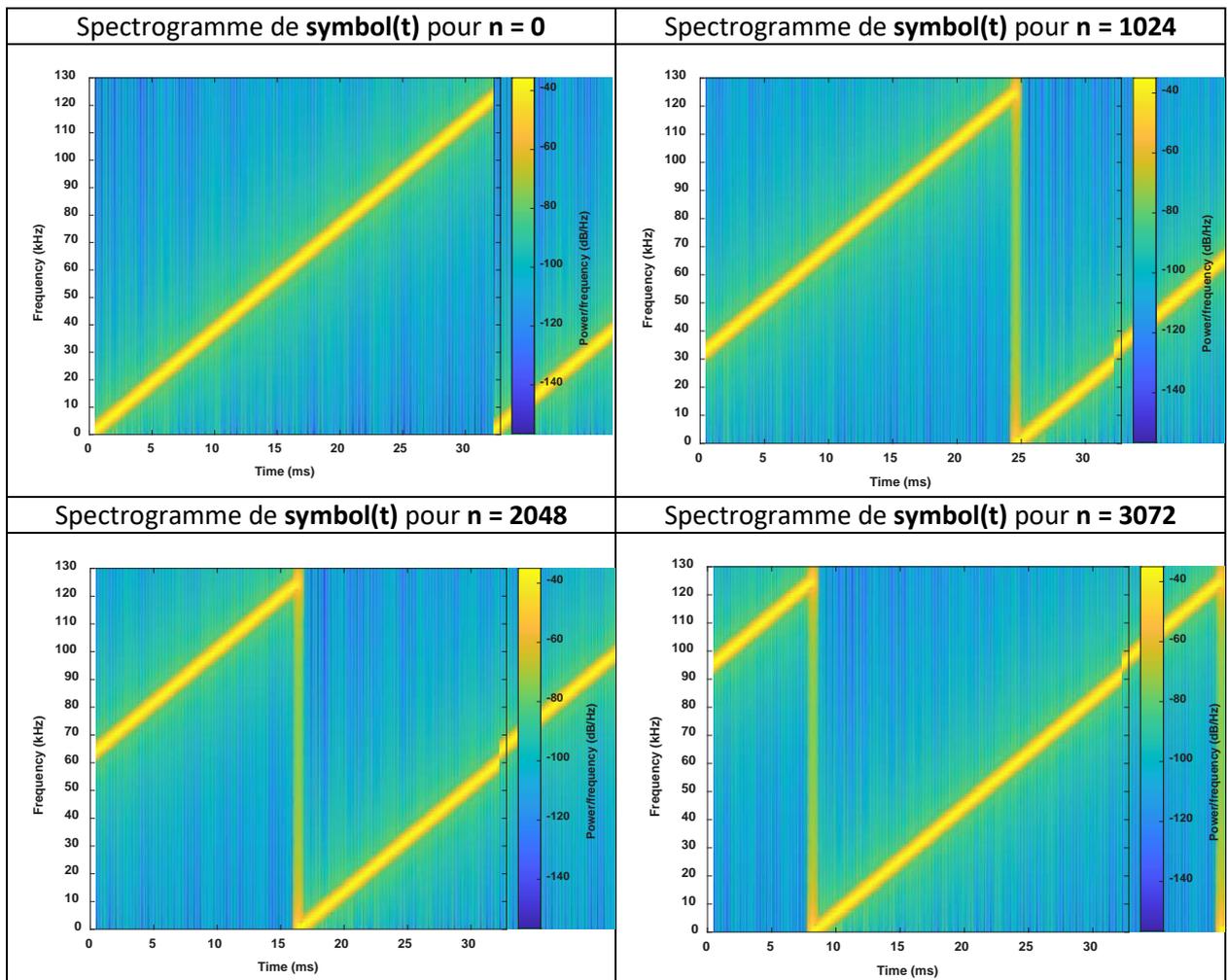


Figure 23: Spectrogramme des symboles (simulation)

3.3.3 Démodulation LoRa (Matlab)

La première étape du processus de démodulation décale $lora(t)$ dans la bande de base afin de trouver le symbole transmis $symbol(t)$. Puis le résultat est ensuite mélangé avec $down(t)$, qui est un down chirp (chirp inversé). Le résultat $demod(t)$ est filtré et nous utilisons une FFT pour récupérer l'indice n représentant les 12 bits.

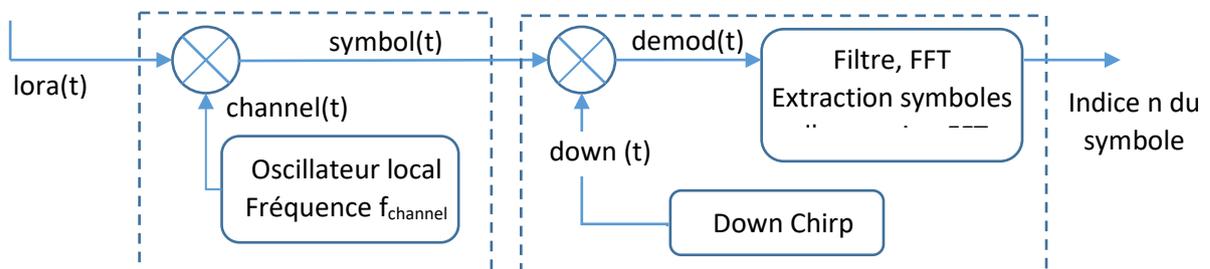


Figure 24: Schéma bloc de la démodulation LoRa

Un tel démodulateur peut être simulé en utilisant Matlab pour une transmission en SF12 et une bande passante de 125 kHz. Dans les spectrogrammes suivants, nous simulons uniquement la dernière partie, c'est-à-dire l'étape qui consiste à trouver $demod(t)$ après multiplication par un

down chirp. On s'aperçoit que demod(t) est un signal à fréquence fixe dont la valeur est directement représentative de l'indice n, et donc des 12 bits transmis.

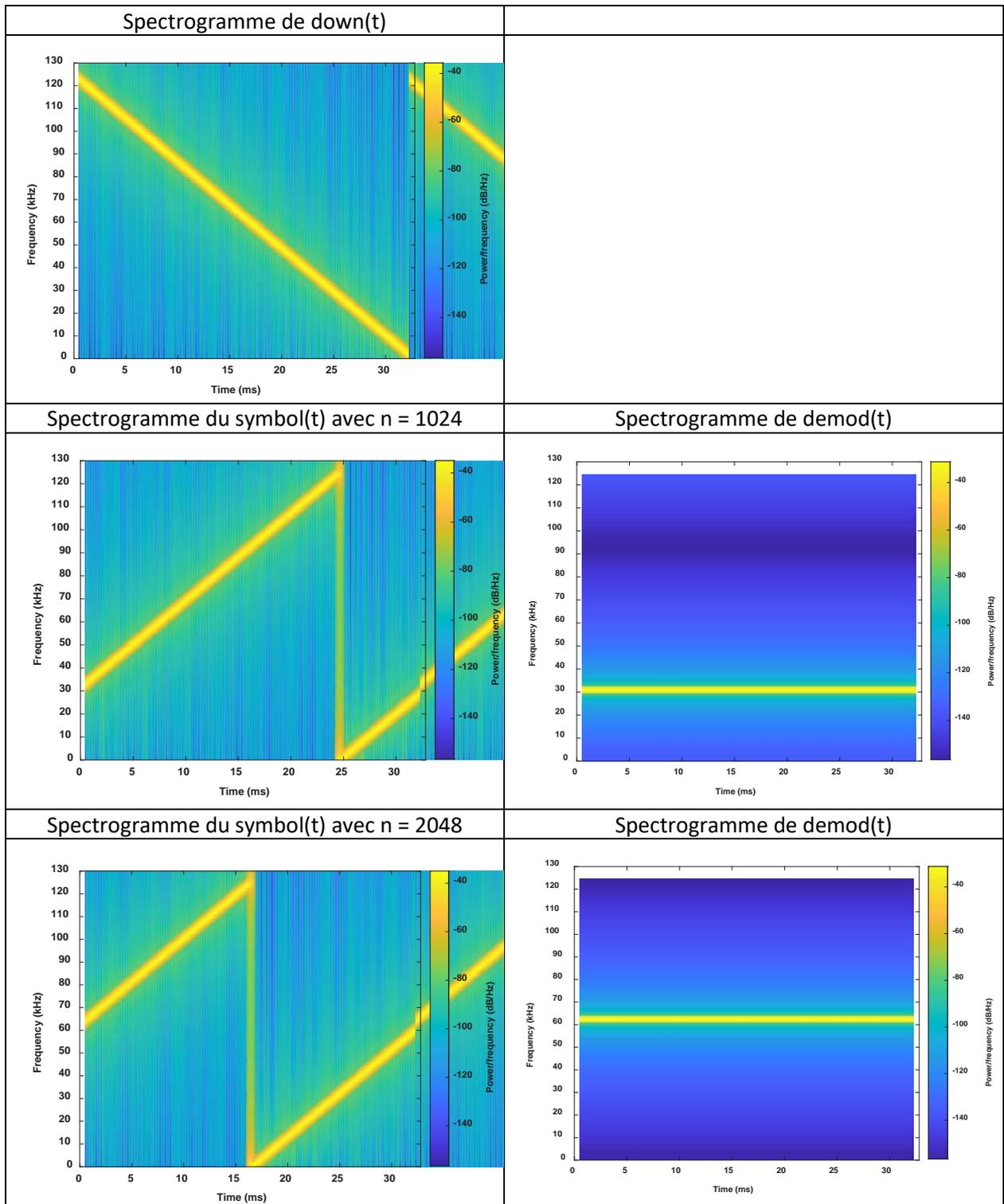


Figure 25: Spectrogramme de démodulation LoRa pour n=1024 et n=2048

Maintenant, nous pouvons imaginer un signal réel reçu sur le démodulateur. Soit symbol(t) composé de 12 symboles avec n prenant les valeurs aléatoires suivantes : 153, 4012, 2122, 251, 947, 3050, 21, 1555, 2954, 84, 454, 812.

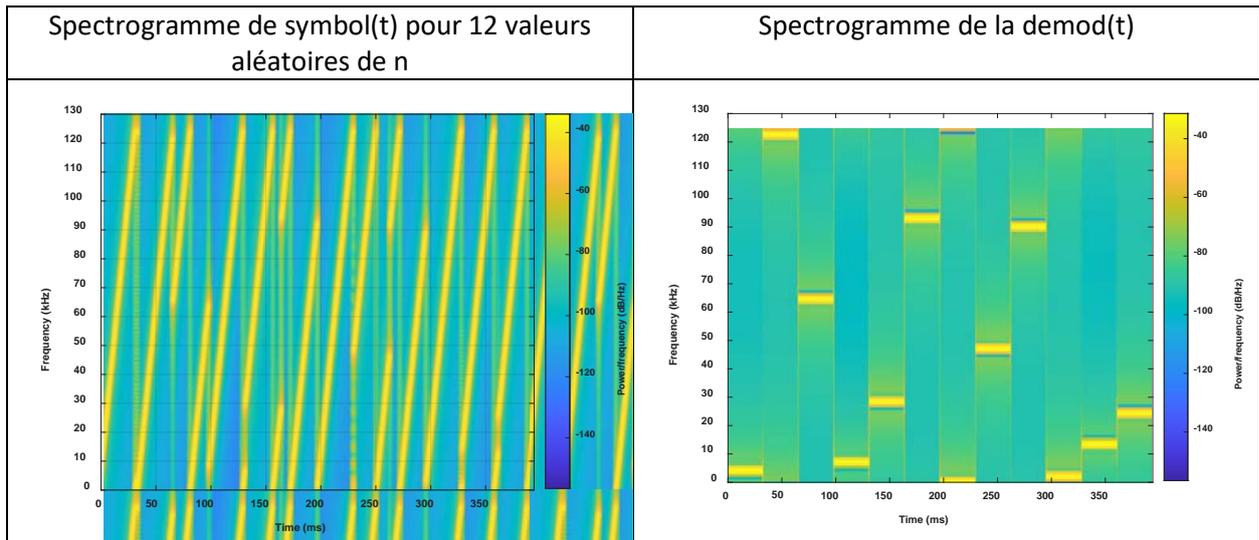


Figure 26: Spectrogramme de la démodulation LoRa pour 12 valeurs de n aléatoires

L'indice peut facilement être extrait de demod(t) avec un algorithme de FFT inverse.

3.4 Test réel d'une transmission LoRa

Pour cette démonstration, nous utilisons un Device LoRa transmettant un Payload simple avec les paramètres suivants :

- Canal : 868,1 Mhz
- Spreading factor : 12
- Bande passante : 125 kHz
- Coding Rate : 1 (4/5)
- Nombre de préambules : 8
- PHY Payload : "HELLO"

Le SDR (Software Defined Radio) ADALM-PLUTO sera utilisé comme récepteur LoRa. Nous utilisons SDR Angel [<https://github.com/f4exb/sdrangel>] pour piloter le PLUTO ADALM et afficher la modulation LoRa.



Figure 27: Analog Device ADALM PLUTO

Nous pouvons voir dans le spectrogramme suivant, les 8 up-chirp (préambule) plus 4,25 symboles de synchronisation (2 up-chirp et 2,25 down-chirp) indiquant le début de la trame.

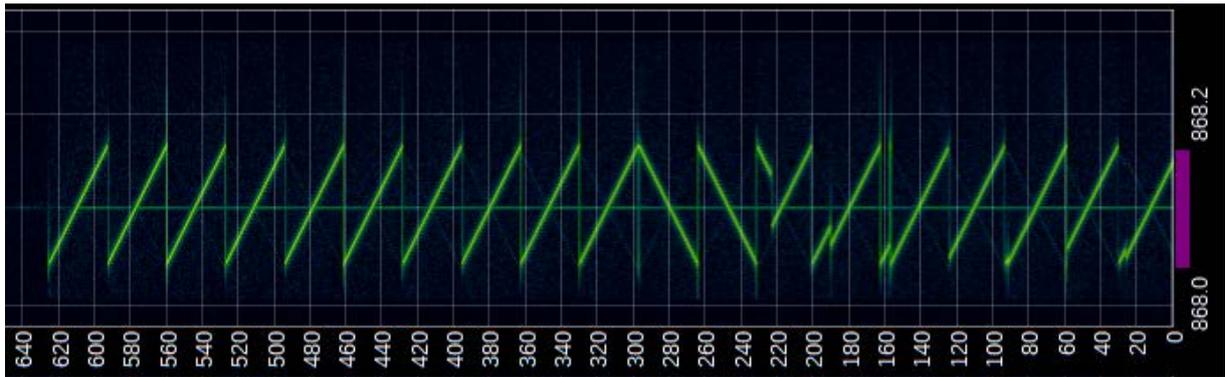


Figure 28: Spectrogramme d'une trame LoRa

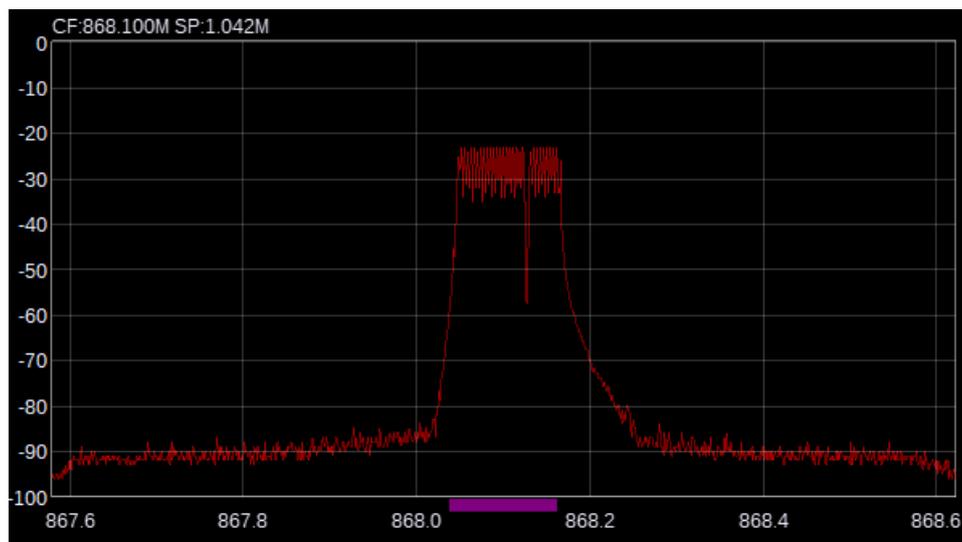


Figure 29: Spectre autour de la bande 868.1Mhz

3.5 Consommation d'énergie

Le protocole LoRaWAN est conçu pour cibler les applications à très faible consommation. Il est courant de trouver des dispositifs LoRaWAN ayant une autonomie de plusieurs années. La consommation réelle d'un système LoRa dépend de plusieurs paramètres :

- La quantité de données à transmettre (Payload)
- Le Spreading Factor
- Les éventuelles collisions à l'émission (provoquant des retransmissions)
- Le souhait d'avoir des acquittements des trames transmises
- Le rythme des transmissions
- La puissance de transmission du Transceiver
- La puissance consommée en veille entre deux transmissions

Ce [simulateur en ligne](#) permet d'avoir une première approximation de la consommation et donc de l'autonomie d'un Device LoRaWAN.

4 Le protocole LoRaWAN

4.1 LoRa - LoRaWAN - LoRa Alliance

4.1.1 La LoRa Alliance

La [LoRa Alliance](#) est une organisation à but non lucratif née en 2015 qui vise à développer la technologie et l'ensemble de l'écosystème LoRaWAN. Ses membres sont des entreprises très impliquées qui investissent dans la LoRa Alliance pour la développer. Toute structure peut demander à faire partie de la LoRa Alliance et ainsi à participer au développement de LoRaWAN.



➔ L'Université Savoie Mont Blanc fait partie de la LoRa Alliance depuis 2021.

4.1.2 Les versions du protocole

L'un des principaux rôles de la LoRa Alliance est la spécification et l'évolution du protocole LoRaWAN. Voici les différentes versions ainsi que leur évolution.

- **Version 1.0.0** (janvier 2015) : Version initiale.
- **Version 1.0.1** (février 2016) : Ajout d'un nouveau plan de fréquences pour la Chine et l'Australie. Correction et clarification de nombreux points.
- **Version 1.0.2** (juillet 2016) : Les sections sur la couche physique sont maintenant dans un document appelé "LoRaWAN Regional Parameters". Première version stable.
- **Version 1.1** (octobre 2017) : Amélioration de la sécurité et de l'itinérance. Ajout de nouvelles "Root Keys" et "Sessions Keys". Nouveaux "Frame Counters" et nouvelles commandes MAC. JoinEUI remplace AppEUI. Clarification sur la classe B et la classe C.
- **Version 1.0.3** (juillet 2018) : version 1.0.3 = version 1.0.2 + section Classe B de la version 1.1.
- **Version 1.0.4** (octobre 2020) : AppEUI devient JoinEUI. Nombreuses clarifications. Dernière version 1.0.x.

On constate que très tôt, une version 1.1 a été publiée, mais elle n'a pas été adoptée par l'industrie. La LoRa Alliance a ensuite continué la clarification de la version 1.0.x en ajoutant la version 1.0.3 et la version 1.0.4, qui devrait être la dernière de la série 1.0.x.



A moins que cela ne soit spécifié, ce document traite du protocole LoRaWAN version 1.0.x dans son ensemble.

Vous pouvez trouver sur le "[LoRa Alliance ressource HUB](#)", toutes les versions de la spécification LoRaWAN :

- [Spécification LoRaWAN version 1.0](#)
- [Spécification LoRaWAN version 1.0.1](#)
- [Spécification LoRaWAN version 1.0.2](#)

- [Spécification LoRaWAN version 1.0.3](#)
- [Spécification LoRaWAN version 1.0.4](#)
- [Spécification LoRaWAN version 1.1](#)

4.1.3 Les paramètres régionaux

Lorsque la LoRa Alliance publie une spécification, elle fournit un autre document appelé "LoRaWAN Regional Parameters". Ce document complémentaire décrit les spécificités régionales pour les différentes régions régulatrices dans le monde. La séparation des paramètres régionaux avec le reste de la spécification du protocole permet l'ajout ou la modification de nouvelles régions sans impacter sur cette dernière.

La spécification LoRaWAN : Ce document détaille le protocole LoRaWAN, par exemple les classes des Devices, le format des messages, le format des trames, la liste des commandes MAC, les modes d'activation (ABP, OTAA), etc...

Les paramètres régionaux : Ce document détaille les paramètres spécifiques à chaque région (EU868, EU433, US915...) comme par exemple les fréquences des canaux, le débit de données, la puissance de sortie, la taille maximale du Payload, etc...

4.1.4 La Différence entre le LoRa et LoRaWAN

Le protocole LoRa est le type de modulation utilisé entre deux Devices LoRa ou entre un Device et une Gateway. Lorsque nous parlons de l'ensemble de la chaîne de communication (du Device au serveur LoRaWAN), nous parlons alors du protocole LoRaWAN. LoRaWAN est une extension du protocole LoRa qui permet de connecter de manière sécurisée le Device à un serveur afin de fournir des données à l'utilisateur final.

- La couche physique LoRa : C'est le type de modulation (Chirp Spread Spectrum) et format de la trame utilisés pour envoyer des données entre un émetteur et un récepteur.
- Le protocole LoRaWAN : C'est l'architecture complète du réseau (Devices, Gateways, serveurs) et format de la trame permettant à un Device LoRaWAN de transmettre en toute sécurité des données à un serveur LoRaWAN.



Comprendre la différence entre le LoRa et le LoRaWAN en vidéo : <https://youtu.be/m1-B0jbfqOE>

4.2 La structure d'un réseau LoRaWAN

La Figure 30 montre l'ensemble de l'architecture LoRaWAN. Sur le côté gauche, on trouve les Devices LoRaWAN qui transmettent les données. L'utilisateur se trouve de l'autre côté et réceptionne les données qui ont transité sur le réseau.

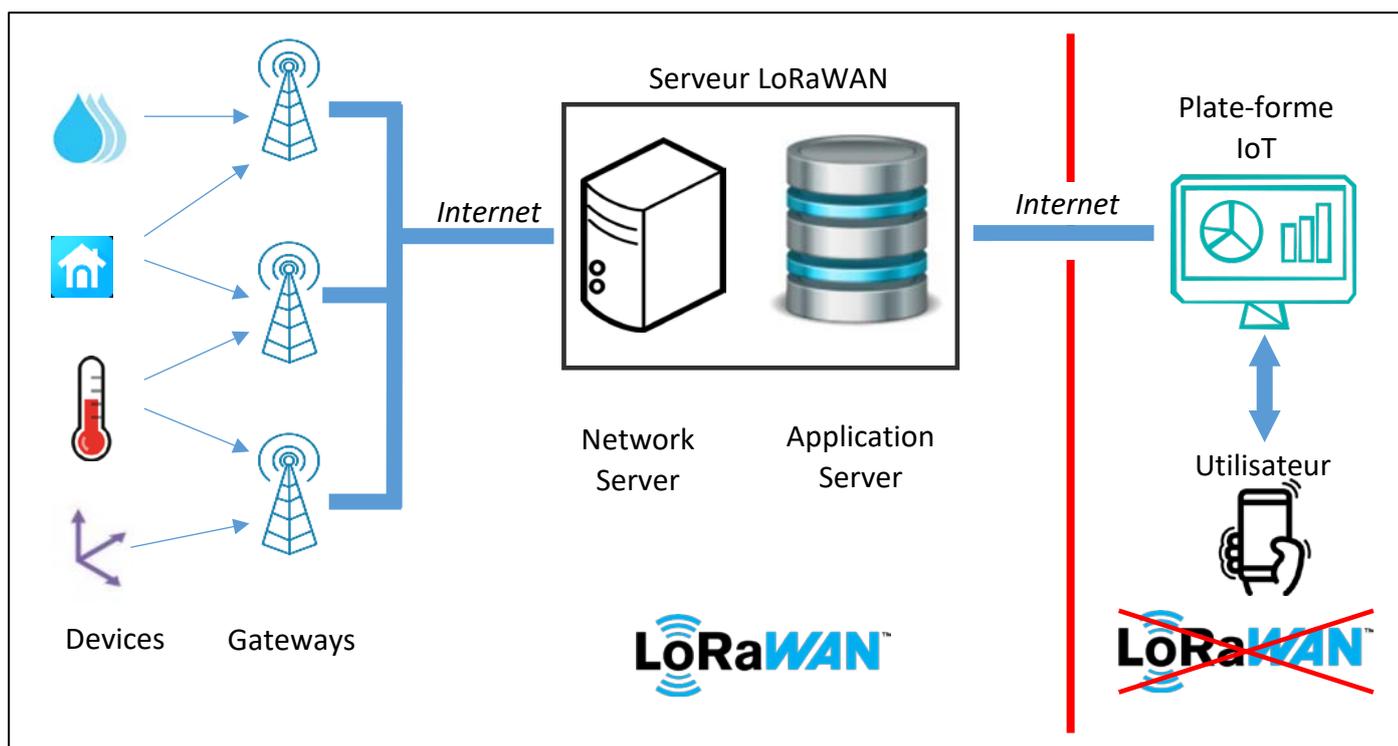


Figure 30: Architecture globale d'un réseau LoRaWAN

Les Devices LoRaWAN, les Gateways, le Network Server et l'Application Server constituent le cœur de l'architecture LoRaWAN, mais la plateforme IoT et la connexion de l'utilisateur (à droite sur la figure) n'ont rien à voir avec ce protocole car il s'agit simplement d'un service web classique.

4.2.1 Les Devices LoRaWAN

Les Devices LoRaWAN sont des systèmes électroniques embarqués appartenant au monde de l'IoT: faible consommation d'énergie, petite taille, faible puissance et faible coût. Ils disposent d'un Transceiver radio LoRa pour atteindre les Gateways. Un Device LoRaWAN ne s'adresse pas spécifiquement à une Gateway car toutes celles présentes dans la zone de couverture reçoivent les messages et les traitent.

Il existe des dizaines de fabricants de Devices LoRaWAN. En voici quelques exemples :

- ATIM [www.atim.com] Concepteur et fabricant de solutions de transmission de données sans fil depuis 1996. Pionnier sur les technologies LPWAN.



- nke-WATTECO [www.nke-watteco.fr] Concepteur et fabricant d'émetteurs radiofréquence pour de nombreux domaines d'application.



- adeunis [www.adeunis.com] Spécialiste des capteurs IoT. Expert en réseau LPWAN.



- Abeeway [www.abeway.com] Fournisseur de solutions de géolocalisation économes en énergie.



4.2.2 Les Gateways LoRaWAN

Elles écoutent sur tous les canaux et sur tous les Spreading Factor en même temps. Lorsqu'une trame LoRa est reçue, elle transmet son contenu par Internet au serveur LoRaWAN réseau qui a été préalablement renseigné dans la Gateway.

D'un côté, la Gateway reçoit un signal modulé en LoRa, et de l'autre côté, elle est connectée à internet via toutes les connexions possibles : 3G, 4G, 5G, Ethernet, LTE-M...

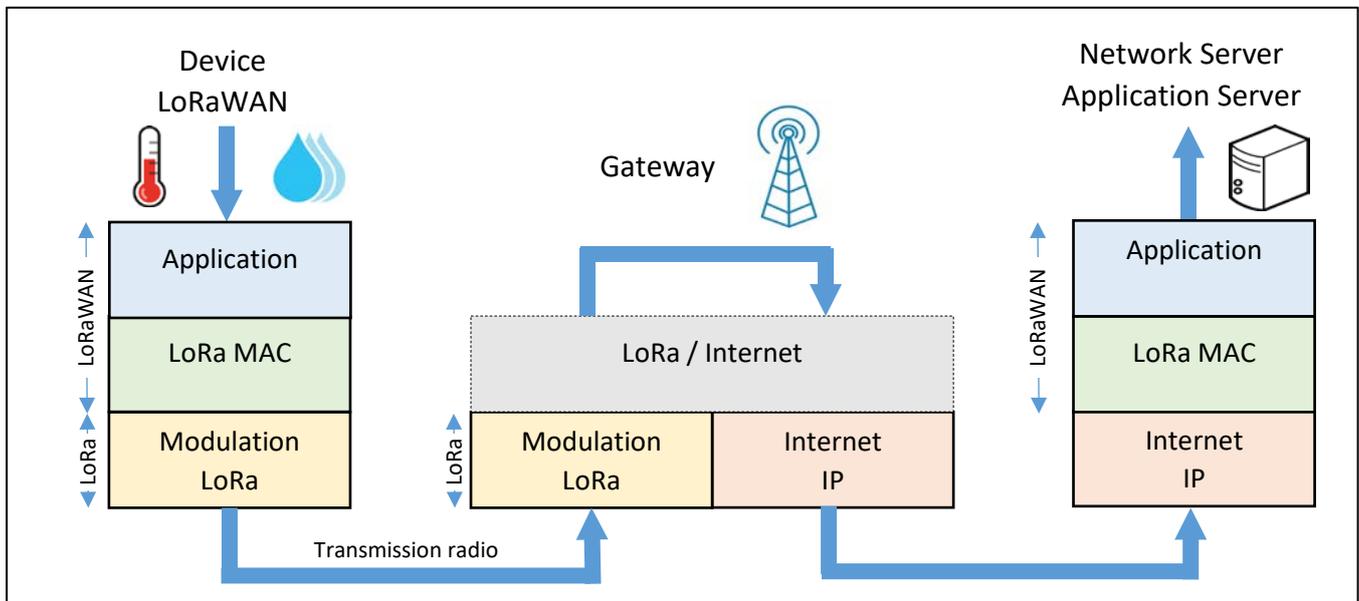


Figure 31: La Gateway LoRaWAN

Chaque Gateway LoRaWAN possède un identifiant unique (64 bits EUI). Cet identifiant est utile pour enregistrer et activer une Gateway sur un Network Server.

4.2.3 Le Network Server

Le Network Server reçoit les messages transmis par les Gateways et supprime les doublons (plusieurs Gateways peuvent recevoir le même message et le transmettre au même Network Server). Ensuite, le Network Server authentifie le message grâce à une clé AES 128 bits appelée **NwkSKey** (Network Session Key). Nous parlons bien ici d'authentification et non pas de chiffrement.

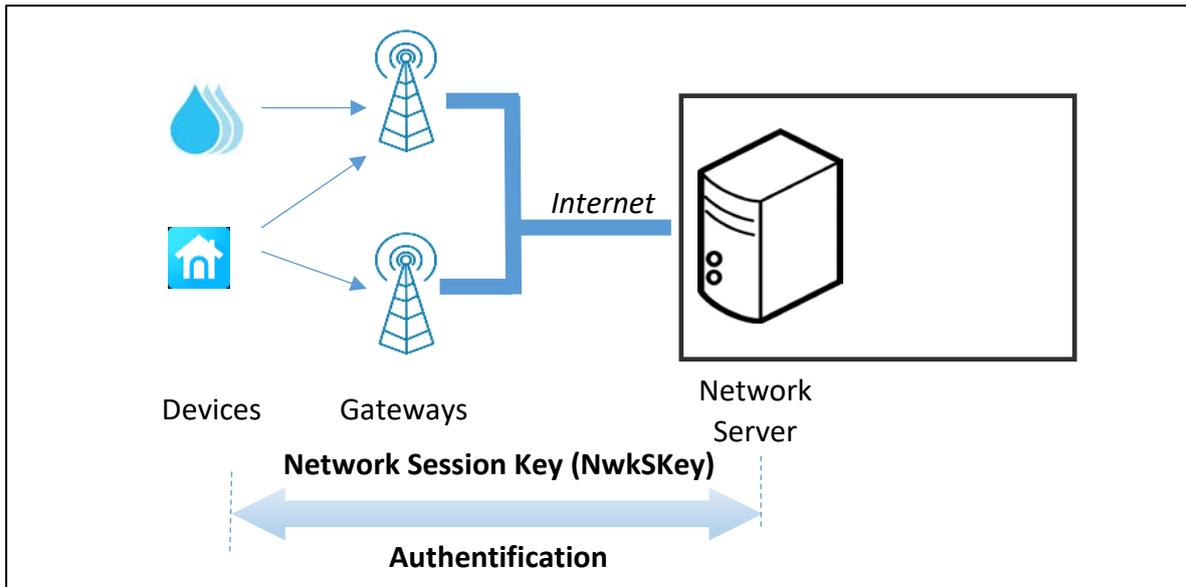


Figure 32: Authentification entre le Device LoRaWAN et le Network Server

- Si le processus d'authentification échoue, le Network Server abandonne le message LoRaWAN.
- Si le processus d'authentification réussit, le Network Server transfère le message à l'Application Server.

4.2.4 L'Application Server

L'Application Server reçoit des messages chiffrés d'un Network Server. Le chiffrement et le déchiffrement se font grâce à une clé AES 128 bits appelée **AppSKey (Application Session Key)**. Nous détaillerons ce processus dans le chapitre 4.2.7

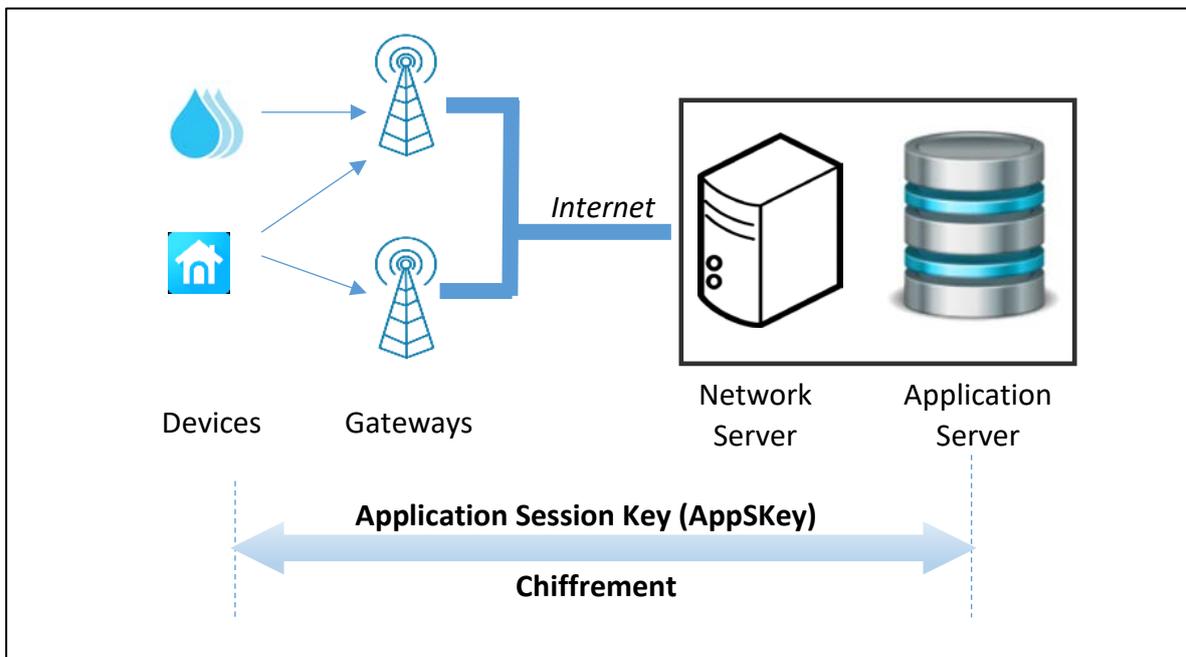


Figure 33: Chiffrement entre le Device LoRaWAN et l'Application Server

4.2.5 La plateforme IoT

C'est là que se trouve l'application utilisateur. Les trois services principaux sont :

1. Un connecteur avec l'Application Server pour collecter les données. La plupart du temps, cela se fait avec les protocoles HTTP(S) ou MQTT(S). Pendant le développement, nous utilisons parfois les versions non sécurisées de ces protocoles
2. Une base de données pour stocker les données.
3. Un tableau de bord accessible par l'utilisateur via une page web ou une application mobile.

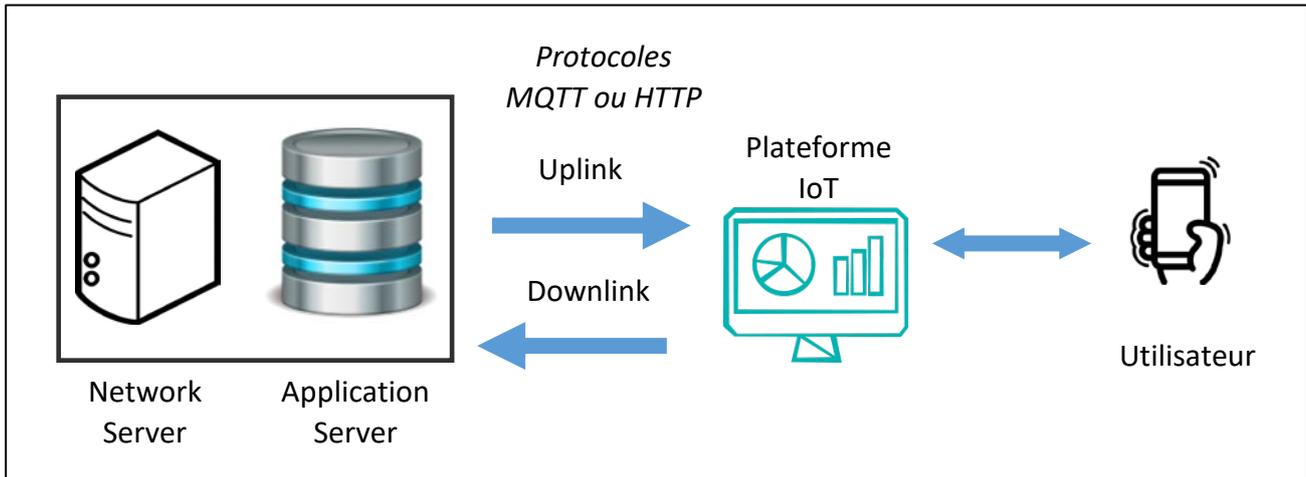


Figure 34: Connexion entre l'Application Server et la plateforme IoT

En LoRaWAN, nous utilisons principalement le flux uplink (données du Device vers le serveur). Comme nous l'expliquerons dans la section 4.3 il est également possible de transférer des données vers les Devices, c'est ce qu'on appelle le flux downlink.

4.2.6 Network Server + Application Server = serveur LoRaWAN

Le serveur LoRaWAN est l'association du Network Server et de l'Application Server.



Le terme "LoRaWAN Server" n'est pas défini par la LoRa Alliance. Il est simplement utilisé dans ce livre pour combiner les deux notions d'authentification et de chiffrement.

- Nous disposons d'une Network Session Key (**NwkSKey**) pour l'authentification entre le Device LoRaWAN et le Network Server.
- Nous disposons d'une Application Session Key (**AppSKey**) pour le chiffrement entre le Device LoRaWAN et l'Application Server.

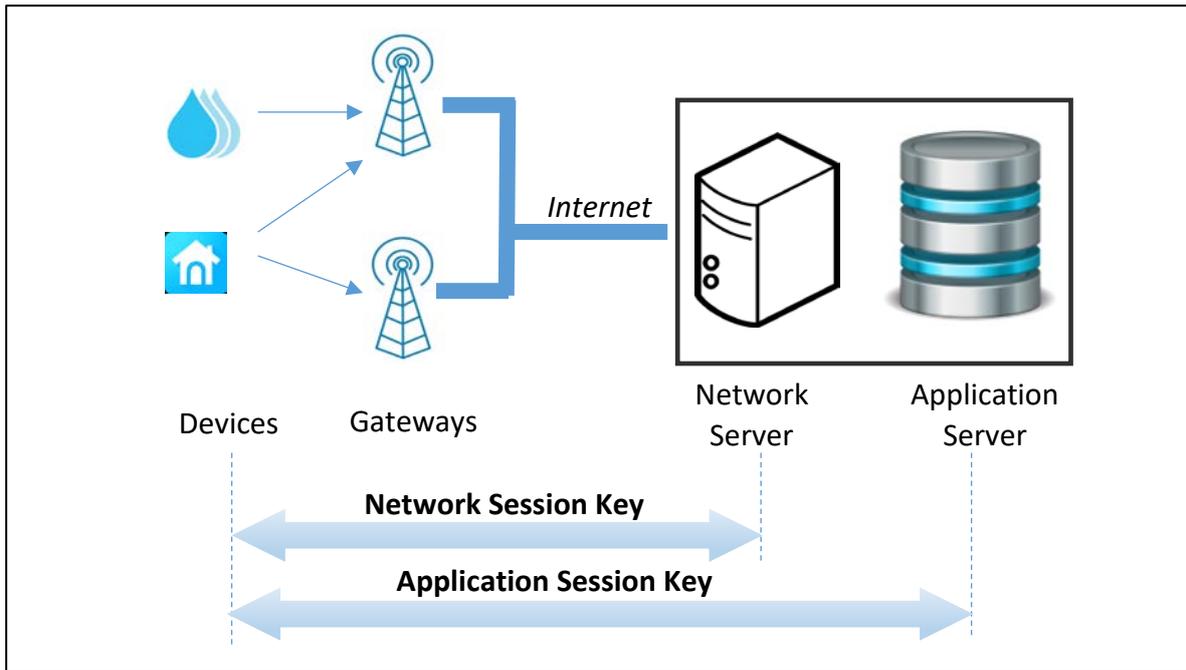


Figure 35: Processus d'authentification et de chiffrement

Malheureusement, le Network Server et l'Application Server sont souvent utilisés dans la même infrastructure. Ce n'est pas l'idée première d'un réseau LoRaWAN car cela ne permet pas d'assurer une sécurité de bout en bout.



La "sécurité de bout en bout" (end to end security) est la capacité à chiffrer un message utilisateur sur le Device et à le déchiffrer uniquement lorsqu'il a atteint l'application utilisateur finale (plateforme IoT par exemple).

Chaque fois que vous voyez vos données déchiffrées dans votre serveur LoRaWAN, cela signifie que la sécurité de bout en bout n'est pas respectée. L'exemple ci-dessous montre un Payload qui transite en clair dans le Serveur LoRaWAN (nombre "01 02 03 04 05"). Votre entreprise pourrait ne pas accepter que votre prestataire de service gérant votre serveur LoRaWAN puisse voir vos données.

Time	Type	Data preview
↑ 18:31:50	Forward uplink data message	MAC payload: 01 02 03 04 05

Figure 36: Message en clair "01 02 03 04 05" dans le serveur LoRaWAN

La solution consiste à intégrer votre **propre** Application Server. Ce n'est pas toujours facile car les protocoles de communication entre Network Server et Application Server ne sont pas normalisés. Cette normalisation est en cours par la LoRa Alliance. L'architecture optimisée sera alors la suivante:

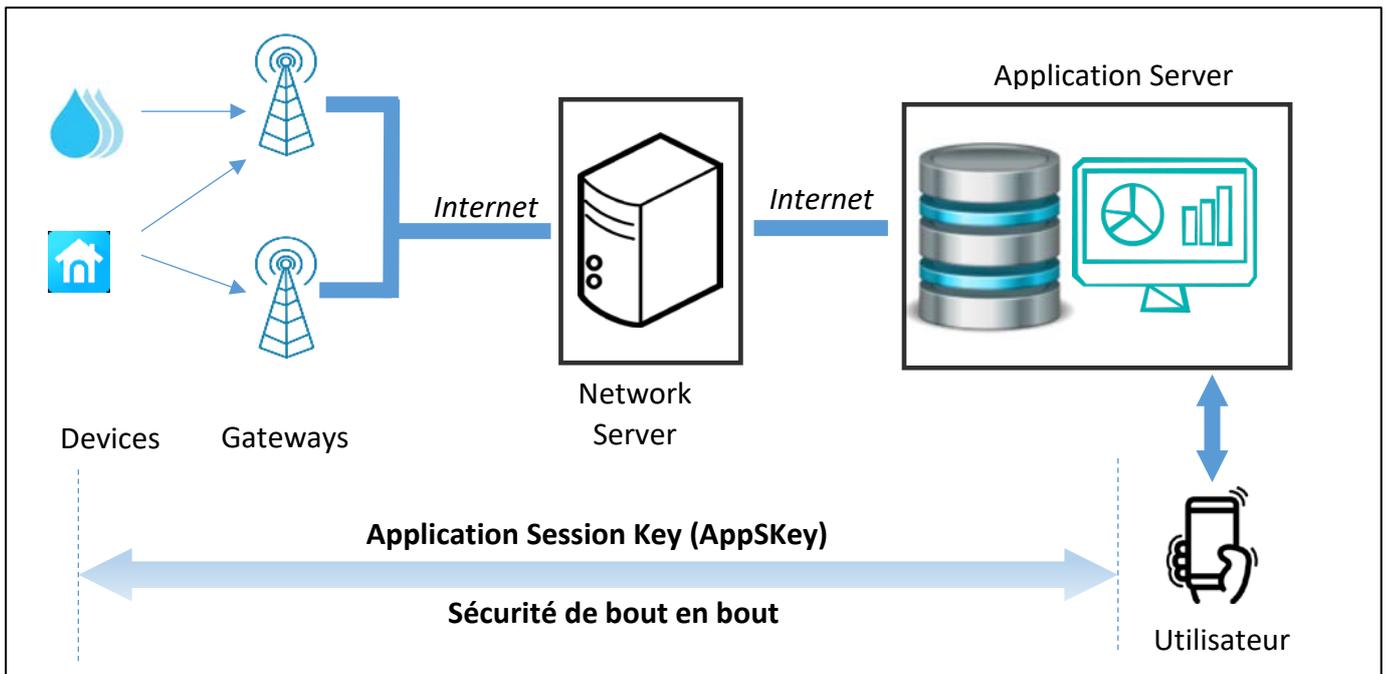


Figure 37: Sécurité de bout en bout dans un réseau LoRaWAN

Dans cette situation, le nom de "Application Server" (défini par la LoRa Alliance) a plus de sens car il s'agit réellement des applications de l'utilisateur combiné avec la sécurité de bout en bout promise par le LoRaWAN.

4.2.7 Le chiffrement des données

L'Application Session Key (**AppSKey**) est utilisée pour chiffrer les données de l'utilisateur sur le Device LoRaWAN. Les données seront déchiffrées sur l'Application Server. Il s'agit d'un chiffrement symétrique : l'AppSKey du Device doit être la même que celle stockée sur l'Application Server.

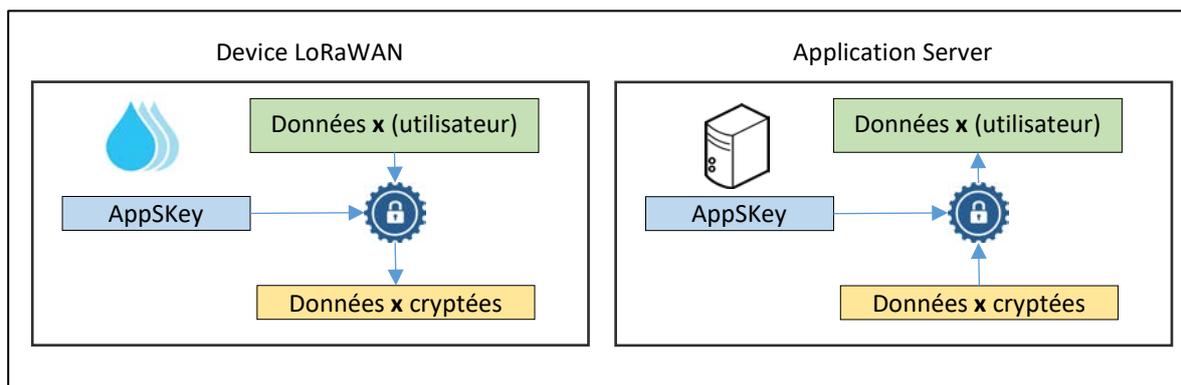


Figure 38: Processus de chiffrement et de déchiffrement

Il n'y a aucun moyen pour la Gateway et le Network Server de comprendre la valeur réelle des données de l'utilisateur. Le canal est sécurisé (confidentiel).

4.2.8 L'authentification avec le Network Server

La Network Session Key (**NwkSKey**) est utilisée pour l'authentification entre le Device LoRaWAN et le Network Server. Afin de réaliser cette authentification, un champ **MIC (Message Integrity Control)** est ajouté à la trame. Le MIC dépend des données transmises cryptées et de la NwkSKey. Lors de la

réception, le même calcul est effectué. Si les NwksKey sont les mêmes dans le Device et dans le Network Server, alors le MIC transmis sera le même que celui calculé à la réception.

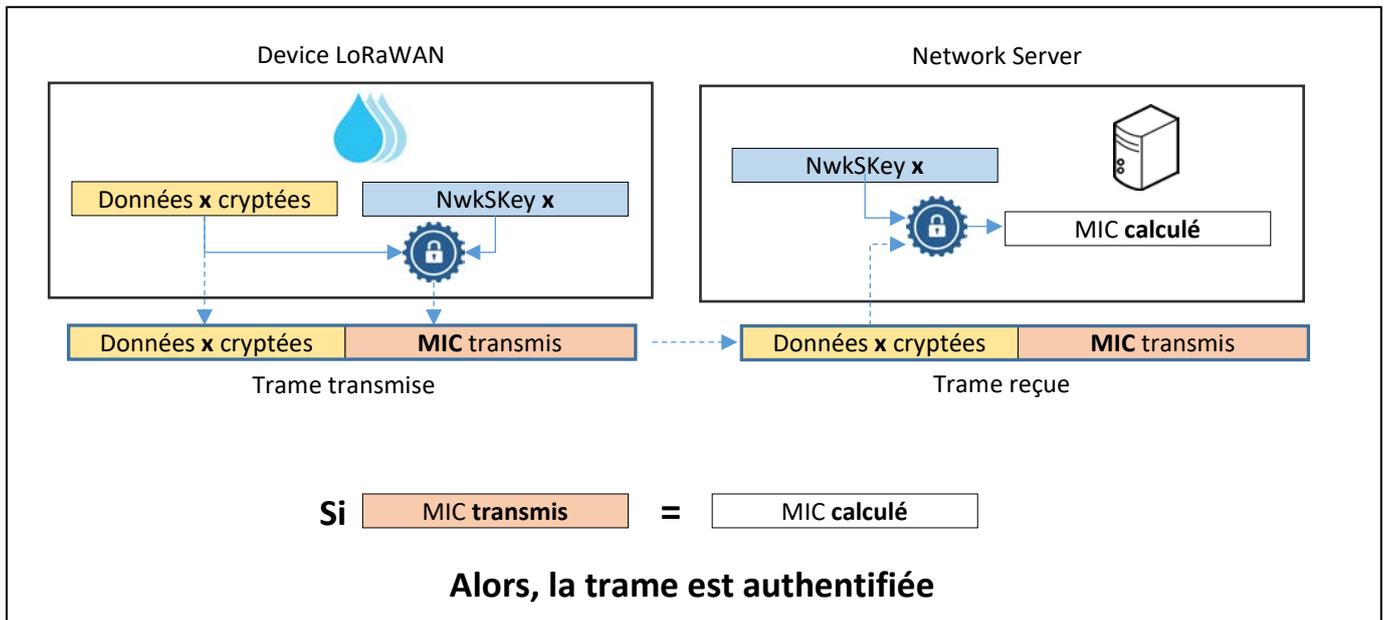


Figure 39: Authentification du Device par le Network Server

4.2.9 Combinaison de l'authentification et du chiffrement

Nous pouvons maintenant représenter sur la même figure la construction de la trame LoRaWAN avec l'authentification et le chiffrement.

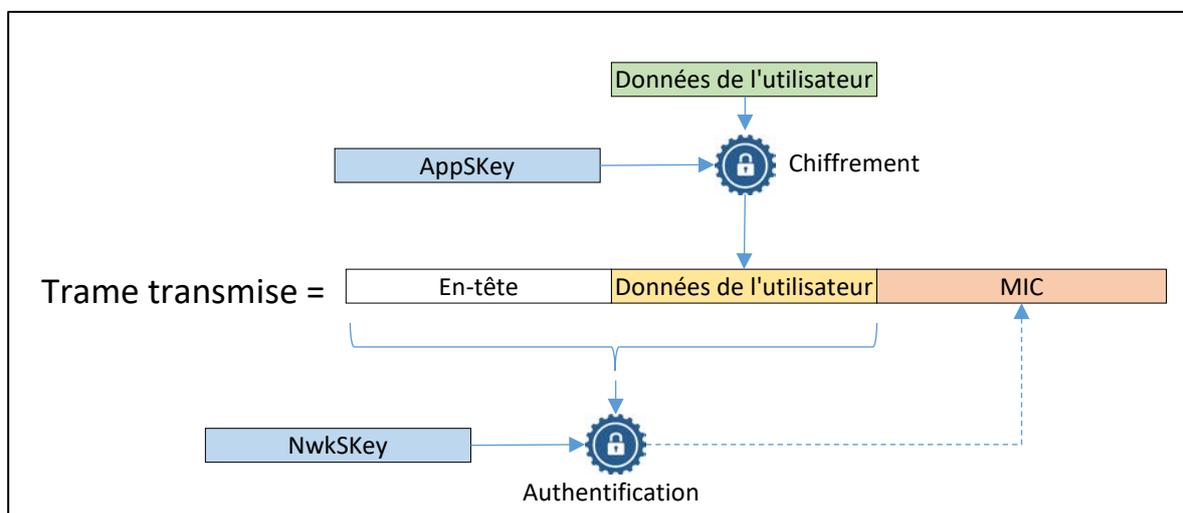


Figure 40: Chiffrement, puis calcul du MIC à transmettre

Du côté du serveur, le processus de déchiffrement ne s'applique que si l'authentification réussit.

4.3 Les classes de Devices LoRaWAN

Les Devices LoRaWAN sont classés en trois catégories (A, B, C) en fonction de leur consommation d'énergie et de leur accessibilité en downlink, c'est-à-dire la facilité avec laquelle un utilisateur peut transmettre une trame au Device.

4.3.1 Classe A (tous) : Application basse consommation

Tous les Devices LoRaWAN sont de classe A. Chaque Device peut émettre (uplink) vers la Gateway sans vérifier la disponibilité de celle-ci. Cette transmission est suivie de deux fenêtres de réception très courtes. Le Network Server (via la Gateway) peut transmettre un message downlink pendant le slot RX1 ou RX2, mais pas les deux.

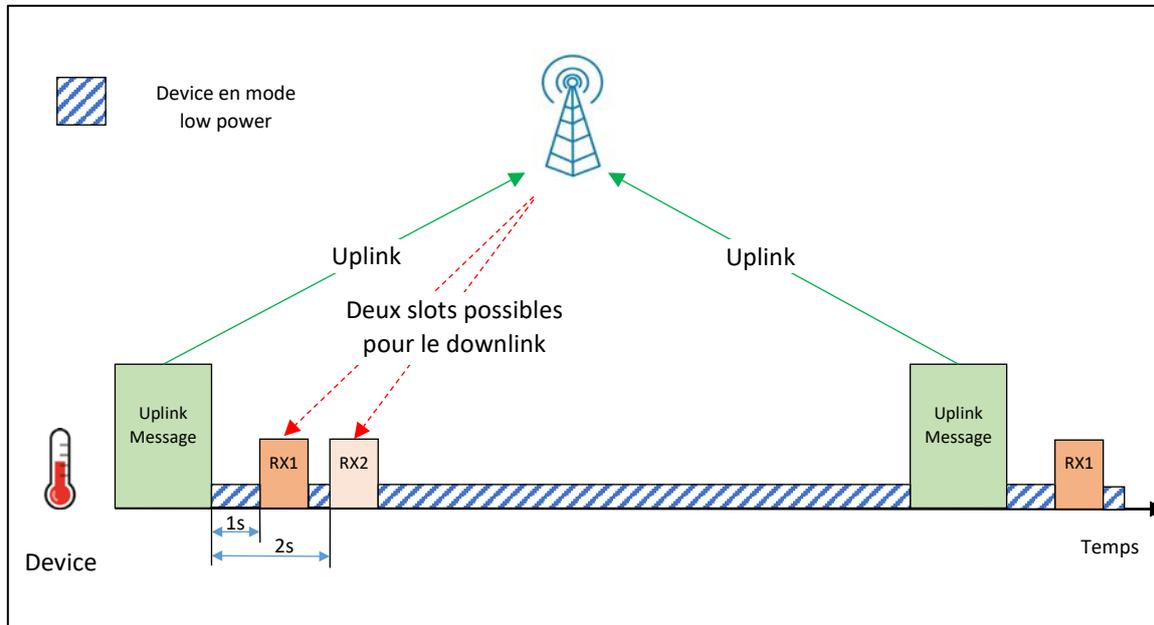


Figure 41: Fenêtres de réception RX1 et RX2 pour un Device de classe A

La durée des fenêtres doit être au moins égale à la longueur du préambule ($12,25 T_{\text{symbole}}$). Lorsqu'un préambule est détecté, le récepteur doit rester actif jusqu'à la fin de la transmission. Si la trame reçue pendant la première fenêtre de réception était destinée au Device LoRaWAN alors la deuxième fenêtre n'est pas ouverte.

Première fenêtre de réception RX1 :

- Le Slot RX1 est programmé par défaut à 1 seconde +/- 20 μ s après la fin de la transmission. Cette valeur peut être modifiée en fonction de la configuration du Network Server. Assez régulièrement, vous pourrez trouver des valeurs de 5 secondes.
- Le canal, le Spreading Factor et la largeur de bande passante sont les mêmes que ceux choisis lors de la transmission.

Deuxième fenêtre de réception RX2 :

- Le Slot RX2 est programmé par défaut à 2 secondes +/- 20 μ s après la fin de la transmission uplink. Cette valeur peut être modifiée en fonction de la configuration du Network Server. Assez régulièrement, vous pourrez trouver des valeurs de 5 secondes.
- Le canal, le Spreading Factor et la bande passante sont configurables mais statiques.



➔ Un Device de classe A ne peut pas recevoir s'il n'a pas transmis de données en uplink. Par conséquent, vous ne pouvez pas le joindre facilement.

Tous les Devices démarrent et rejoignent le réseau en tant que Devices de classe A.

4.3.2 Classe B (Beacon) : Créneau de réception programmé

Les Devices de classe B se comportent de la même manière que les Devices de classe A, mais d'autres fenêtres de réception sont programmées à des moments précis. Afin de synchroniser les fenêtres de réception des Devices LoRaWAN, les Gateways doivent émettre un signal régulier et périodique appelé balise ou "Beacon".

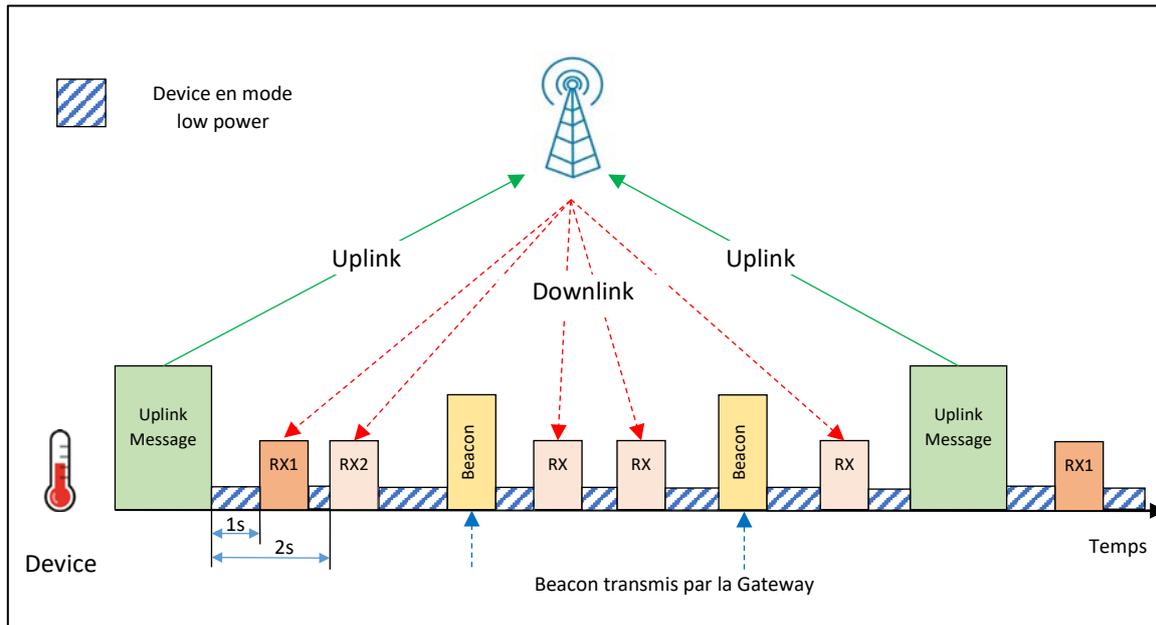


Figure 42: Fenêtres de réception pour un Device de classe B



Un Device de classe B peut être joint régulièrement sans devoir nécessairement émettre. En revanche, il consomme plus d'énergie qu'un Device de classe A.

Tous les Devices peuvent décider de passer en classe B si leur firmware le permet. Bien évidemment, le Network Server doit en être informé, mais cela n'est pas géré par LoRaWAN avant la version 1.2.0 du protocole. La spécification complète du Device de classe B a été publiée à partir de la version 1.0.3 du protocole LoRaWAN.

4.3.3 Classe C (Continu) : Écoute en permanence

Les Devices de classe C ont des fenêtres de réception qui sont constamment ouvertes entre deux transmissions. Ces Devices consomment beaucoup plus d'énergie.

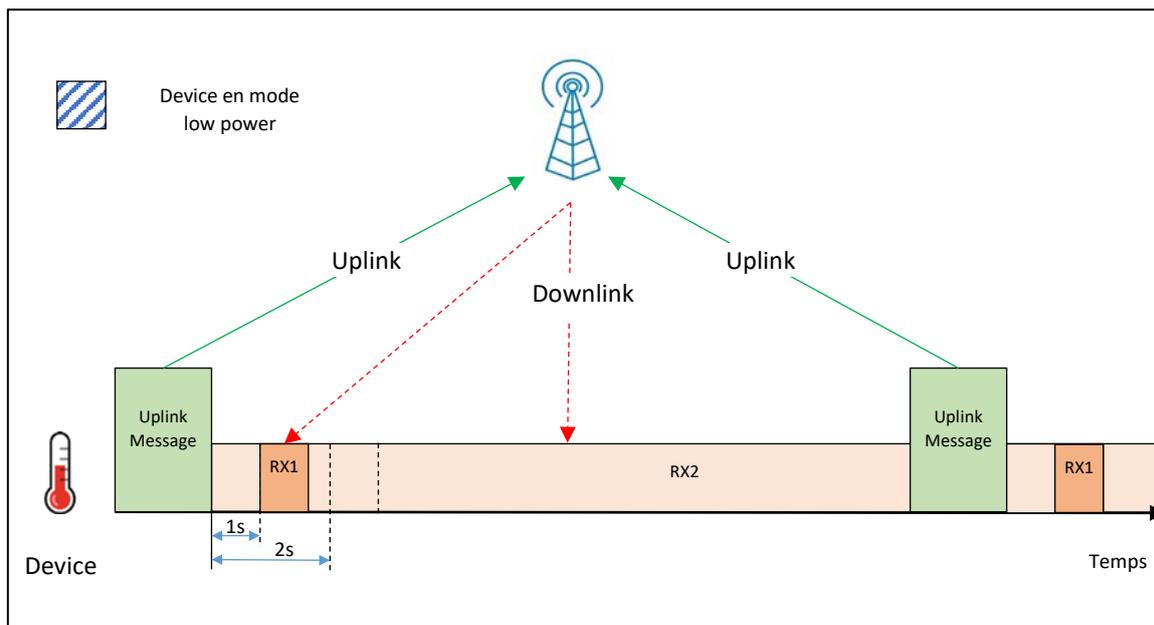


Figure 43: Fenêtres de réception pour un Device de classe C

Le Device LoRaWAN est en écoute permanente entre deux messages uplink. Les paramètres de réception sont fixes (canal, Spreading Factor et bande passante), à l'exception de RX1 qui a toujours le même comportement que dans les classes A et B.



Un Device de classe C est toujours joignable. Cependant, c'est la classe de Device qui consomme le plus d'énergie.

Tous les Devices peuvent décider de passer en classe C. Bien évidemment, le Network Server doit en être informé, mais cela n'est pas géré par LoRaWAN avant la version 1.2.0 de la spécification du protocole.

4.3.4 Résumé des classes de Devices

D'après les figures précédentes, nous pouvons remarquer que :

- Un Device de classe B est également un Device de classe A (les emplacements RX1 et RX2 sont toujours présents).
- Un Device de classe C est également un Device de classe A (les emplacements RX1 et RX2 sont toujours présents).

La classe B et la classe C sont donc une extension de la classe A. Concernant leur consommation, nous pouvons les représenter comme suit :

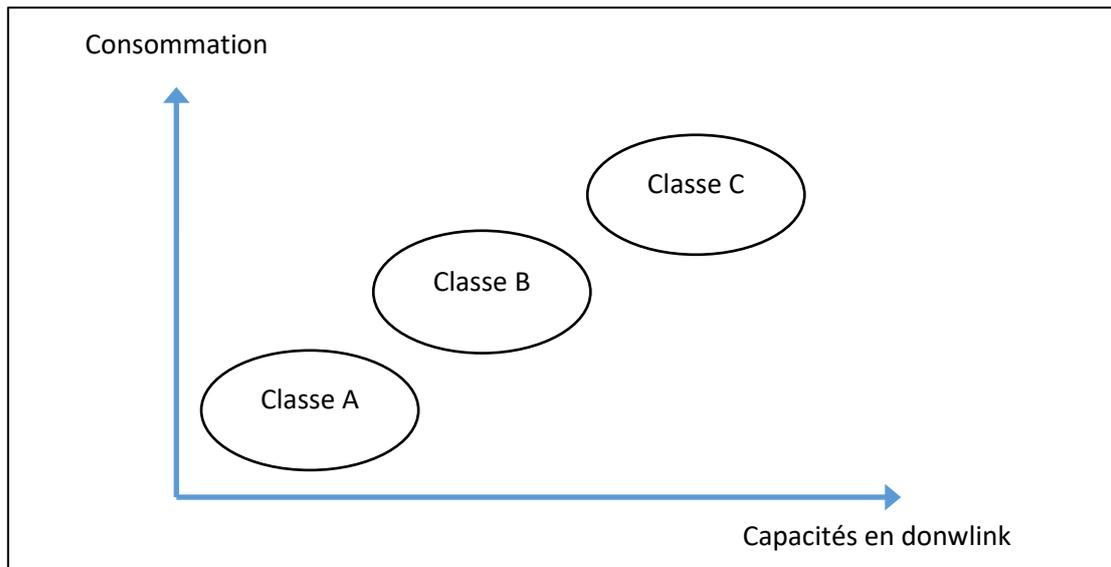


Figure 44: Consommation d'énergie et capacités en downlink

4.3.5 Quelle Gateway pour le downlink?

Le transfert de données le plus courant en LoRaWAN est l'uplink. Mais un utilisateur peut avoir besoin d'envoyer des données à son Device et donc utiliser les capacités de downlink. Dans ce cas, on peut se demander quelle Gateway sera choisie pour transférer les données vers le Device. En effet, sa localisation n'est pas forcément connue à l'avance et il est évident que toutes les Gateways ne vont pas envoyer le message sur l'ensemble du réseau.

La Gateway utilisée pour le downlink est celle qui a reçu le dernier message en uplink. Si plusieurs Gateways ont reçu le dernier message uplink, nous pouvons sélectionner celle qui a la meilleure valeur du RSSI pour assurer la meilleure chance d'atteindre le Device.



Un message downlink n'atteindra jamais un Device LoRaWAN s'il n'a jamais transmis auparavant, quelle que soit sa classe A, B ou C.

4.4 Activation des Devices LoRaWAN : ABP et OTAA

En LoRaWAN, les trois éléments essentiels à la communication sont le **DevAddr** pour l'identification du Device, ainsi que deux clés : la **NwkSKey** pour l'authentification et la **AppSKey** pour le chiffrement. Deux méthodes sont possibles pour fournir ces informations à la fois sur le Device et sur le serveur LoRaWAN :

- Activation By Personalization : **ABP**.
- Over The Air Activation : **OTAA**.

4.4.1 ABP : Activation By Personalization

C'est la méthode la plus simple. C'est donc peut-être celle que nous avons tendance à utiliser pour tester un prototype et établir une communication LoRaWAN.

- Des **DevAddr**, **NwkSKey** et **AppSKey** statiques sont stockés dans le Device.
- Les même **DevAddr**, **NwkSKey** et **AppSKey** sont stockés dans le serveur LoRaWAN.



En ABP, toutes les informations nécessaires à la communication sont déjà connues par le Device, le Network Server et l'Application Server.

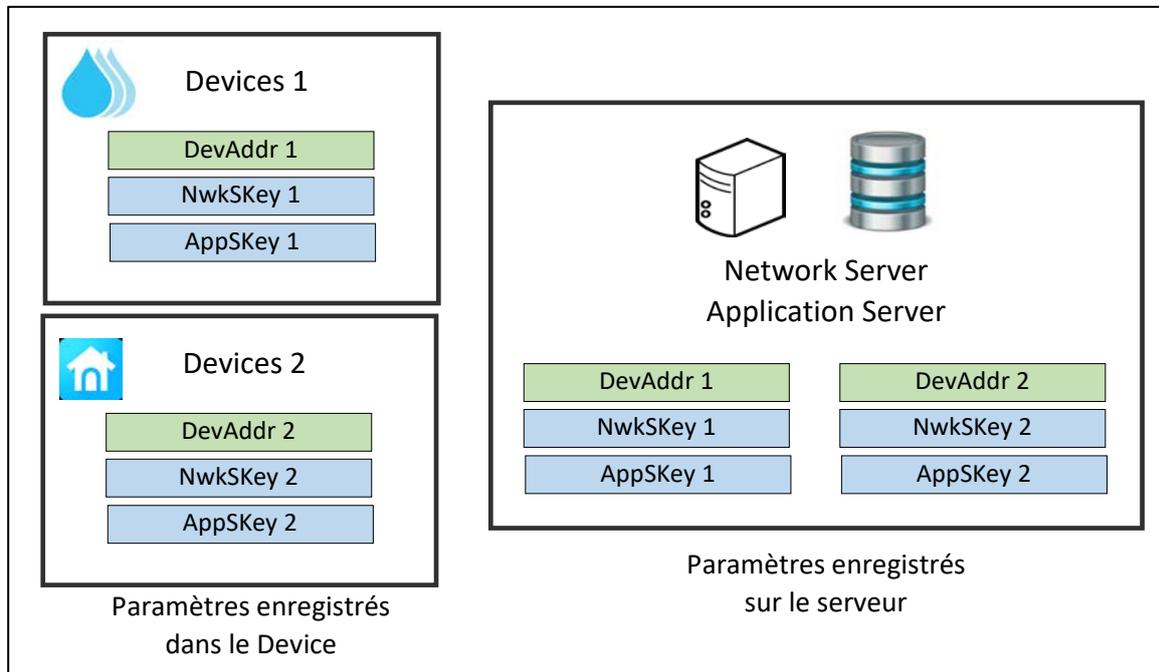


Figure 45: DevAddr, NwkSKey et AppSKey en ABP

Dès que le Device démarre, il peut envoyer et recevoir des messages LoRaWAN.

4.4.2 OTAA : Over The Air Activation

Avec ce mode d'activation, le **DevAddr**, l'**AppSKey** et le **NwkSKey** seront générés lors d'une procédure de Join. Pour réaliser cette procédure, le Device LoRaWAN doit être configuré avec les éléments suivants :

- **DevEUI**
- **AppEUI/JoinEUI**
- **AppKey**

Le Network Server doit connaître les mêmes **DevEUI**, **AppEUI/JoinEUI**, et **AppKey** et le but principal du Join-Request est de récupérer la configuration finale : **DevAddr**, **NwkSKey** et **AppSKey** des deux côtés.



Tous les éléments nommés "EUI" (Extended Unique Identifier) sont toujours uniques et ont une taille de 8 octets. Les clés sont toujours sur 128 bits (16 octets).

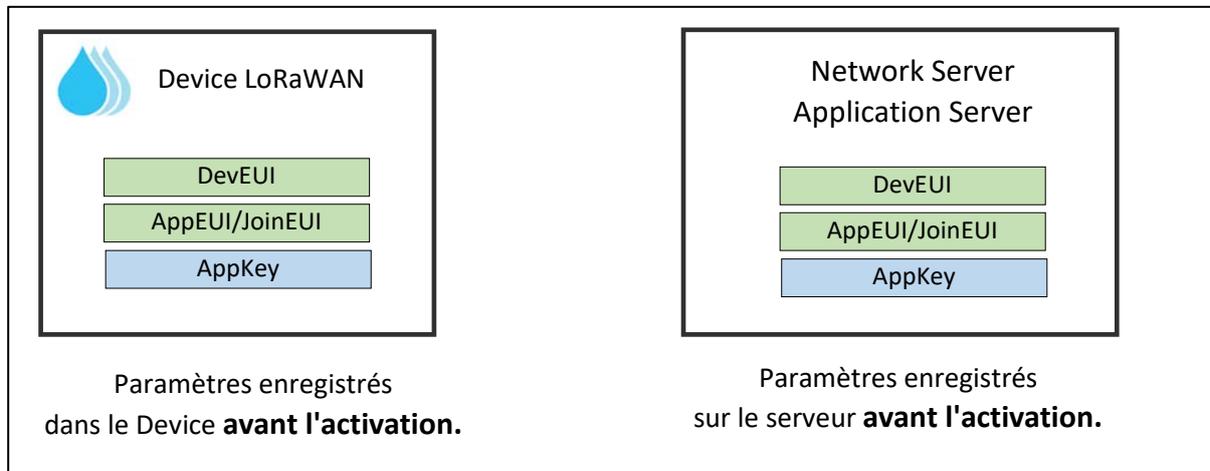


Figure 46: Paramètres stockés **avant** l'activation OTAA

Pendant la procédure de Join, les mêmes paramètres sont générés de part et d'autre : **DevAddr**, **NwkSKey** et **AppSKey**.

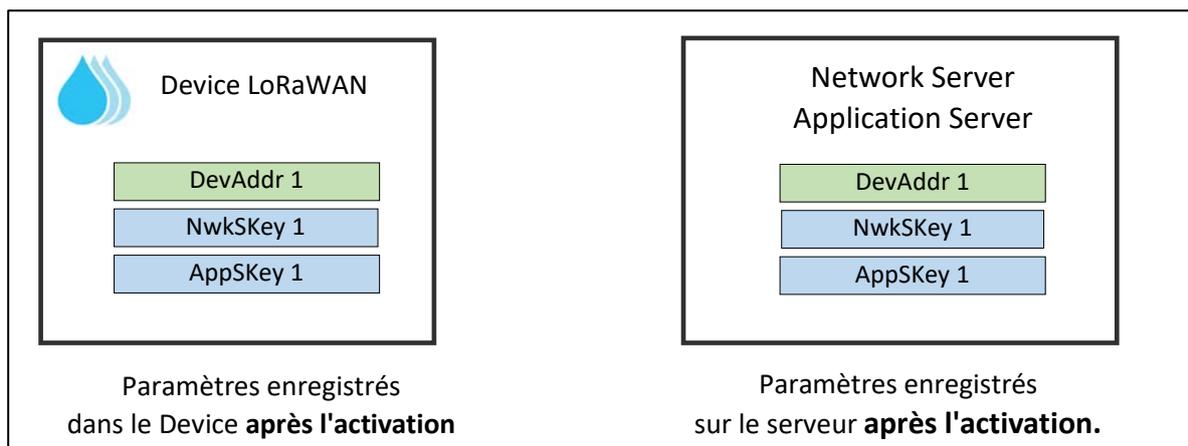


Figure 47: Configuration **après** le Join-Request (OTAA)

Nous expliquerons plus tard les avantages et les inconvénients de chaque mode d'activation. Pour l'instant, nous devons comprendre la signification de la configuration initiale stockée dans le Device et sur le serveur avant le Join-Request :

- **DevEUI** : Identifiant unique pour le Device LoRaWAN. C'est l'équivalent d'une adresse MAC sur Ethernet. Certains Devices LoRaWAN ont déjà un DevEUI fixe stocké lors de la programmation en usine et ne peuvent pas être modifiés.
- **AppKey** : Clé AES 128 utilisée pour authentifier le Join-Request, pour crypter le Join-Accept et pour générer les "Sessions Keys" (Network Session Key et Application Session Key). Cette clé doit être gardée secrète et ne jamais être partagée avec qui que ce soit.
- **AppEUI/JoinEUI** : Ce paramètre a une signification différente selon la version du protocole LoRaWAN. Dans les versions LoRaWAN 1.0.3 et antérieures, il s'agissait d'un identifiant

d'application (AppEUI). A partir de LoRaWAN 1.0.4, ce paramètre a été renommé en JoinEUI car il définit un identifiant de serveur (Join Server) dont nous verrons l'utilité plus tard.

Pour rester simple et dans un but éducatif, nous n'utiliserons pas de Join Server au début de ce livre (du moins, nous ne le feront pas apparaître). Nous pouvons donc simplifier cet EUI et utiliser "0000000000000000" comme AppEUI/JoinEUI.

Pour rappel, nous détaillons une nouvelle fois l'objectif de la configuration finale après la procédure de Join :

- **NwkSKey** : Utilisé pour l'authentification avec le Network Server.
- **AppSKey** : Utilisé pour le chiffrement des données avec l'Application Server.
- **DevAddr** : identifiant de 32 bits au sein d'un réseau LoRaWAN.

La Figure 48 montre la représentation simplifiée de la procédure de Join.

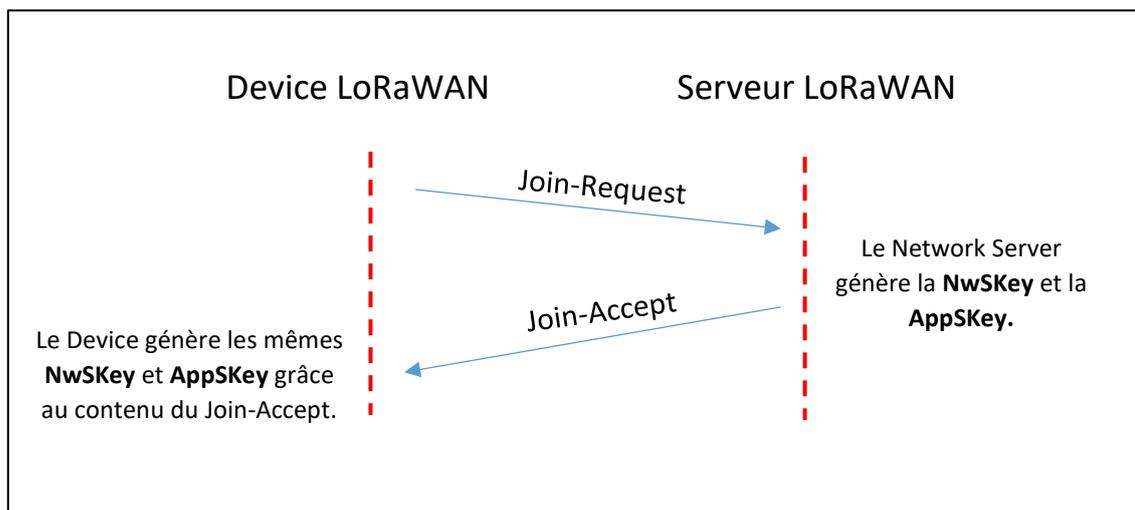


Figure 48: Join-Request et Join-Accept en OTAA

4.4.3 La procédure de Join

La Figure 48 ne représente que la procédure simplifiée de l'activation. Dans ce chapitre, nous allons aller plus loin afin de mieux comprendre le processus.

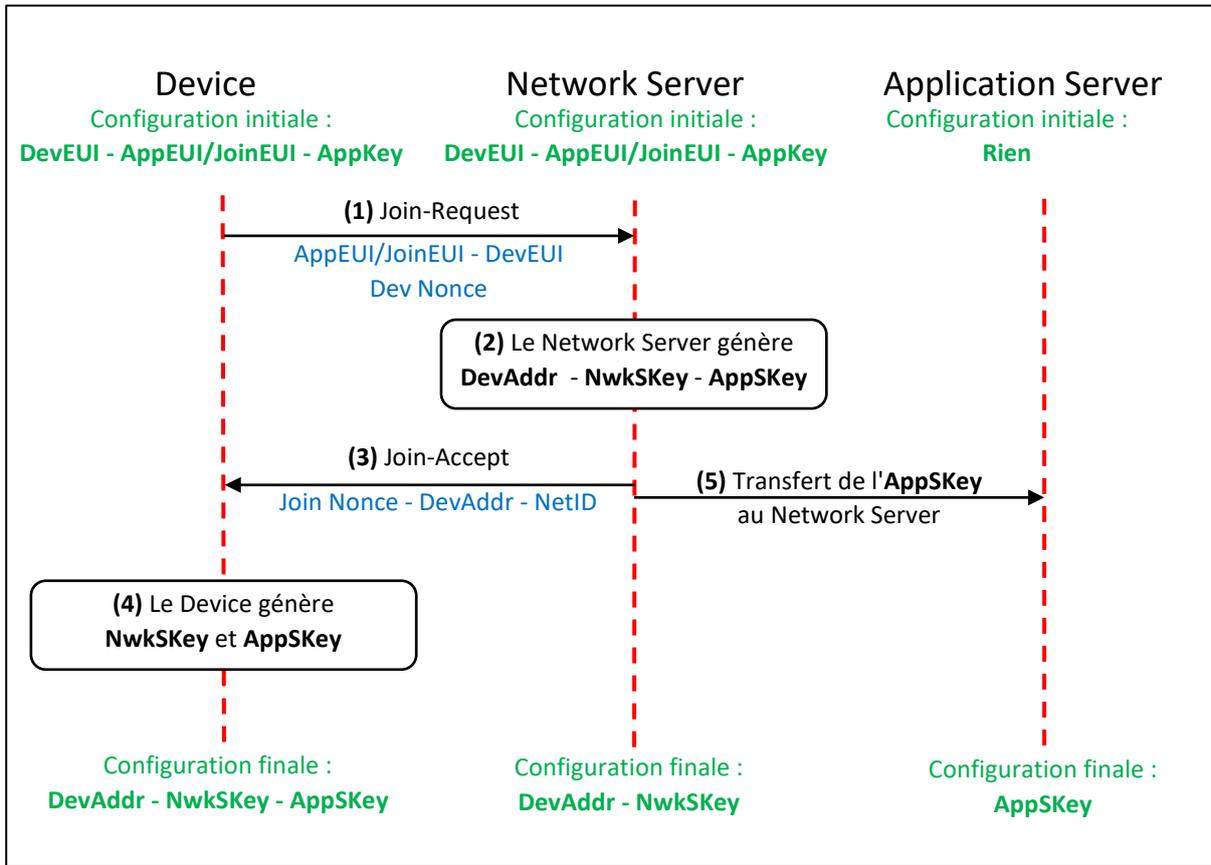


Figure 49: Détail du Join-Request et Join-Accept

(1) - Le Device LoRaWAN transmet un MIC (**M**essage **I**ntegrity **C**ode) pour authentifier le Join-Request afin que seul un Device enregistré sur le Network Server puisse être autorisé. Le champ MIC (4 octets) est calculé grâce à l'AppKey, au JoinEUI, au DevEUI et au DevNonce. La trame Join-Request n'est pas cryptée, vous pouvez donc facilement voir l'AppEUI/JoinEUI et le DevEUI sur les logs de votre Gateway ou du Network Server. DevNonce est juste un nombre aléatoire ou un compteur pour prévenir des attaques par Replay.

Size (octets)	8	8	2
Join-Request payload	JoinEUI	DevEUI	DevNonce

Figure 50: La trame Join-Request

La Figure 51 représente un Join-Request capturé sur le serveur LoRaWAN d'Activity en utilisant l'outil Wireless-Logger. Nous pouvons clairement voir le DevEUI, le JoinEUI, le DevNonce et le MIC qui a été calculé.

Last packets					
			Local Timestamp	DevEUI ▲	MIC
📄	↑	join	2021-11-19 08:51:36.299	E24F43FFFE44BFEE	06ed7af5
Mtype: JoinRequest					
Mac (hex): 0000000000000000eebf44feff434fe23c3a06ed7af5					
MAC.Command.JoinRequest					
MAC.JoinRequest JoinEUI : 0x0000000000000000					
MAC.JoinRequest DevEUI : 0xe24f43fffe44bfEE					
MAC.JoinRequest DevNonce : 0x3a3c					

Figure 51: DevEUI, JoinEUI, DevNonce et MIC dans un Join-Request.

(2) Si le Device est authentifié, le Network Server génère un DevAddr, NwkSKey et AppSKey.

(3) Une trame Join-Accept est transmise 5 secondes (RX1) ou 6 secondes (RX2) après le Join-Request. La NwkSKey et l'AppSKey ne sont pas directement transférées.

Size (octets)	3	3	4	1	1	(16) optional
Join-Accept payload	JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

Figure 52: La trame Join-Accept

Le DevAddr et le NetID sont les deux premières informations que le Device enregistre. Il reçoit également les paramètres nécessaires à une bonne communication avec le Network Server :

- Les paramètres pour le downlink (**DLSettings**)
- Le délai pour le downlink (**RXDelay**)
- La liste des canaux que le Device doit utiliser (**CFList**)

La dernière valeur est le JoinNonce qui doit être utilisé pour recalculer le NwkSKey et AppSKey du côté du Device.

Le Join-Accept est crypté par l'AppKey, donc seul le Device peut comprendre son contenu. La Figure 53 représente le Join-Accept avec son contenu crypté capturé 5 secondes après le Join-Request.

Last packets					
			Local Timestamp	DevEUI ▲	MIC
📄	↓	join	2021-11-19 08:51:41.299	E24F43FFFE44BFEE	
Mtype: JoinAccept					
Requested RX1/RX2Delay: 5000					
Mac (hex): 2073b53c5d26349ef88c58bd49068e835c42dabab58f482de9a0c804afcd7f7695					
Encrypted Content					

Figure 53: Join-Accept

(4) Le Device récupère le DevAddr, puis génère la NwkSKey et l'AppSKey en utilisant le contenu du Join-Accept.

(5) L'AppSKey est transférée au Application Server.



A partir du LoRaWAN 1.0.3, le Network Server n'est plus en charge de la procédure d'activation. Cette procédure est gérée par un autre serveur appelé **Join Server**. Ainsi, l'AppSKey n'est pas connue par le Network Server (NS) et donc le NS n'a pas accès aux données de l'utilisateur. C'est une grande amélioration en termes de sécurité. Nous verrons cette architecture plus tard dans ce livre.

4.5 Avantages et inconvénients de l'ABP et de l'OTAA

Nous avons discuté des deux méthodes d'activation d'un Device et nous devons maintenant expliquer quand il est plus approprié d'utiliser l'une ou l'autre méthode.

4.5.1 Sécurité

Les points faibles de chaque méthode sont les clés stockées en permanence dans le Device LoRaWAN:

- Clés de session : **NwkSKey** et **AppSKey** en ABP.
- Clé racine : **AppKey** en OTAA.

Elles doivent donc être stockées dans des mémoires hautement sécurisées.

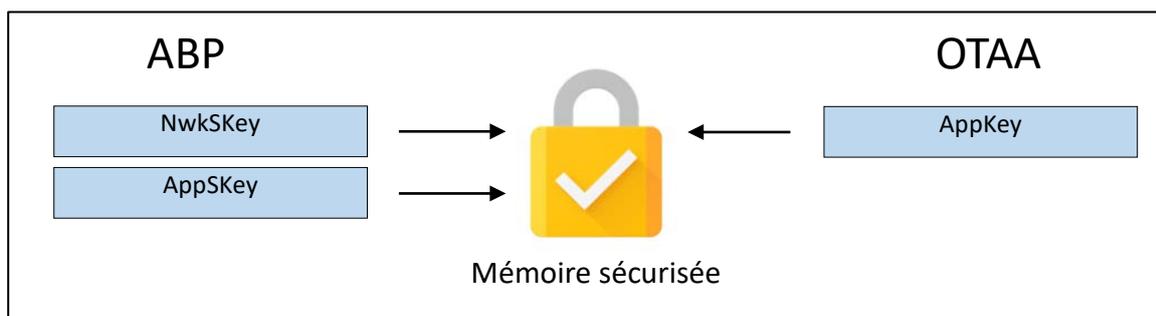


Figure 54: Clés stockées dans une mémoire sécurisée

Pendant, étant donné que la méthode ABP conserve éternellement les clés de session, il y a plus de chances qu'elles soient volées par une attaque par force brute, surtout si le Device rejoue les mêmes séquences plusieurs fois ou se réinitialise avec le même comportement : ceci était autorisé jusqu'à LoRaWAN 1.0.3. De plus, la manipulation des clés de session en ABP (si l'on veut changer de réseau LoRaWAN par exemple) donne plus de possibilités d'exposer les clés lors du transfert et donc de les rendre visibles.



Du point de vue de la sécurité, le mode d'activation OTAA doit toujours être privilégié.

4.5.2 Les changements de réseau

Un client peut décider de changer de fournisseur gérant le Network Server. Dans le cas du mode d'activation ABP, il devra transférer manuellement tous les DevAddr et les clés de session d'un serveur à l'autre. De plus, il ne peut pas être certain que l'ancien fournisseur efface toutes les clés de son côté et ne continue à écouter le trafic. Il s'agit d'une menace pour la sécurité.

Dans le cas d'OTAA, un Join Server peut être utilisé. Nous présenterons plus tard le rôle de ce serveur, mais pour l'instant, nous pouvons seulement dire que le client qui souhaite changer de

réseau devra simplement indiquer au Join Server qu'un autre Network Server est utilisé. Les clés racines (AppKey) resteront en lieu sûr et seules les clés de sessions seront régénérées (grâce à un nouveau Join-Request).

4.5.3 Protection contre les attaques par Replay (messages uplink)

Les clés AES 128 bits permettent le chiffrement des informations transmises en LoRaWAN. Malgré ce chiffrement, une attaque connue en transmission radio est l'attaque par Replay : le pirate enregistre les trames chiffrées transmises sur le réseau LoRa et les retransmettra ultérieurement. Même si le pirate ne comprend pas le contenu (les trames sont chiffrées), les données qui sont transportées sont bien comprises par l'Application Server. Des actions non souhaitées peuvent donc être réalisées en répétant simplement une trame précédente.

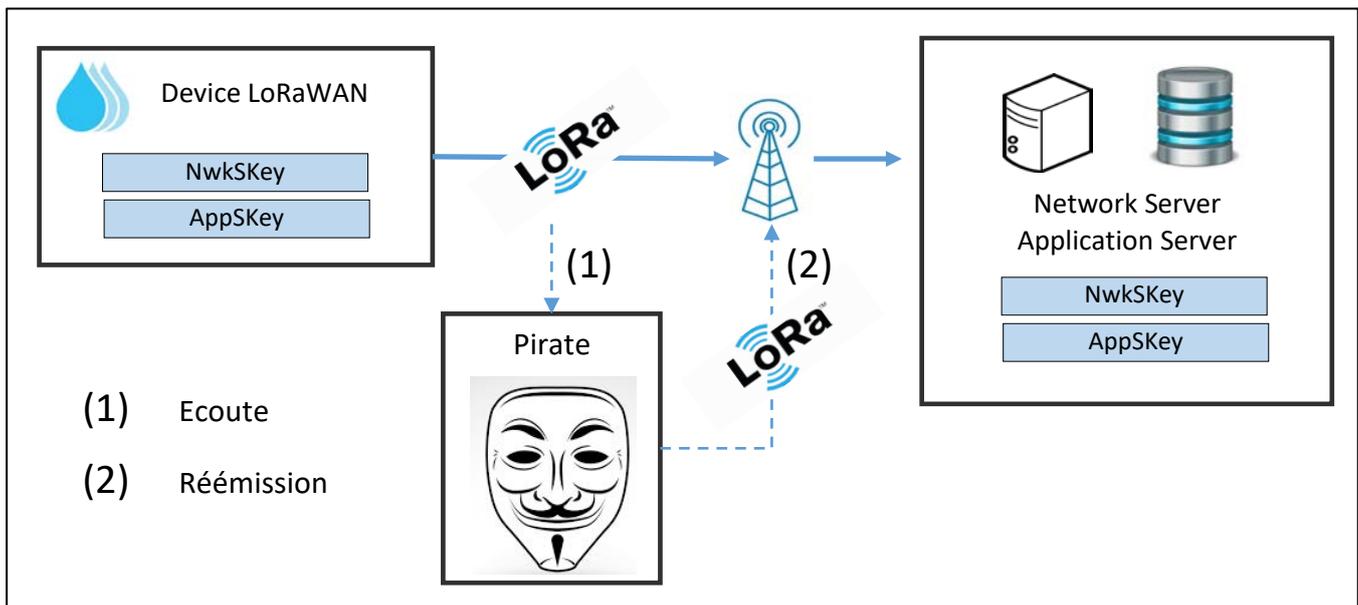


Figure 55: Attaque par REPLAY

Pour éviter cela, la trame LoRaWAN comprend un champ variable appelé **Frame Counter**. Ce nombre s'incrémente à chaque fois qu'une nouvelle trame est transmise par le Device. Le même type de compteur existe du côté du serveur et il s'incrémente à chaque fois qu'il reçoit une trame valide.

- ✔ ■ Le serveur n'accepte une trame que si son Frame Counter est strictement supérieur au dernier Frame Counter valide reçu de ce Device.
- ✘ ■ Si le pirate retransmet la trame telle qu'il l'a enregistrée, le Frame Counter reçu sur le serveur sera inférieur (ou égal) à celui du serveur : le serveur supprime cette trame.

Si le pirate décide de modifier le champ Frame Counter avec une valeur aléatoire, l'authentification échouera car le calcul du champ MIC (avec la Network Session Key) ne sera pas valide.

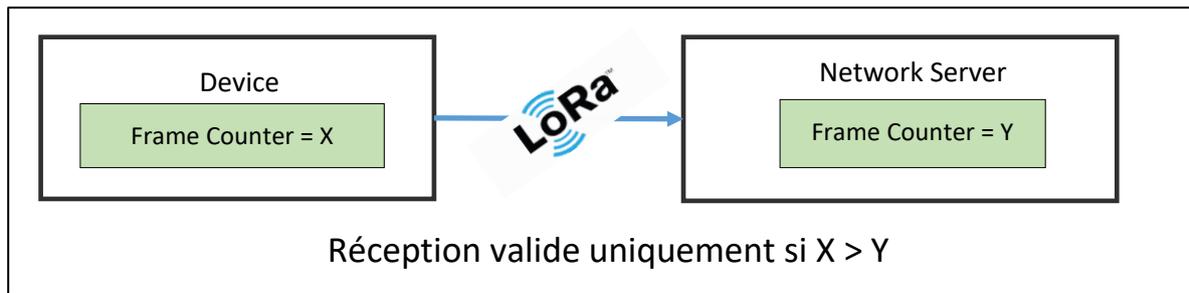


Figure 56: Utilisation d'un "Frame Counter" pour éviter l'attaque par Replay

Le Frame Counter est intéressant pour éviter une attaque par Replay, mais lors du développement d'un Device LoRaWAN, il peut poser problème. En effet, à chaque fois que nous redémarrons le microcontrôleur, notre Frame Counter revient à zéro, alors que le Frame Counter du serveur continue de s'incrémenter. Ce problème peut être résolu de différentes manières :

1. Activer la possibilité de "Réinitialiser le compteur de trames". Sur certains serveurs LoRaWAN, vous avez la possibilité d'accepter un Frame Counter même s'il n'est pas égal ou supérieur à celui du serveur. Bien sûr, vous devez garder à l'esprit que c'est une menace pour la sécurité.



Figure 57: Activation de l'option "Resets Frame Counters" sur TTN



Figure 58: "Frame Counter Validation" option sur Chirpstack

2. Utiliser l'activation OTAA au lieu de ABP. En effet, à chaque Join-Request, le Frame-Counter est remis à zéro sur le Device **et** sur le serveur.
3. Conserver la valeur du Frame-Counter dans une mémoire non volatile et récupérer sa valeur lorsque le Device LoRaWAN redémarre.



La sauvegarde du Frame-Counter dans une mémoire non volatile est obligatoire dans les versions récentes (1.0.4) de la spécification LoRaWAN.



Exercice : Essayez de jouer le rôle du pirate en produisant une attaque par Replay sur votre serveur LoRaWAN.

Solution :

1. Envoyez un message depuis votre Device LoRaWAN avec un Payload simple et vérifiez sa bonne réception sur votre Application Server.
2. Allez sur les logs de votre Gateway et récupérez le PHY Payload. Vous ne pourrez pas le comprendre puisqu'il est chiffré, mais vous le retransmettez tel quel. C'est ce que fait un pirate.

- Désactivez la vérification du Frame Counter sur votre serveur LoRaWAN : Attention, vous êtes vulnérable aux attaques par Replay.
- Programmez votre Device (ou utilisez-en un autre) afin de transmettre une trame LoRa (pas LoRaWAN !) avec le PHY Payload volé.
- Vous devriez voir la donnée transmise par le pirate sur votre Application Server.

Si ce n'est pas le cas, [demandez de l'aide](#) ;-)

4.5.4 Protection contre les attaques par Replay (messages downlink)

Nous avons exactement la même procédure pour les messages en downlink. Il existe deux autres Frame Counter, un côté serveur et un coté Device. Un Device accepte un message downlink que si le Frame Counter reçu est strictement supérieur au dernier Frame Counter valide qu'il a reçu.

Last packets		Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	ESP	SF/DR
↓		2021-11-27 23:45:05.202	0493C363	1		5				SF7
↑	data	2021-11-27 23:45:04.202	0493C363	1	6		0.0	10.75	-0.3508...	SF7
↓		2021-11-27 23:44:57.151	0493C363	1		4				SF7
↑	data	2021-11-27 23:44:56.151	0493C363	1	5		-1.0	9.5	-1.4618...	SF7
↓		2021-11-27 23:44:52.946	0493C363	1		3				SF7
↑	data	2021-11-27 23:44:51.946	0493C363	1	4		0.0	9.75	-0.4372...	SF7
↓		2021-11-27 23:44:35.259	0493C363	1		2				SF9
↑	data	2021-11-27 23:44:34.259	0493C363	1	3		-1.0	11.75	-1.28097	SF9

Figure 59: Frame Counter uplink (vert) et Downlink (rouge)

La Figure 59 représente le Frame Counter uplink et downlink capturé dans le serveur LoRaWAN d'Activity à l'aide de l'outil Wireless-Logger. Nous pouvons clairement voir qu'ils augmentent à chaque trame.

4.5.5 Protection contre les attaques par Replay (Join-Request)

Un pirate peut également utiliser une attaque par Replay pendant la procédure de Join en OTAA. Un compteur dont le nom est "DevNonce" est utilisé dans le même but. Un DevNonce ne peut pas être utilisé deux fois. Depuis la version 1.0.4 de la spécification LoRaWAN, un Device en OTAA doit donc sauvegarder ce nombre dans une mémoire non volatile, sinon le Join-Request ne sera pas accepté lorsque le Device redémarrera et réutilisera les mêmes DevNonce.

4.5.6 Paramètres de communication

Lorsque nous fonctionnons en OTAA, le Device LoRaWAN envoie un Join-Request. Un Join-Accept est renvoyé par le Network Server si le Device a été autorisé à échanger avec le serveur. Nous avons vu au chapitre 4.4.3 que le Join-Accept comprend :

Un champ DLSettings : Le champ **DLSettings** donne des informations sur le Spreading Factor et la bande passante que le Device doit utiliser pour recevoir sur les fenêtres de réception RX1 et RX2.

Un champ RXDelay : Le champ **RXDelay** indique le temps entre la fin de la transmission (uplink) et le début de la fenêtre de réception (downlink). On appelle ce délai RX1.

Un champ CFList : **CFList** indique la liste de tous les canaux disponibles pour ce réseau en plus des canaux 0 à 2 (qui sont obligatoires) : 868,1 MHz, 868,3 MHz et 868,5 MHz. Les autres canaux peuvent être définis dans le champ CFList (canaux 3 à 7). Cela donne un maximum de 8 canaux au total.

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch3	Freq Ch4	Freq Ch5	Freq Ch6	Freq Ch7	CFListType

Figure 60: Canaux 3 à 7 définis dans le champ CFList du Join-Accept

Ces trois paramètres sont présents dans le Join-Accept. Ainsi, un Device fonctionnant en ABP ne peut pas en bénéficier.

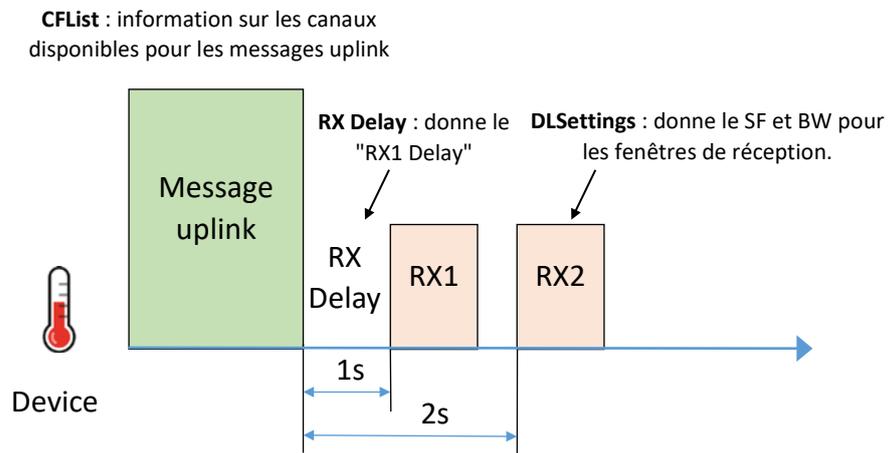


Figure 61: Paramètres DLSettings, RX Delay et CFList

4.5.7 Résumé

Nous pouvons donc résumer les deux méthodes d'activation dans le tableau suivant.

	ABP	OTAA
Sécurité globale	Moins sécurisé que l'OTAA	Plus de sécurité
Gestion du Frame Counter	La sauvegarde en mémoire non volatile est obligatoire. Possibilité de désactiver le contrôle du Frame Counter (faille de sécurité).	Supporté par l'OTAA
Changement de réseau	Complicé et non sécurisé	Supporté par l'OTAA avec un Join Server
Modification RX Delay DLSettings	Gestion par "MAC commands"	Supporté par l'OTAA
Ajout de canaux	Gestion par "MAC commands"	Supporté par l'OTAA

Tableau 14 : Comparaison des méthodes d'activation ABP et OTAA

4.6 Types de trames LoRaWAN

Un Device LoRaWAN peut envoyer et recevoir des trames :

- Uplink : Trame envoyée par le Device
- Downlink : Trame reçue par le Device

Si une collision se produit, ou si la Gateway est en dehors de la zone de couverture, une trame peut ne pas atteindre le serveur LoRaWAN. Il peut être important pour un Device d'avoir une connexion plus fiable. Ainsi, on peut envoyer deux types de trames.

- Unconfirmed : Le serveur LoRaWAN n'envoie pas d'acquittement.
- Confirmed : Le serveur LoRaWAN envoie un acquittement.

Nous avons exactement le même comportement pour les trames en Downlink. Ainsi on peut envoyer deux types de trames :

- Unconfirmed : Le Device n'envoie pas d'acquittement.
- Confirmed : Le Device envoie un acquittement.

Nous pouvons tester ces différentes configurations avec l'outil Wireless-Logger du serveur LoRaWAN d'ACTILITY.

4.6.1 Trame uplink unconfirmed

Dans ce cas, il n'y a aucun moyen de savoir si la trame a atteint le Network Server. C'est pourtant le moyen de communication le plus efficace en terme de consommation et d'occupation du réseau. Sur la Figure 62, on peut voir le Join-Request, le Join-Accept et toutes les trames uplink (FCnt = 1, 2, 3) sans acquittement.

Last packets											
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	ESP	SF/DR
+	↑	data	2021-11-18 00:36:56.788	0493C363	1	3		-12.0	10.5	-12.370...	SF7
+	↑	data	2021-11-18 00:36:39.018	0493C363	1	2		-10.0	9.25	-10.487...	SF7
+	↑	data	2021-11-18 00:35:16.044	0493C363	1	1		-10.0	8.75	-10.543...	SF7
+	↓	join	2021-11-18 00:34:38.339		None						SF12
+	↑	join	2021-11-18 00:34:32.339		None			-11.0	9.75	-11.437...	SF12

Figure 62: Uplink unconfirmed

Si nous détaillons une trame uplink, nous notons que dans l'en-tête MAC le champ MType indique que nous avons à faire à un "UnconfirmedDataUp".

Last packets											
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	ESP	SF/DR
+	↑	data	2021-11-18 00:35:16.044	0493C363	1	1		-10.0	8.75	-10.543...	SF7
<div style="border: 1px solid red; padding: 2px; display: inline-block;">Mtype: UnconfirmedDataUp</div> Flags: ADR : 0, ADRAckReq : 0, ACK : 0											

Figure 63: Détail de la trame uplink unconfirmed

4.6.2 Trame uplink confirmed

Dans ce cas, chaque trame uplink est confirmée par un acquittement en downlink. Sur la Figure 64, nous pouvons voir le Join-Request, le Join-Accept, puis chaque uplink (FCnt = 1, 2) est suivi de son acquittement en downlink (NFCnt = 0, 1).

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↓		2021-11-17 23:51:32.038	0493C363	1		1			SF7
+	↑	data	2021-11-17 23:51:31.038	0493C363	1	2		-12.0	9.75	SF7
+	↓		2021-11-17 23:50:57.081	0493C363	1		0			SF7
+	↑	data	2021-11-17 23:50:56.081	0493C363	1	1		-11.0	9.25	SF7
+	↓	join	2021-11-17 23:50:12.593		None					SF12
+	↑	join	2021-11-17 23:50:06.593		None			-11.0	10.0	SF12

Figure 64: Uplink confirmed et acquittement

Si nous détaillons la trame uplink (FCnt = 2) nous notons que dans l'en-tête MAC, le champ MType indique que nous avons à faire à un "ConfirmedDataUp".

			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↑	data	2021-11-17 23:51:31.038	0493C363	1	2		-12.0	9.75	SF7
Mtype: ConfirmedDataUp Flags: ADR : 0, ADRAckReq : 0, ACK : 0										

Figure 65: Détail de la trame uplink confirmed

On peut voir que le serveur LoRaWAN envoie un ACK dans une trame downlink (NFCnt = 1) juste après l'uplink.

			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↓		2021-11-17 23:51:32.038	0493C363	1		1			SF7
Mtype: UnconfirmedDataDown Requested RX1/RX2Delay: 1000 Flags: ADR : 0, ADRAckReq : 0, ACK : 1, FPending : 0										

Figure 66: Acquittement avec le flag ACK

4.6.3 Trame downlink unconfirmed

Dans ce cas, il n'y a aucun moyen de savoir si la trame a atteint le Device. Sur la Figure 67, nous pouvons voir le Join-Request, le Join-Accept puis l'uplink unconfirmed (FCnt = 1). A ce moment, le serveur programme un nouveau downlink unconfirmed. Mais en classe A, un downlink (NFCnt = 0) ne peut être envoyé qu'après la réception d'un nouvel uplink (FCnt = 2).

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↓	data	2021-11-18 21:52:53.191	0493C363	1		0			SF7
+	↑	data	2021-11-18 21:52:52.191	0493C363	1	2		-5.0	9.25	SF7
+	↑	data	2021-11-18 21:52:42.210	0493C363	1	1		-7.0	9.75	SF7
+	↓	join	2021-11-18 21:52:36.004		None					SF12
+	↑	join	2021-11-18 21:52:30.004		None			-3.0	11.0	SF12

Figure 67: Downlink unconfirmed

Si nous détaillons la trame downlink (NFCnt = 0), nous notons que dans l'en-tête MAC, le champ MType indique que nous avons à faire à un "UnconfirmedDataDown".

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
⊞	↓	data	2021-11-18 21:52:53.191	0493C363	1		0			SF7
Mtype: UnconfirmedDataDown Requested RX1/RX2Delay: 1000 Flags: ADR : 0, ADRAckReq : 0, ACK : 0, FPending : 0										

Figure 68: Trame downlink unconfirmed

Lorsque le serveur a plusieurs trames à transmettre au Device, cela peut prendre très longtemps car il faut attendre à chaque fois que le Device émette pour pouvoir lui envoyer une trame downlink. Afin d'accélérer ce processus, il est possible de dire au Device que d'autres trames sont en attente en utilisant un Flag spécifique (FPending). Il appartiendra au Device de décider s'il souhaite accélérer la transmission de ces trames, mais en toutes circonstances, il devra à nouveau transmettre pour réceptionner les trames downlink du serveur.

Pour mettre en valeur ce comportement, nous lançons le même scénario que précédemment à la Figure 69, : Join-Request, Join-Accept, puis uplink unconfirmed (FCnt = 1). Ensuite, nous programmons plusieurs downlink qui sont en file d'attente sur le Network Server. Puis nous attendons un nouvel uplink (FCnt = 2) pour envoyer le downlink (NFCnt = 0).

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
⊞	↓	data	2021-11-18 22:04:56.161	0493C363	1		0			SF7
⊞	↑	data	2021-11-18 22:04:55.161	0493C363	1	2		-6.0	9.0	SF7
⊞	↑	data	2021-11-18 22:04:38.920	0493C363	1	1		-6.0	9.75	SF7
⊞	↓	join	2021-11-18 22:04:35.864		None					SF12
⊞	↑	join	2021-11-18 22:04:29.864		None			-4.0	11.5	SF12

Figure 69: Scénario de "Frame Pending" sur le Network Server

Les données en downlink ne peuvent être transmises que "une par une". Donc lorsque le premier message est transmis (NFCnt = 0), le deuxième message est toujours en attente sur le Network Server. On peut voir que le flag FPending est activé.

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
⊞	↓	data	2021-11-18 22:04:56.161	0493C363	1		0			SF7
Mtype: UnconfirmedDataDown Requested RX1/RX2Delay: 1000 Flags: ADR : 0, ADRAckReq : 0, ACK : 0, FPending : 1										

Figure 70: Le flag FPending

4.6.4 Trame downlink confirmed

Dans ce cas, toutes les trames downlink sont confirmées. L'acquiescement est effectué dans la trame uplink suivante avec le bit ACK activé. Sur la Figure 71 nous pouvons voir le Join-Request, le Join-Accept, et un uplink (FCnt = 1). Ensuite, un downlink confirmed est programmé. Lorsque l'uplink suivante arrive (FCnt = 2), le downlink confirmed est transmis (NFCnt = 0). L'uplink suivant (FCnt = 3) transmet l'acquiescement.

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↑	data	2021-11-18 22:29:13.144	0493C363	1	3		-5.0	9.5	SF7
+	↓	data	2021-11-18 22:28:41.462	0493C363	1		0			SF7
+	↑	data	2021-11-18 22:28:40.462	0493C363	1	2		-6.0	9.0	SF7
+	↑	data	2021-11-18 22:28:26.073	0493C363	1	1		-7.0	10.0	SF7
+	↓	join	2021-11-18 22:28:19.768		None					SF12
+	↑	join	2021-11-18 22:28:13.768		None			-2.0	11.25	SF12

Figure 71: Downlink confirmed

Si nous détaillons la trame downlink (NFCnt = 0), nous notons que dans l'en-tête MAC, le champ MType indique que nous avons à faire à un "ConfirmedDataDown".

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
-	↓	data	2021-11-18 22:28:41.462	0493C363	1		0			SF7
Mtype: ConfirmedDataDown Requested RX1/RX2Delay: 1000 Flags: ADR : 0, ADRAckReq : 0, ACK : 0, FPending : 0										

Figure 72: Trame downlink confirmed

Nous voyons aussi que la trame uplink suivante (FCnt = 3) a son Flag ACK à 1.

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
-	↑	data	2021-11-18 22:29:13.144	0493C363	1	3		-5.0	9.5	SF7
Mtype: UnconfirmedDataUp Flags: ADR : 0, ADRAckReq : 0, ACK : 1										

Figure 73: Acquiescement avec le Flag ACK

4.6.5 Uplink confirmed et downlink confirmed

Nous pouvons avoir une transmission confirmée dans les deux sens. Voici le scénario :

Last packets										
			Local Timestamp	DevAddr	FPort	FCnt ↑	NFCnt ↓	RSSI	SNR	SF/DR
+	↓		2021-11-18 22:57:37.208	0493C363	1		2			SF7
+	↑	data	2021-11-18 22:57:36.208	0493C363	1	3		-6.0	9.75	SF7
+	↓	data	2021-11-18 22:57:32.041	0493C363	1		1			SF7
+	↑	data	2021-11-18 22:57:31.041	0493C363	1	2		-5.0	8.75	SF7

Figure 74: Uplink confirmed et downlink confirmed

- FCnt = 2 ConfirmedDataUp Uplink data
- NFCnt = 1 ConfirmedDataDown Downlink data + Flag ACK pour confirmer FCnt =2
- FCnt = 3 ConfirmedDataUp Uplink data + Flag ACK pour confirmer NFCnt =1
- NFCnt = 2 ConfirmedDataDown No Data + Flag ACK pour confirmer FCnt =3

Même si le réseau LoRaWAN offre de telles possibilités, nous devons être conscients que les capacités downlink de ces réseaux sont limitées. Une communication LoRaWAN devrait limiter autant que possible le flux descendant (c'est-à-dire la confirmation des messages).

4.7 Les commandes MAC

Pour l'administration du réseau LoRaWAN, un ensemble de commandes MAC peut être échangé entre le Network Server et le Device. Ces commandes appartiennent à la couche LoRaWAN MAC et ne doivent jamais atteindre l'application utilisateur car elles servent uniquement au bon fonctionnement du protocole et non pas de l'application.

- L'Application Server ne recevra jamais ces commandes.
- L'application utilisateur du Device ne recevra jamais ces commandes.

Chaque commande MAC possède un identifiant spécifique appelé CID (**Command IDentifier**).

CID	Commande	Transmis par		Description
		Device	Network Server	
0x02	LinkCheckReq	X		Utilisé par un Device pour valider sa connectivité à un réseau.
0x02	LinkCheckAns		X	Acquittement de LinkCheckReq .
0x03	LinkADRReq		X	Demande au Device de modifier le SF, la puissance et le nombre de retransmission dans le cadre de l'ADR (Adaptive Data Rate)
0x03	LinkADRAns	X		Acquittement de LinkADRReq
0x04	DutyCycleReq		X	Définit le Duty Cycle maximum autorisé pour le Device.
0x04	DutyCycleAns	X		Acquittement de DutyCycleReq .
0x05	RXParamSetupReq		X	Définit les paramètres de RX1 et RX2.
0x05	RXParamSetupAns	X		Acquittement de RXParamSetupReq .
0x06	DevStatusReq		X	Demande l'état du Device.
0x06	DevStatusAns	X		Acquittement de DevStatusAns avec l'état du Device (niveau de batterie et l'état de sa radio)
0x07	NewChannelReq		X	Demande de création ou de modification d'un nouveau canal radio.
0x07	NewChannelAns	X		Acquittement de NewChannelReq .
0x08	RXTimingSetupReq		X	Définit les temps de RX1 et RX2.
0x08	RXTimingSetupAns	X		Acquittement de RXTimingSetupReq .
0x09	TXParamSetupReq		X	Utilisé par un Network Server pour définir la durée maximale d'émission (dwell time) et la puissance maximale (MaxEIRP) en fonction des réglementations locales.
0x09	TXParamSetupAns	X		Acquittement de TXParamSetupReq .
0x0A	DIChannelReq		X	Modifie le canal de réception RX1 par offset de la fréquence utilisée en uplink.
0x0A	DIChannelAns	X		Acquittement de DIChannelReq .
0x0D	DeviceTimeReq	X		Utilisé par un Device pour demander l'heure GPS actuelle.
0x0D	DeviceTimeAns		X	Acquittement de DeviceTimeReq

Tableau 15 : Commandes MAC LoRaWAN (source : LoRa Alliance - LoRaWAN 1.0.4)

4.8 Data rate, canaux et puissance

4.8.1 Data Rate (DR)

Comme nous l'avons vu précédemment, le Spreading Factor (SF) et la bande passante (BW) sont les deux paramètres qui ont un impact sur le débit binaire.

$$\text{Bit Rate (bit/s)} = SF \cdot \frac{\text{Bandwidth}}{2^{SF}}$$

La combinaison de SF et BW est appelée DR (Data Rate). Dans la bande EU868, il est normalisé de DR0 à DR6.

Data rate	Spreading Factor	Bande passante
DR 0	SF12	125 KHz
DR 1	SF11	125 KHz
DR 2	SF10	125 KHz
DR 3	SF9	125 KHz
DR 4	SF8	125 KHz
DR 5	SF7	125 KHz
DR 6	SF7	250 KHz

Tableau 16 : Dénomination des DR en fonction de SF et de la bande passante

4.8.2 Adaptive Data Rate (ADR)

Le réglage du SF et du P_T (puissance transmise) n'est pas simple. Même si l'on trouve une bonne configuration, la transmission peut être altérée par l'environnement local ou la météo. Pour surmonter cette difficulté, une méthode d'ajustement automatique a été mise en place par le protocole LoRaWAN : il s'agit de l'Adaptive Data Rate (**ADR**). L'idée est de laisser le Network Server calculer la meilleure combinaison de SF / P_T .

Pour effectuer cette opération, le Network Server vérifie :

- Le **RSSI** (puissance reçue sur la Gateway) : Le RSSI doit être supérieur à la sensibilité du récepteur après application d'une marge.
- Le **SNR** de la transmission : Le SNR de la transmission doit être supérieur au SNR minimum accepté par le récepteur après application d'une marge.
- L'estimation de la perte de paquets : en vérifiant le Frame Counter, le Network Server peut estimer le nombre de paquets perdus lors des dernières transmissions.

L'algorithme ADR n'est pas défini par la spécification LoRaWAN. Chaque Network Server peut utiliser sa propre stratégie. Le diagramme ci-dessous montre un algorithme très simple de l'ADR.

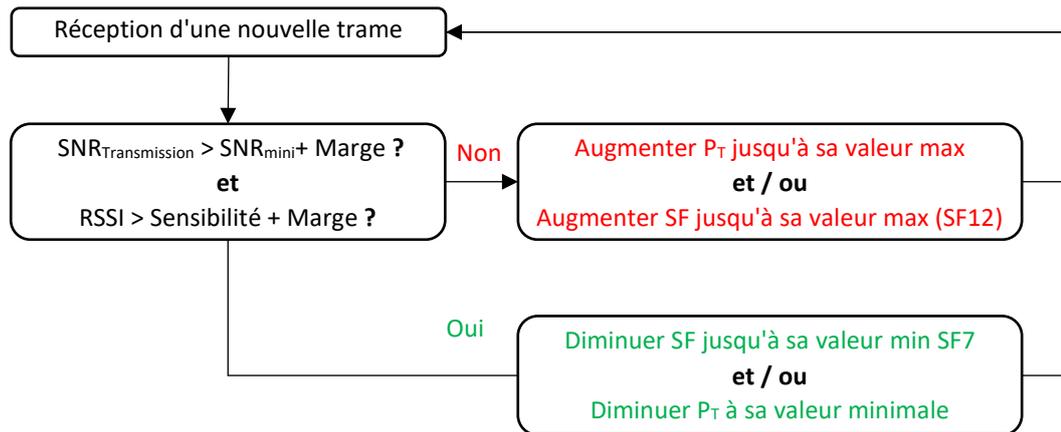


Figure 75: Exemple d'un algorithme ADR



Un Device LoRaWAN doit activer le mode ADR (Flag ADR) pour utiliser cette fonctionnalité.

Les commandes ADR font partie des commandes MAC LoRaWAN (voir chapitre 4.7).

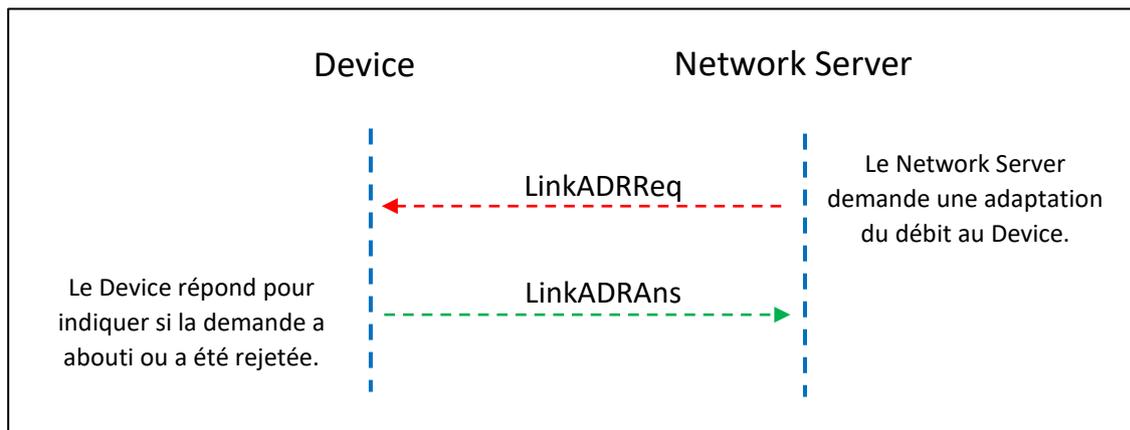


Figure 76: Commandes MAC Adaptive Data Rate

La commande MAC **LinkADRReq** n'est pas envoyée dans une trame séparée. Elle est intégrée dans un message de données downlink.



Le piggybacking est un moyen de profiter d'un transfert de données pour y inclure une commande, un acquittement ou toute administration du réseau.

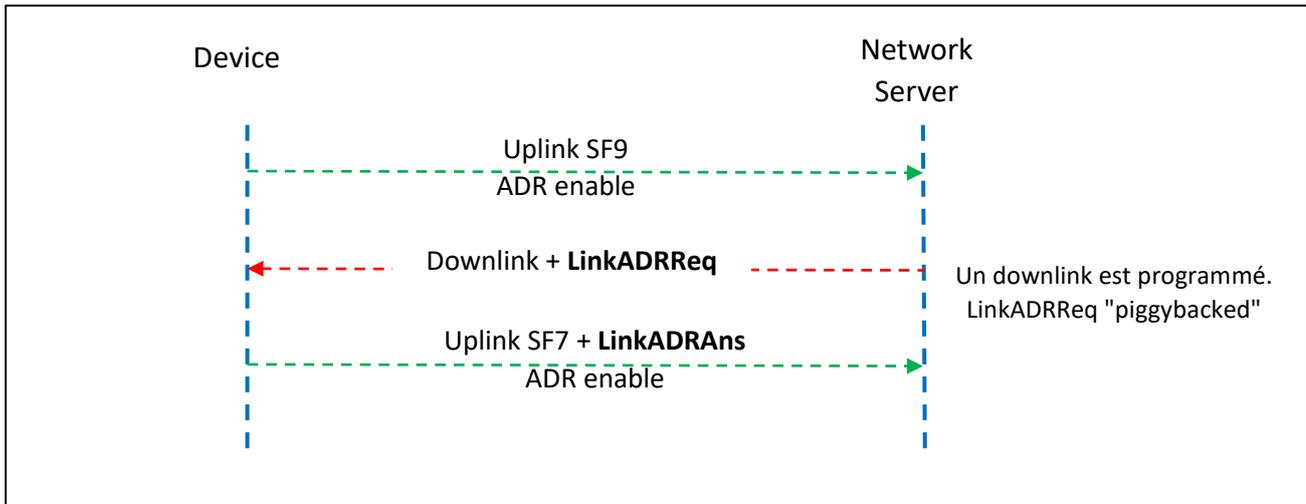


Figure 77: Commandes MAC Adaptive Data Rate

Avec le Wireless-Logger du serveur LoRaWAN d'Actility, nous mettons en place le scénario suivant : un uplink confirmé est envoyé avec SF9 et ADR activé (FCnt = 5). Le Network Server utilise l'ACK pour transmettre la commande MAC **LinkADRReq** (NFCnt = 7). L'uplink suivant envoie la commande MAC **LinkADRAns** pour confirmer le processus d'optimisation (FCnt = 6).

Last packets							
			Local Timestamp	FPort	SF/DR	FCnt ↑	NFCnt ↓
+	↑	mac data	2021-11-28 18:46:20.008	1	SF7	6	
+	↓	mac	2021-11-28 18:45:59.743	0	SF9		7
+	↑	data	2021-11-28 18:45:58.743	1	SF9	5	

Figure 78: Optimisation du Data Rate

Si nous détaillons la trame uplink confirmé, nous notons que dans l'entête MAC, le Flag ADR est activé (FCnt = 5) pour signaler que le Device souhaite utiliser l'ADR.

Last packets							
			Local Timestamp	FPort	SF/DR	FCnt ↑	NFCnt ↓
+	↑	data	2021-11-28 18:45:58.743	1	SF9	5	
Mtype: ConfirmedDataUp							
Flags: ADR : 1, ADRAckReq : 0, ACK : 0							

Figure 79: Flag ADR

Le Network Server utilise l'ACK pour envoyer la commande MAC **LinkADRReq** (NFCnt = 7). Une nouvelle puissance et un nouveau Data Rate (DR5 = SF7) sont proposés.

Last packets							
			Local Timestamp	FPort	SF/DR	FCnt ↑	NFCnt ↓
⊟	↓	mac	2021-11-28 18:45:59.743	0	SF9		7
Mtype: UnconfirmedDataDown Requested RX1/RX2Delay: 1000 Flags: ADR : 0, ADRAckReq : 0, ACK : 1, FPending : 0 Mac (hex): 0351ff0001 MAC.Command.LinkADRReq MAC.LinkADRReq.DataRate.TXPower : 0x51 MAC.LinkADRReq.DataRate.TXPower.DataRate : 5 MAC.LinkADRReq.DataRate.TXPower.TXPower : 1 MAC.LinkADRReq.ChMask : 0x00ff MAC.LinkADRReq.Redundancy : 0x01 MAC.LinkADRReq.Redundancy.ChMaskCnt1 : 0 MAC.LinkADRReq.Redundancy.NbTrans : 1							

Figure 80: Commande MAC LinkADRReq

L'uplink suivant envoie la commande MAC **LinkADRAns** pour confirmer le processus d'optimisation (FCnt = 6). Le Spreading Factor est maintenant SF7.

Last packets							
			Local Timestamp	FPort	SF/DR	FCnt ↑	NFCnt ↓
⊟	↑	mac data	2021-11-28 18:46:20.008	1	SF7	6	
Mtype: ConfirmedDataUp Flags: ADR : 1, ADRAckReq : 0, ACK : 0 Mac (hex): 0307 MAC.Command.LinkADRAns MAC.LinkADRAns.Status : 0x07 MAC.LinkADRAns.Status.ChannelMaskAck : 1 MAC.LinkADRAns.Status.DataRateAck : 1 MAC.LinkADRAns.Status.PowerAck : 1							

Figure 81: Commande LinkADRAns

Un problème survient lorsqu'il n'y a pas de trame downlink disponible pour ajouter la commande **LinkADRReq**. Dans ce cas, le processus d'optimisation ne se produit jamais. Pour éviter cette situation, lorsque le Device n'a pas de nouvelles du Network Server pendant un certain temps, il peut forcer une demande d'optimisation. Ceci est fait grâce au bit **ADRAckReq**.

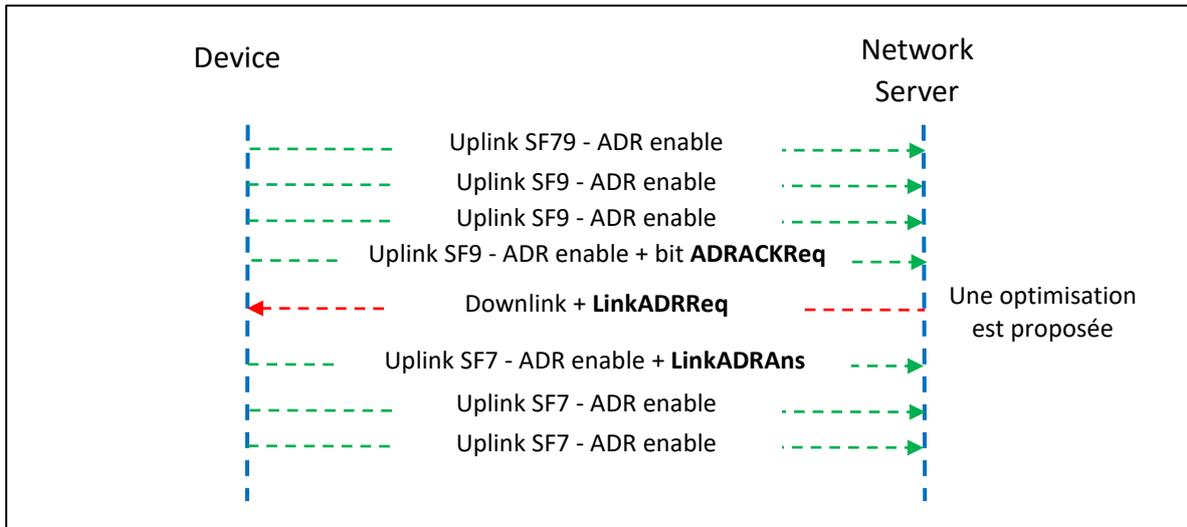


Figure 82: Bit **ADRACKReq** pour demander l'optimisation

Dans le scénario suivant, nous envoyons uniquement en uplink unconfirmed. La trame uplink (FCnt = 77) possède le bit **ADRACKReq**, et nous pouvons constater que le Network Server donne alors un **LinkADRReq**. Dans ce cas, il n'y avait pas d'optimisation à faire car nous étions déjà avec le meilleur SF7 et la puissance P_T minimale.

Last packets							
			Local Timestamp	FPort	SF/DR	FCnt ↑	NFCnt ↓
+	↑	data	2021-11-28 22:40:57.470	1	SF7	80	
+	↑	data	2021-11-28 22:40:47.449	1	SF7	79	
+	↓		2021-11-28 22:40:44.294	1	SF7		2
+	↑	data	2021-11-28 22:40:43.294	1	SF7	77	
+	↑	data	2021-11-28 22:40:33.283	1	SF7	76	

Figure 83: Bit **ADRACKReq** pour lancer un **LinkADRReq**

Si le Device n'obtient pas de réponse alors qu'il a forcé une demande d'optimisation grâce au bit **ADRACKReq**, alors la communication avec le Network Server est considérée comme perdue et le Device LoRaWAN augmente SF / P_T jusqu'à ce qu'il puisse le rejoindre à nouveau.



Les vidéos de démonstration sont disponibles sur notre site web : www.univ-smb.fr/lorawan .



L'algorithme ADR peut également proposer au Device un nombre de retransmission. Ainsi chaque uplink sera envoyé **NbTrans** fois. Le Device reçoit l'information de **NbTrans** dans la trame **LinkADRReq** comme vous pouvez le voir sur la Figure 80.

4.8.3 Les canaux

Le LoRa utilise différentes bandes de fréquence dans différentes parties du monde. En Europe, la bande utilisée est 868 MHz [De 863 MHz à 870 MHz]. Parmi cette bande, le serveur LoRaWAN définit un plan avec un nombre de canaux et Data Rate à utiliser en uplink et downlink. Un Device LoRaWAN **doit** connaître au moins les trois canaux suivants : 868,1 MHz, 868,3 MHz et 868,5 MHz de DR0 à DR5. Les autres canaux dépendent du serveur LoRaWAN. Le Tableau 17 représente les canaux obligatoires.

	Canaux	Data rate	Direction
Obligatoire	868,1 MHz	DR0 à DR5	Uplink / Downlink
	868,3 MHz	DR0 à DR5	Uplink / Downlink
	868,5 MHz	DR0 à DR5	Uplink / Downlink

Tableau 17 : Plan de fréquence LoRaWAN obligatoire

En plus de ce plan de fréquences obligatoire, le Network Serveur est libre de proposer d'autres canaux et DR au Device pendant la procédure de Join (OTAA). Lors de l'utilisation de l'ABP, le Device respectera uniquement les fréquences obligatoires, ce qui limitera ses capacités, sinon l'utilisateur devra les ajouter manuellement ou via des "MAC Commands" supplémentaires. Un certain nombre de plan de fréquence sont prédéfinis :

	Canaux	Data Rate	Direction
Frequency plan Default EU868	867,1 MHz	DR0 à DR5	Uplink / Downlink
	867,3 MHz	DR0 à DR5	Uplink / Downlink
	867,5 MHz	DR0 à DR5	Uplink / Downlink
	867,7 MHz	DR0 à DR5	Uplink / Downlink
	867,9 MHz	DR0 à DR5	Uplink/ Downlink
	869.525 MHz	DR0	Downlink – RX2

Tableau 18 : Plan de fréquence par défaut EU868

	Canaux	Data Rate	Direction
Other Frequency plan (TTN, Actility...)	867,1 MHz	DR0 à DR5	Uplink / Downlink
	867,3 MHz	DR0 à DR5	Uplink / Downlink
	867,5 MHz	DR0 à DR5	Uplink / Downlink
	867,7 MHz	DR0 à DR5	Uplink / Downlink
	867,9 MHz	DR0 à DR5	Uplink/ Downlink
	869.525 MHz	DR3	Downlink

Tableau 19 : Autre plan de fréquence avec DR3 pour RX2

Enfin, si nous souhaitons rendre les réseaux interopérables avec comme objectif de faire du Roaming (faire fonctionner des Devices sur un autre réseau que celui sur lequel est enregistré notre Device), il est important que le plan de fréquence soit commun. La LoRa Alliance a donc préconisé les canaux suivants :

	Canaux	Data Rate	Direction
Roaming Frequency plan	867,1 MHz	DR0 à DR5	Uplink / Downlink
	867,3 MHz	DR0 à DR5	Uplink / Downlink
	867,9 MHz	DR0 à DR5	Uplink/ Downlink

Tableau 20: Plan de fréquence pour le Roaming



Une Gateway LoRaWAN doit être correctement configurée avec le plan de fréquences du serveur LoRaWAN auquel elle est connectée. La configuration du Device se fait pendant le Join Accept en OTAA.

4.8.4 La puissance consommée par le Device

La consommation électrique d'un Device LoRaWAN est directement liée à deux paramètres de la transmission LoRa :

- Le temps d'émission (Time on Air) : plus le message est long, plus la radio sera alimentée longtemps.
- La puissance transmise P_T : plus la puissance est importante, plus le Device consomme de l'énergie.

Évidemment, le premier conseil est de réduire la puissance. La conséquence directe est que le Device peut ne plus atteindre les Gateways. La réduction de la puissance ne peut se faire que si nous avons une marge suffisante entre la puissance reçue par la GW et la sensibilité de la Gateway.

Le deuxième conseil consiste à réduire le temps d'émission. Une solution simple est de réduire le SF (Spreading Factor). En effet, lorsque le SF est réduit de 1, le temps d'émission est divisé par deux (voir chapitre 3.1.2). Mais réduire le SF a une autre conséquence du côté de la Gateway : cela réduit drastiquement sa sensibilité ainsi que sa capacité à détecter le signal parmi le bruit.

A titre d'exemple, le Tableau 20 liste plusieurs transmissions avec différents SF utilisés sur un émetteur. D'autre part, nous calculons sur le récepteur pour chaque transmission la meilleure sensibilité que nous pouvons atteindre, ainsi que la plus basse valeur du SNR acceptable.

Émetteur Spreading Factor	Récepteur	
	Sensibilité	SNR minimum
7	-123 dBm	7,5 dB
8	-126 dBm	10 dB
9	-129 dBm	12,5 dB
10	-132 dBm	15 dB
11	-134,5 dBm	17,5 dB
12	-137 dBm	20 dB

Tableau 21 : SF, Sensibilité et SNR

Remarque : Le Tableau 20 est un recueil de données provenant de la documentation du SX1261 (SNR) et de calculs effectués par le LoRa Calculator (Sensibilité).

Ainsi, réduire le SF et le P_T entraînera une diminution de la portée et doit donc être ajustés de manière cohérente. Le choix de SF et P_T n'est pas simple et nous utilisons souvent une méthode empirique pour déterminer leur valeur en vérifiant le SNR et le RSSI sur la Gateway. L'autre choix est bien sûr d'utiliser l'ADR.

5 Les Réseaux LoRaWAN et les serveurs LoRaWAN

5.1 Les différents types de réseaux

Nous avons le choix entre mettre en place toute l'infrastructure du réseau ou nous en remettre à un opérateur. Il existe trois possibilités pour l'architecture LoRaWAN.

- Nous pouvons faire appel à des opérateurs publics qui disposent de réseaux opérationnels à l'échelle nationale.
- Nous pouvons construire notre propre réseau LoRaWAN privé.
- Nous pouvons construire un réseau hybride en ne mettant en place qu'une partie de l'infrastructure.

5.1.1 Les réseaux LoRaWAN d'opérateurs publics

Les opérateurs publics proposent leur réseau LoRaWAN à l'échelle nationale pour connecter les Devices IoT. Objenious (une filiale de Bouygues), Orange, KPN, Proximus en sont quelques exemples. Ils ont généralement une excellente couverture. Dans cette situation, l'utilisateur doit simplement s'occuper de ses Devices LoRaWAN et de l'application utilisateur (plateforme IoT). Les opérateurs publics gèrent les Gateways et le serveur LoRaWAN.

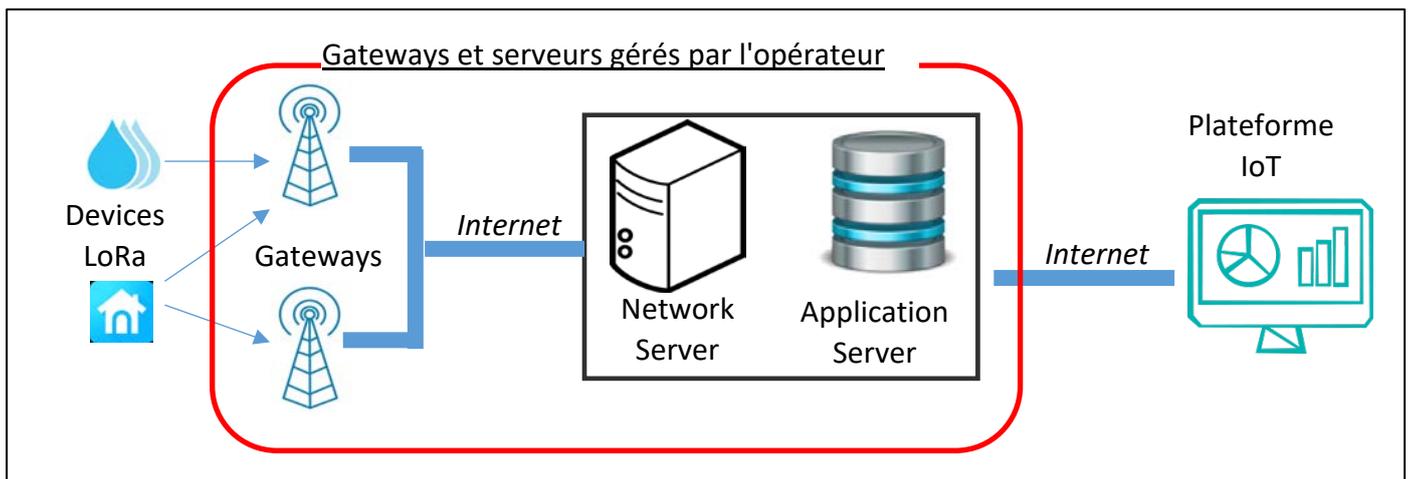


Figure 84: Infrastructure d'un réseau d'opérateur LoRaWAN

L'utilisateur souscrit à un ou plusieurs forfaits pour pouvoir connecter sa flotte de Device. A titre d'exemple, voici les abonnements proposés par Objenious et Orange en 2022 pour avoir accès à leur réseau LoRaWAN :

Orange :

- Uplink illimité (dans le respect du Duty-cycle).
- Le prix de chaque message downlink est de 5 cts.
- L'abonnement varie de 1€/mois (36 mois) à 2€/mois (sans engagement).



Bouygues Objenious :

- 144 messages uplink par jour.
- 6 messages downlink par jour.
- L'abonnement est de 20 € / an



La démarche de l'utilisateur pour connecter les Devices LoRaWAN est simple :

1. Souscrire à un abonnement.
2. Déclarer les Devices sur la plateforme de l'opérateur ("Live Object" pour Orange, "Spot" pour Objenious).
3. Activer les Devices.
4. Récupérer les données sur la plateforme de l'opérateur.
5. Rediriger vos données vers une plateforme IoT.

Il existe 156 opérateurs LoRaWAN dans 171 pays.

5.1.2 Les réseaux LoRaWAN privés

Chacun est libre de créer son propre réseau privé. Vous devrez mettre en place votre propre Gateway et votre propre infrastructure de serveurs pour communiquer avec vos Devices. Vous devrez également vous occuper de l'administration de votre serveur LoRaWAN.

Dans certaines Gateway, une instance d'un serveur LoRaWAN est proposée. Cela simplifie l'infrastructure globale car vous avez tout dans un seul boîtier (la Gateway) mais cela limitera considérablement le potentiel de votre réseau. Sur la Figure 85 nous pouvons voir que la Gateway comprend le Network Server et L'Application Server (ligne continue verte). La plateforme IoT est parfois également incluse (ligne verte en pointillés).

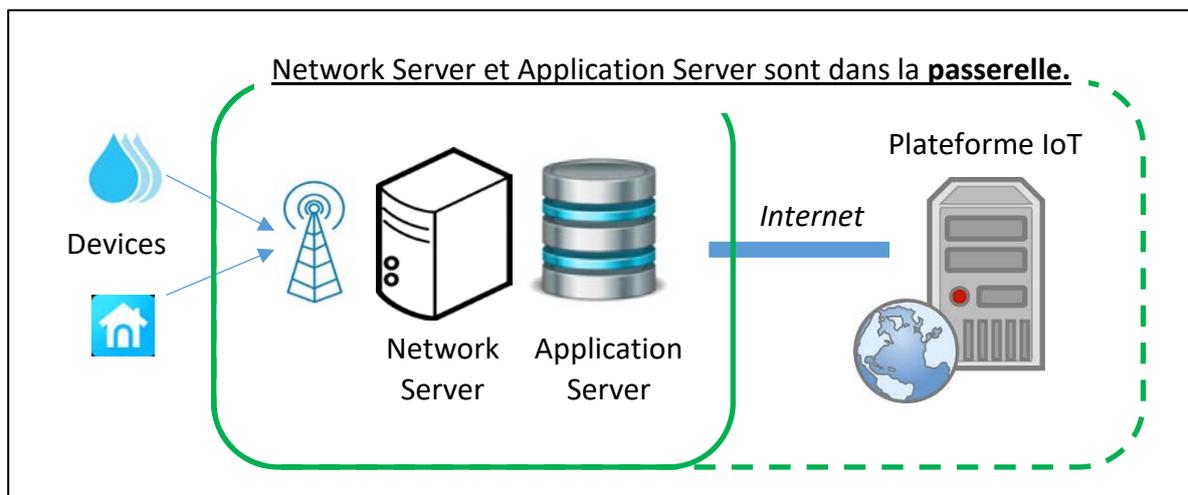


Figure 85: Réseau privé regroupé dans une seule Gateway

Voici un exemple d'un réseau LoRaWAN privé embarqué dans une Gateway :

- Kerlink : Wanesy SPN (Small Private Network)

À moins de développer votre propre serveur LoRaWAN (ce qui représente un travail énorme), vous devrez acheter ce logiciel. Le coût dépend des fonctionnalités, du service et du nombre de Gateways/Device que vous souhaitez enregistrer. Une fois que vous aurez acheté la licence, vous serez autorisé à installer le serveur LoRaWAN sur votre propre serveur. C'est ce qu'on appelle une "licence sur site" ou "on premises licence". En voici quelques-unes :

- Actility : [Actility on premises](#) 
- Kerlink : [Wanasy WMC](#) on premises (**Wanasy Management Center**) 
- The Things Industries : [The Things Stack Enterprise](#) (on premises) 
- ResIOT : [Network Server ResIOT](#) (on premises) 

Une autre possibilité pour mettre en place un réseau privé est d'utiliser un serveur LoRaWAN gratuit et open source dont les plus connus sont :

- The Things Network : [The Things Stack](#). 
- Chirpstack : [LoRaWAN® open-source de ChirpStack](#) 

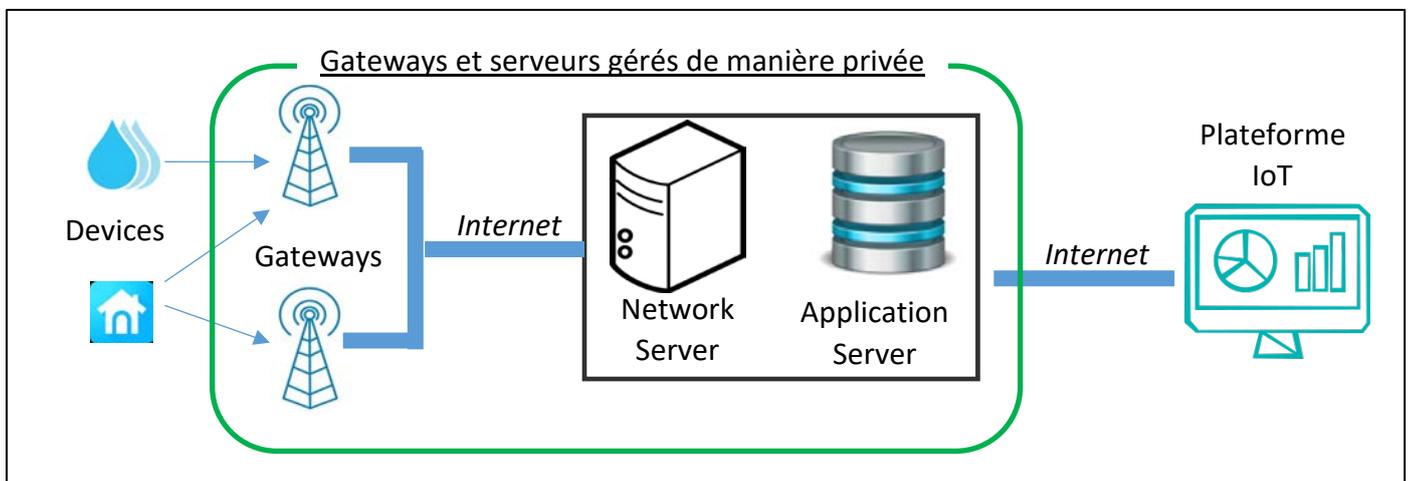


Figure 86: Infrastructure d'un réseau LoRaWAN privé

La démarche de l'utilisateur pour connecter les Devices LoRaWAN à un réseau privé LoRaWAN requiert des compétences. Il faut :

1. Acheter une ou plusieurs Gateways.
2. Les déployer sur site.
3. Acheter un serveur LoRaWAN (ou utiliser un serveur gratuit).
4. Installer le serveur LoRaWAN sur votre propre infrastructure.
5. Enregistrer les Devices sur votre serveur LoRaWAN.
6. Activer les Devices.
7. Récupérer les données sur votre serveur.
8. Rediriger vos données vers une plateforme IoT.

5.1.3 Choix du type de réseau : opéré ou privé?

Nous pouvons résumer les avantages et les inconvénients de chacun de ces types de réseau dans le Tableau 21.

	Réseau privé	Réseau exploité
Coût de l'abonnement	Pas d'abonnement	Environ 1,5 € / mois par Device LoRaWAN
Coûts d'infrastructure	Investissement important au début (Gateways et Serveurs)	Inclus dans l'abonnement
Compétences requises	Requiert des compétences pour l'installation, l'administration et la maintenance.	Tout est géré par l'opérateur
Couverture	Optimisé en fonction des besoins	Dépend de l'opérateur choisi Possibilité d'itinérance entre opérateurs
Uplink	Illimité dans le respect du Duty-cycle	Limité selon l'abonnement
Downlink	Illimité dans le respect du Duty-cycle	Nombre limité ou paiement en fonction du nombre

Tableau 22 : Avantages et inconvénients des réseaux privés et publics

5.1.4 Une alternative, le réseau LoRaWAN hybride

Dans le cas où aucune des solutions précédentes ne convienne, il existe une autre possibilité qui est un intermédiaire entre le réseau public et le réseau privé. Elle présente l'avantage de gérer la couverture du réseau en utilisant ses propres Gateways, tout en confiant l'infrastructure du serveur LoRaWAN à un fournisseur de service afin de limiter les investissements et la maintenance.

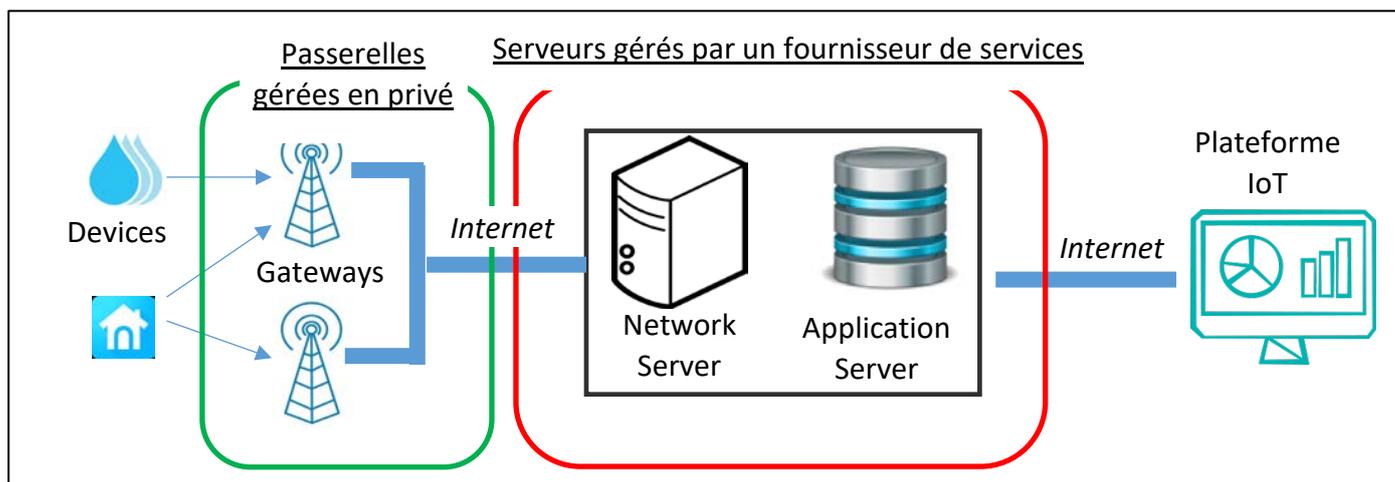


Figure 87 : Infrastructure d'un réseau hybride

Vous devrez acheter un hébergement cloud pour le serveur LoRaWAN. Voici quelques exemples:

- Actility : [ThingPark Enterprise](#) 
- Kerlink : [Wanasy Management as a Service](#) 
- The Things Industries : [The Things Stack Enterprise \(Cloud\)](#) 
- Loriot : [Loriot Network Server \(Cloud\)](#) 
- ResIOT : [ResIOT Network Server \(Cloud\)](#) 

Sur chaque serveur LoRaWAN hébergé sur un cloud, il y a très souvent une inscription gratuite mais limité à quelques Gateways et quelques Devices.

- Actility: [ThingPark Community](#)
- The Things Network: [The Things Stack community Edition](#)
- Et bien d'autres encore...



Le parcours de l'utilisateur pour connecter des Devices LoRaWAN à un réseau hybride LoRaWAN est le suivant :

1. Acheter une ou plusieurs Gateways.
2. Les déployer sur site.
3. S'abonner à un serveur LoRaWAN hébergé sur un cloud.
4. Déclarer les Devices sur votre serveur LoRaWAN.
5. Activer les Devices.
6. Récupérer les données sur votre serveur.
7. Rediriger vos données vers une plateforme IoT.

Dans ce cours, la plupart du temps, nous utiliserons un réseau hybride. En effet, nous utilisons nos propres Gateways que nous nous connectons à un serveur LoRaWAN hébergé sur un cloud (Actility, TTN, Lorient...).

5.1.5 Les zones de couverture

Les zones de couverture du réseau public sont mises à jour sur les sites web de l'opérateur. Pour les réseaux communautaires, certains utilisent des applications spécifiques comme [TTN Mapper](#). L'idée est d'associer le Device LoRaWAN à un GPS dans la zone de couverture des Gateways qui nous entourent et pour chaque trame reçue, TTN Mapper enregistre les coordonnées du Device et les affiche sur une carte.

5.2 La configuration d'un réseau LoRaWAN

Dans ce chapitre, quelle que soit la solution choisie pour votre réseau, l'infrastructure doit être opérationnelle :

- Si vous disposez d'un **réseau public d'opérateur**, il n'y a rien à faire.
- Si vous disposez d'un **réseau hybride**, vos Gateways doivent être connectées à Internet et vous avez souscrit à un serveur LoRaWAN hébergé sur un cloud.
- Si vous disposez d'un **réseau privé**, vos Gateways sont connectées à Internet et vous avez installé vous-même un serveur LoRaWAN sur votre infrastructure. La mise en place d'un serveur LoRaWAN privé est décrite dans le chapitre 9.

La configuration est toujours la même :

- Étape 1 : Configuration de la Gateway.
- Étape 2 : Enregistrement de la Gateway sur le serveur LoRaWAN.
- Étape 3 : Enregistrement du Device sur le serveur LoRaWAN.
- Étape 4 : Configuration du Device.

5.2.1 Étape 1 : Configuration de la Gateway

Il existe de nombreuses marques de Gateway LoRaWAN. Chaque modèle est destiné à un usage particulier (Intérieur, extérieur, prototypage, ...). Dans tous les cas, voici les éléments que nous devons configurer :

1. Le type de **Packet Forwarder**.

2. L'**adresse IP** du Network Server.
3. La **liste des canaux** et le Data Rate associé.

Le **Packet Forwarder** est une partie du logiciel interne qui va spécifier le protocole utilisé pour communiquer avec le Network Server. Il existe de nombreux Packet Forwarder disponibles. Le plus connu est probablement le "Semtech UDP Packet Forwarder", mais c'est aussi probablement le moins sécurisé et il n'a pas beaucoup de fonctionnalités. Le "Semtech UDP Packet Forwarder" est maintenant obsolète et ne devrait plus être utilisé tel quel. Certains Network Server sont compatibles avec plusieurs Packet Forwarder, mais d'autres en exigent un en particulier et ne vous laissent pas le choix.

La Gateway reçoit les données LoRaWAN et les transfère au Network Server. La Gateway doit connaître l'**adresse IP du Network Server** ainsi que les ports TCP/UDP à utiliser. Vous trouverez sur notre site web www.univ-smb.fr/lorawan, l'adresse IP et les ports de quelques Network Server.

La Gateway reçoit un signal modulé en LoRa. Elle n'écoute que sur une liste spécifique de **canaux** et de **Data Rate**. Ces informations doivent être spécifiées à la Gateway.

```
CHANNEL0: 8671000000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL1: 8673000000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL2: 8675000000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL3: 8677000000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL4: 8679000000, A, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL5: 8681000000, B, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL6: 8683000000, B, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL7: 8685000000, B, SF7/SF12, BW125KHz (LORA_MULTI_SF)
CHANNEL8: 8683000000, B, SF7, BW250KHz (LORA_STANDARD)
```

Figure 88: Liste des canaux et Data Rate pour TTNv3

Si le Packet Forwarder est fourni par l'entreprise gérant votre serveur LoRaWAN, il contient déjà l'adresse IP et le port. Il y a également de fortes chances que les informations de la liste des canaux et des Data rate soient connus ou échangées lorsque la Gateway se connecte au Network Server. Il n'y a donc rien de spécifique à faire, mis à part installer le Packet Forwarder et tout devrait fonctionner.

Par exemple, si vous utilisez Actility comme Network Server, vous devez utiliser le **LRR (Long Range Relay Packet Forwarder)**. Lorsque vous enregistrez votre Gateway (ici une Kerlink iBTS compact), Actility propose de télécharger l'image de la Gateway avec le LRR déjà configuré (voir Figure 89).

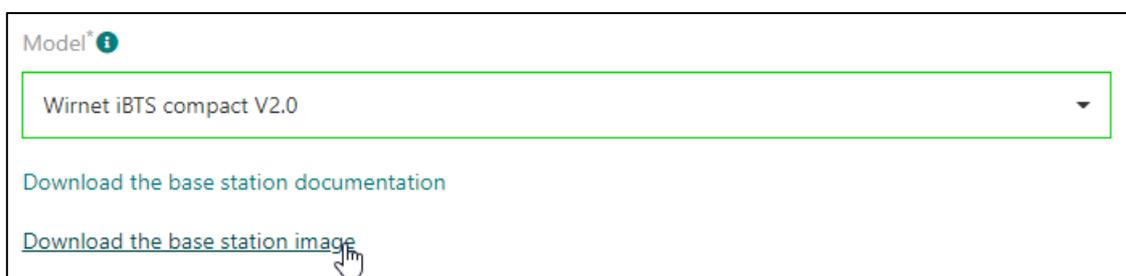


Figure 89: Enregistrement d'une passerelle chez Actility

5.2.2 Étape 2 : Enregistrement de la Gateway

L'objectif de cet enregistrement est d'autoriser la Gateway à envoyer des données au serveur. Il s'agit d'une tâche très simple. Vous devez entrer :

- Le nom
- L'ID
- La région (bande de fréquence)
- Le lieu (facultatif)

La Gateway apparaît alors sur une liste où vous pouvez généralement vérifier le trafic.

5.2.3 Étape 3 : Enregistrement du Device

Tous les Devices LoRaWAN possède un DevEUI. Il s'agit d'un identifiant unique qui est parfois difficilement modifiable. Quel que soit le mode d'activation utilisé (ABP ou OTAA), vous avez besoin du DevEUI et même si, techniquement parlant, l'activation ABP pourrait s'en passer, il est tout de même utile au Network Server pour identifier les Devices enregistrés.

Les Devices qui font tous partie d'un cas d'usage spécifique, peuvent être regroupés dans des applications. Il faut considérer cela comme un simple dossier où l'administrateur pourra appliquer des paramètres communs à une flotte de Device. Par exemple :

- Ajouter un script pour déchiffrer le Payload reçu au format JSON.
- Ajouter un connecteur pour exporter les données reçues vers une plateforme IoT.

L'enregistrement d'un Device nécessite plusieurs informations :

- Le nom : Un identifiant défini par l'administrateur.
- La version du protocole LoRaWAN : De la version 1.0.0 à la dernière version disponible (voir chapitre 4.1.2)
- La classe du Device : A, B ou C.
- La version des paramètres régionaux : voir chapitre 4.1.3
- Le plan de fréquence : Pour spécifier la bande de fréquences avec laquelle vous travaillez (EU868, US915, ...)
- Le Mode d'activation : ABP ou OTAA



Si vous ne connaissez pas la version LoRaWAN exacte de votre Device et que vous essayez simplement de mettre en place un test de base, vous pouvez utiliser sans risque la version LoRaWAN 1.0.0.

Ensuite, en fonction du mode d'activation choisi, vous aurez besoin des informations suivantes :

Pour ABP (Activation By Personalization) :

- Le **DevAddr** : vous devez générer une adresse de Device et la programmer dans le Device.
- La **NwkSKey** : vous devez générer une Network Session Key et la programmer dans le Device.
- L'**AppSKey** : vous devez générer une Application Session Key et la programmer dans le Device.

Pour OTAA (Over The Air Activation) :

- L'AppEUI/JoinEUI : Elle devrait être fournie par le fabricant du Device. Sinon, vous n'utilisez probablement pas de Join Server. Donc vous pouvez utiliser 00 00 00 00 00 00 00 00 00.
- L'AppKey : Elle doit être fournie par le fabricant du Device. Sinon, vous pouvez en générer une et la programmer dans votre Device.

5.2.4 Étape 4 : Configuration du Device

Quel que soit le mode d'activation utilisé, toutes les informations stockées dans le serveur LoRaWAN doivent être les mêmes que celles du Device.

- Si votre appareil a été pré-provisionné, alors vous n'avez rien à faire.
- Si votre Device est programmable et que vous avez généré de nouvelles clés, adresses ou DevEUI dans le serveur LoRaWAN, alors vous devez les programmer.

6 La trame LoRa / LoRaWAN

6.1 Les couches de protocole LoRaWAN

LoRa est une méthode de modulation permettant de transférer des données d'un point à un autre. Elle fait référence à la couche physique et est appelée **LoRa PHY**.

Le protocole LoRaWAN ajoute l'authentification du Device, le chiffrement des données, l'acquittement, l'administration du réseau... etc. Toutes ces propriétés protocolaires sont ajoutées au protocole LoRa, dans une couche appelée **LoRa MAC**.

Enfin, la couche **Application** correspond aux données de l'utilisateur, mais il y a aussi d'autres services disponibles dans la spécification LoRaWAN.

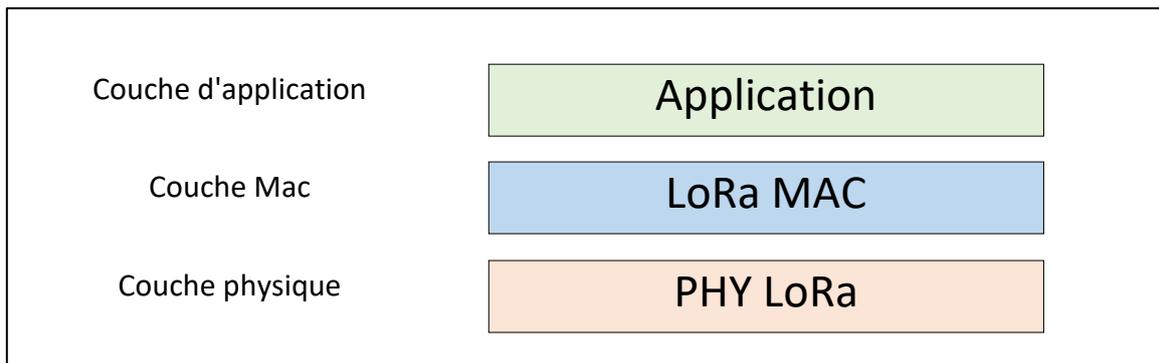


Figure 90: Les couches du protocole LoRaWAN

Chaque couche ajoute une fonctionnalité et lorsque la trame est envoyée, les données de l'utilisateur sont encapsulées dans toutes les couches inférieures. Le détail de l'ensemble de la trame LoRaWAN est décrit à la Figure 91:

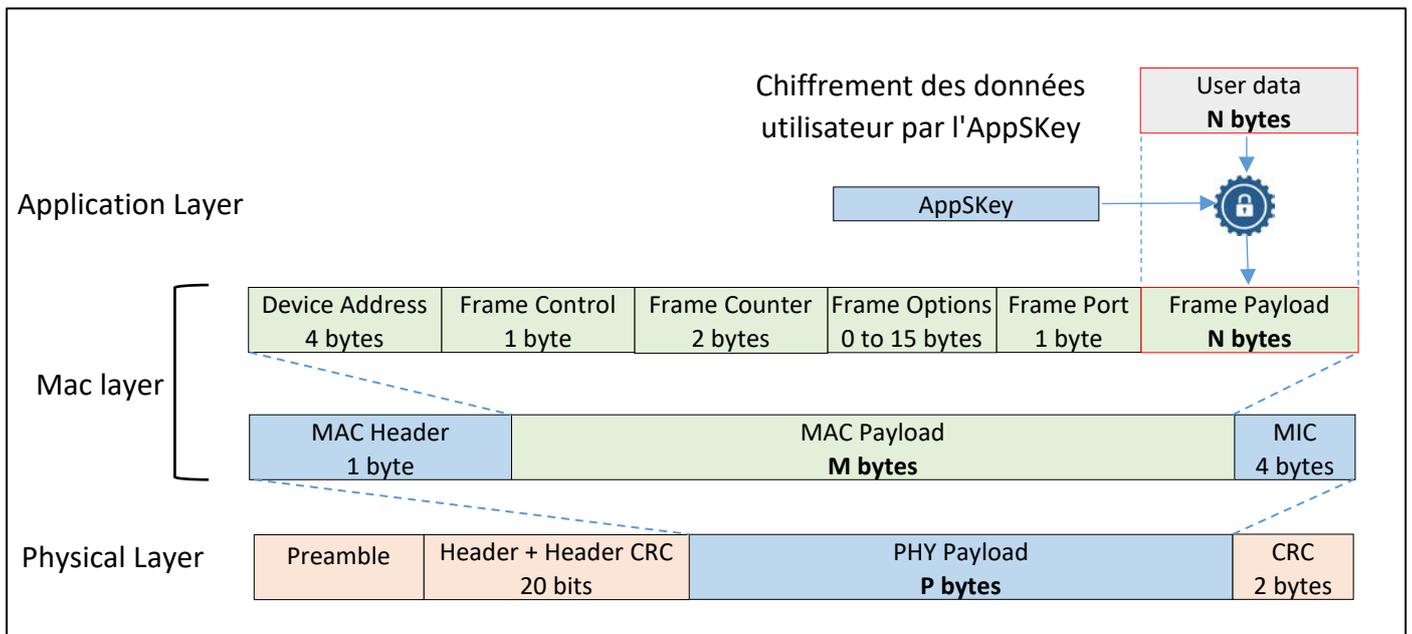


Figure 91 : Les couches de protocole

6.1.1 La couche application

La plupart du temps, la couche Application n'est composée que des données de l'utilisateur. Avant de les encapsuler dans la trame LoRaWAN, elles sont cryptées avec l'AppSKey pour sécuriser la transaction. Les données peuvent être aussi simples qu'un seul octet provenant d'un capteur. La manière dont l'utilisateur organise son Payload est totalement libre, tant qu'il respecte la taille maximale globale d'une trame LoRaWAN.



La LoRa Alliance travaille sur une proposition permettant d'harmoniser les formats de Payload. L'objectif est d'avoir des décodeurs de trame commun à un ensemble de Device LoRaWAN.

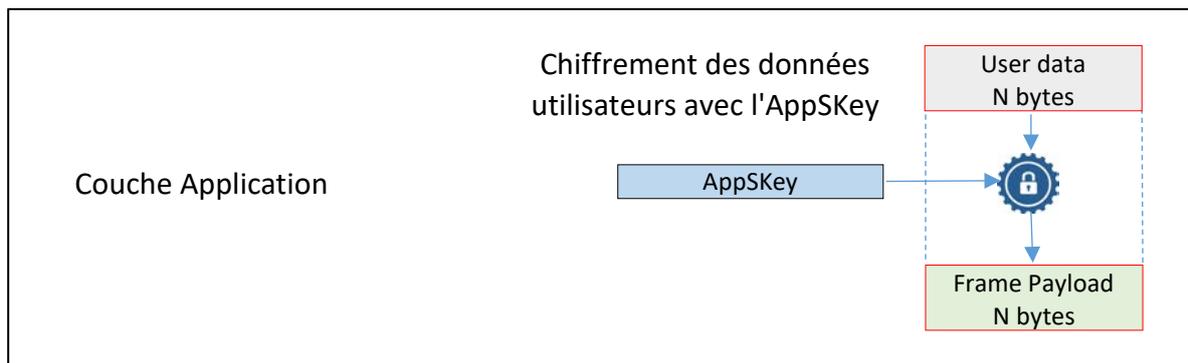


Figure 92: Couche Application LoRaWAN

En plus des Payloads utilisateur classique, le LoRaWAN propose trois services sur la couche applicative :

La synchronisation de l'horloge : La précision de l'heure peut être très utile pour effectuer une mesure de manière synchrone. L'application de synchronisation d'horloge fournit une méthode pour remettre à jour l'horloge temps réel d'un Device avec l'heure du réseau. La précision est de l'ordre de la seconde. Cette application fonctionne sur le **port 202** par défaut.

➔ Télécharger la spécification LoRaWAN sur la synchronisation d'horloge : [[ici](#)]

Le transport de blocs de données fragmentés : Cette application permet d'envoyer un bloc de données fragmenté à un ou plusieurs Devices. Elle fonctionne sur le **port 201** par défaut.

➔ Télécharger la spécification LoRaWAN sur les blocs fragmentés [[ici](#)]

Transmission en Multicast : Cette application peut envoyer des trames à un groupe de Device. Elle fonctionne sur le **port 200** par défaut.

➔ Télécharger la spécification LoRaWAN sur le multicast [[ici](#)]

6.1.2 La couche LoRa MAC

La couche LoRa MAC est le cœur du protocole LoRaWAN.

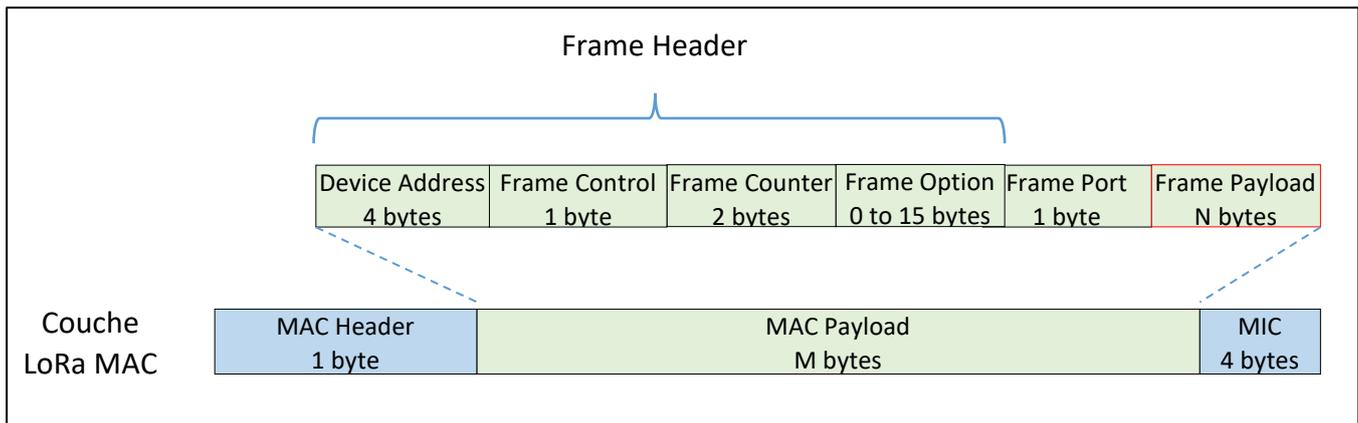


Figure 93: Couche LoRa MAC

- Le champ **MAC Header** définit le type de message : Join-Request, Join-Accept, data up, data down, confirmed, unconfirmed...
- Le **DevAddr** est l'adresse du Device.
- Le **Frame Control** donne des informations sur l'ADR (Adaptive Data Rate), l'acquittement des messages et donne également la longueur du champ **Frame Option**.
- Les propriétés du **Frame Counter** sont détaillées dans le chapitre 4.5.3.
- Le champ **Frame Option** comporte les éventuelles MAC Commands (voir chapitre 4.7).
- Le **Frame Port** est le port de l'application.
- Le **Frame Payload** comporte les données utilisateur chiffrées.
- **MIC** est le **Message Integrity Control** qui permet au message d'être authentifié par le Network Server.

Le nombre maximal d'octets pouvant être transmis par le MAC Payload (M octets) est indiqué dans le tableau suivant :

Taux de données	Facteur d'étalement	Bande passante	Charge utile maximale de la trame (nombre N)
DR 0	SF12	125 KHz	51 octets
DR 1	SF11	125 KHz	51 octets
DR 2	SF10	125 KHz	51 octets
DR 3	SF9	125 KHz	115 octets
DR 4	SF8	125 KHz	242 octets
DR 5	SF7	125 KHz	242 octets
DR 6	SF7	250 KHz	242 octets

Tableau 23 : Taille maximale du MAC Payload

6.1.3 La couche LoRa PHY

La transmission d'un message LoRaWAN est simple car aucune synchronisation n'est nécessaire entre le Device et la Gateway. Par conséquent, la couche PHY est très légère et ne comporte qu'un préambule, un entête (facultatif) et un CRC.

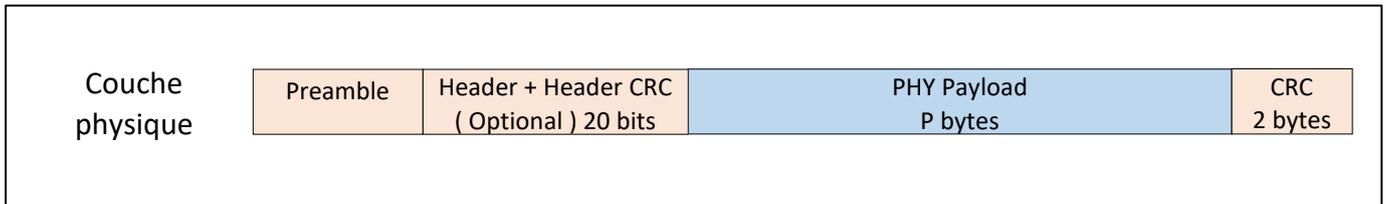


Figure 94: Détail de la couche physique LoRa

Le **préambule** est représenté par 8 symboles + 4,25 = 12,25 T_{symbole} (voir chapitre 3.1 pour un rappel de la définition d'un symbole).

L'entête **optionnel** n'est présent que dans le mode de transmission par défaut (explicite). Il est transmis avec un Coding Rate de 4/8. Il indique la taille des données, le Coding Rate pour le reste de la trame. Il annonce également si un CRC sera présent à la fin de la trame.

Le **PHY Payload** contient toutes les informations de la couche LoRa MAC.

Le **CRC** est utilisé pour détecter les erreurs dans la trame LoRa.

6.2 La communication entre les Gateways et le Network Server

La Gateway reçoit d'un côté un message radio avec une modulation LoRa et transmet de l'autre côté une trame IP au Network Server.

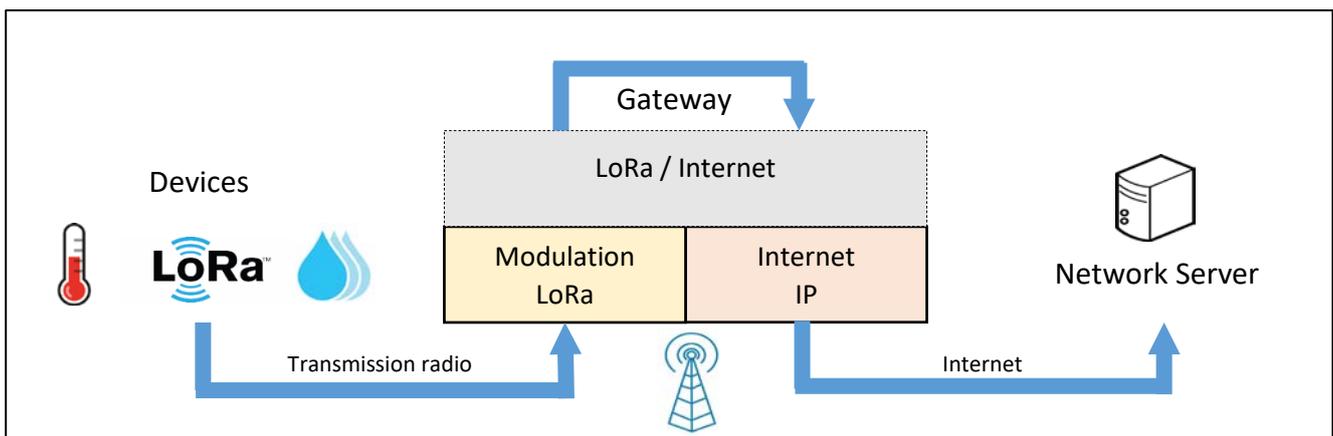


Figure 95: Rôle de la Gateway

Côté interface radio : La Gateway reçoit une trame LoRaWAN et en extrait le PHY Payload. Le format ASCII en base 64 est souvent utilisé pour représenter ce payload (voir paragraphe 6.3.2). La Gateway extrait également toutes les informations utiles sur les paramètres de transmission : SF, Bande passante, RSSI, Time On Air... etc.

Du côté de l'interface réseau IP : La Gateway transmet toutes ces informations dans un paquet IP au Network Server. Les données transmises sont au format texte JSON (voir paragraphe 6.3).

6.2.1 Le Packet Forwarder

Sur chaque Gateway, un service appelé Packet Forwarder est configuré pour effectuer la transmission des données vers le Network Server.

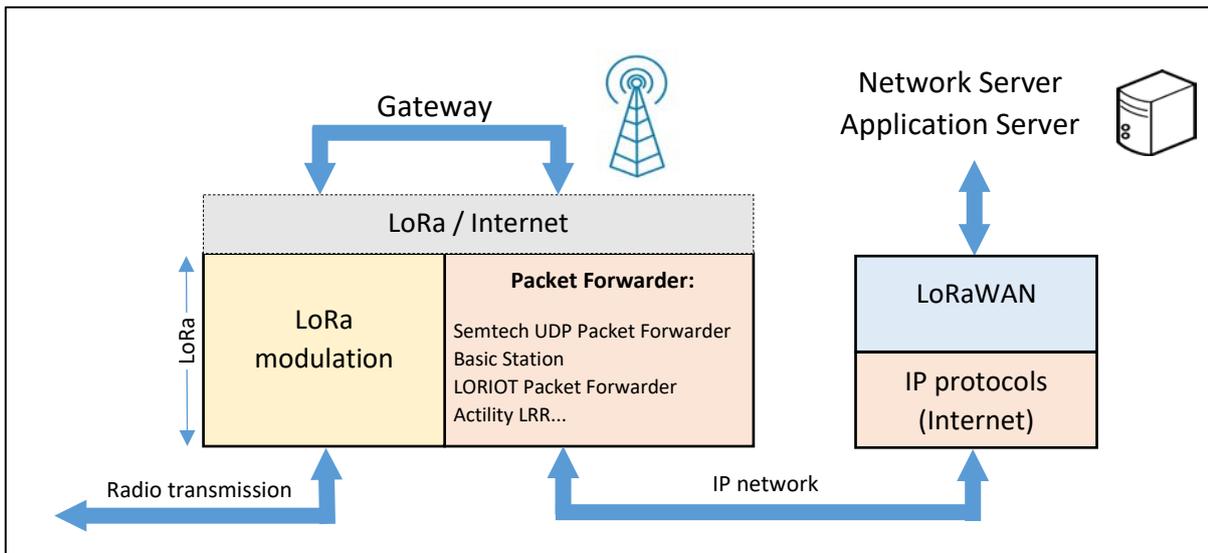


Figure 96: Gateway et serveur LoRaWAN

En fonction du Packet Forwarder utilisé, le protocole de communication entre la Gateway et le Network Server sera différent. Tout d'abord, il est obligatoire de vérifier si le Packet Forwarder est supporté par le Network Server car il en existe énormément et aucun n'est imposé par les spécifications.

- **Semtech UDP Packet Forwarder** : C'était le premier Packet Forwarder largement utilisé. Il est maintenant obsolète car il n'a pas d'autres fonctionnalités que de transmettre des données uplink et downlink et c'est un protocole non sécurisé. Ce Packet Forwarder est pris en charge par The Things Stack, mais selon la documentation, il sera supprimé à l'avenir. Il est toujours compatible avec Chirpstack si on utilise le Gateway Bridge.
- **Basic Station Packet Forwarder** : Basic Station est le nouveau Packet Forwarder fourni par SEMTECH. Il offre de nombreuses fonctionnalités supplémentaires : TLS, mise à jour du logiciel de la Gateway, synchronisation du temps, gestion de la Gateway, etc....
- **Long Rang Relay (LRR) Packet Forwarder** : Il s'agit du Packet Forwarder utilisé par Activity. Il est sécurisé et possède de nombreuses fonctionnalités.

6.2.1 Présentation de UDP Packet Forwarder (SEMTECH)

Malgré toutes les lacunes du **Semtech UDP Packet Forwarder**, son utilisation à des fins éducatives simplifie l'analyse du trafic. La documentation de ce protocole est disponible sur GitHub : https://github.com/Lora-net/packet_forwarder. Dans ce dossier, un fichier nommé PROTOCOL.TXT explique que ce protocole fonctionne au-dessus de UDP.

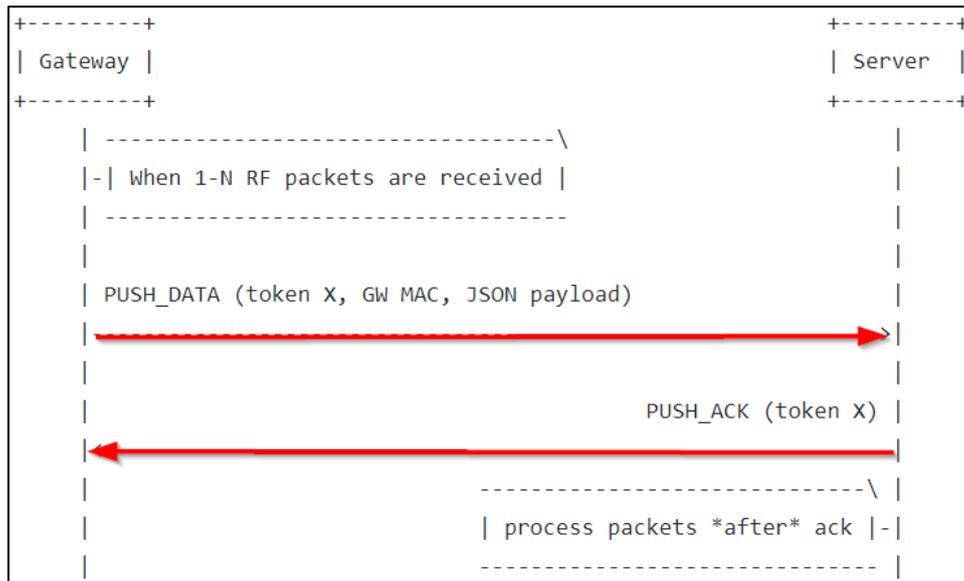


Figure 97: Protocole uplink (PUSH_DATA du Packet Forwarder)

Les paquets sont envoyés en UDP au Network Server dans une trame appelée **PUSH_DATA**. Le Network Serveur acquitte cette trame avec un **PUSH_ACK**.

```

▶ Ethernet II, Src: Raspberr_ae:26:f5 (b8:27:eb:ae:26:f5), Dst: Raspberr_5b:ce:07 (b8:27:eb:5b:ce:07)
▶ Internet Protocol Version 4, Src: 192.168.138.151, Dst: 192.168.138.168
▶ User Datagram Protocol, Src Port: 39579, Dst Port: 1700
▼ Data (197 bytes)
  Data: 02f93000b827ebfffeae26f57b227278706b223a5b7b2274...
  [Length: 197]
  
```

Figure 98: Trame PUSH_DATA dans Wireshark

Ce Packet Forwarder a été configuré pour utiliser le port UDP 1700. Le contenu des données (Data field) est détaillé dans le tableau suivant :

Champ	octet	Fonction
[1]	0	Version du protocole = 0x02
[2]	1-2	Jeton aléatoire
[3]	3	PUSH_DATA identifiant = 0x00
[4]	4-11	Identifiant unique de la Gateway (adresse MAC)
[5]	12-end	Objet JSON, commençant par { et finissant par }

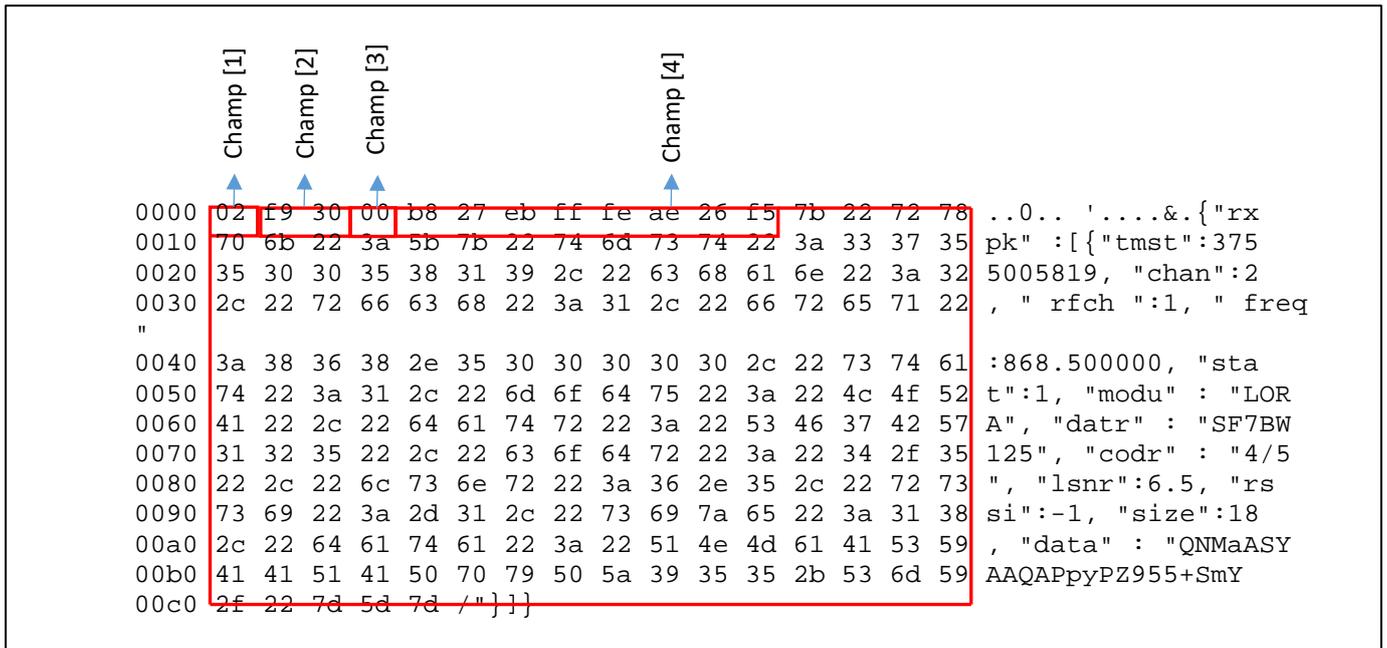


Figure 99: Analyse du Semtech UDP Packet Forwarder

L'objet JSON de la transmission est le suivant :

```

{
  "rxpk": [ {
    "tmst": 3755005819,
    "chan": 2,
    "rfch": 1,
    "freq": 868.500000,
    "stat": 1,
    "modu": "LORA",
    "datr": "SF7BW125",
    "codr": "4/5",
    "lsnr": 6.5,
    "rssi": -1,
    "size": 18,
    "data": "QNMaASYAAQAPpyPZ955+SmY/"
  } ]
}

```

Le champ data correspond au PHY Payload. De la même manière, nous capturons la trame d'acquittement.

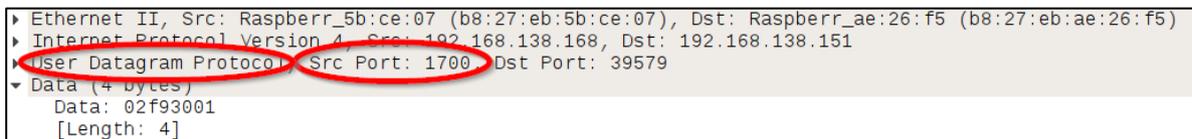


Figure 100: Trame PUSH_ACK dans Wireshark

Champs	Octet	Fonction
[1]	0	Version du protocole = 0x02
[2]	1-2	même jeton que le PUSH_DATA pour accuser réception.
[3]	3	PUSH_ACK identifiant = 0x01

On retrouve bien sûr tous ces champs dans la trame Wireshark.

6.3 L'analyse des trames IP

6.3.1 Le format JSON

Les données de l'application sont formatées en JSON qui est un format texte composé d'une succession de paires nom/valeur. Sur la Figure 101, "gw_id" est un nom et "eui-b427ebfffeae26f5" est la valeur associée (une chaîne de caractères dans ce cas). Les objets sont délimités par une paire d'accolades { }. Le Tableau 23 représente les différents types de valeurs acceptées dans le format JSON.

Type	Exemple
String	"coding_rate": "4/5"
Number	"spreading_factor": 12
Object	"lora": { "spreading_factor": 12, "air_time": 2465792000 }
Boolean	"service": true

Tableau 24 : Type de valeur au format JSON

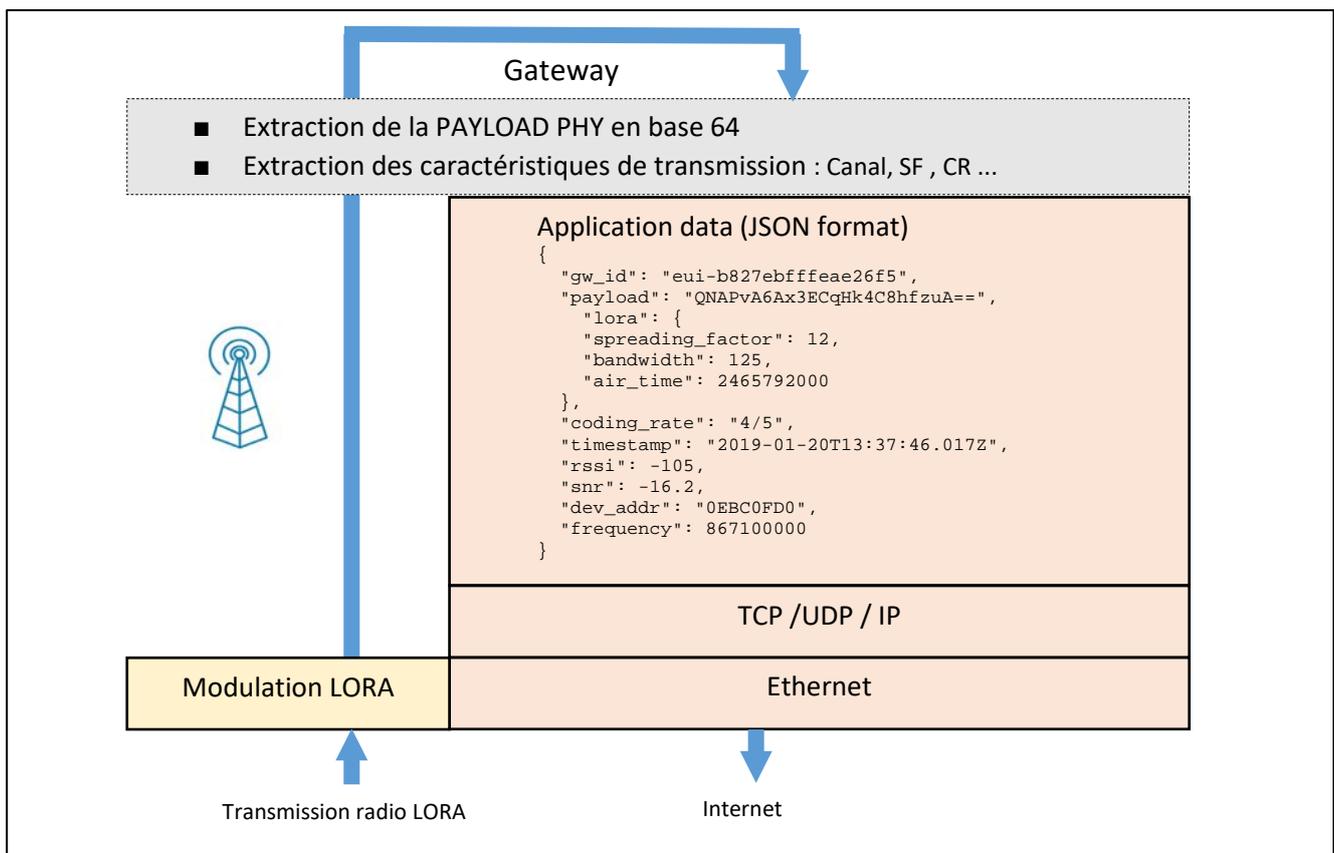


Figure 101: Gateway LORAWAN

6.3.2 La base 64

La Gateway extrait le PHY Payload de la modulation LoRa. Comment pouvons-nous représenter ces données binaires ?

En Binaire (base 2) : La méthode la plus simple consiste à représenter chaque élément binaire (0 et 1) mais elle présente un très gros inconvénient : la représentation est complexe à lire en raison du nombre de bits représentant le message. Si l'on veut représenter une trame LoRa de 50 octets, il faut écrire 400 bits '0' ou '1'.

En Hexadécimal (base 16) : On fait des groupes de 4 bits, ce qui fait 16 combinaisons possibles. Les 16 caractères utilisés sont 0, 1, 2,3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Cette méthode a l'avantage d'être 4 fois plus compacte que la représentation binaire. Peut-on faire mieux ?

En Base 64 : Les bits sont groupés par 6, ce qui donne 64 combinaisons possibles. Les 64 caractères utilisés sont ceux du tableau ci-dessous :

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	ø
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	F	47	101111	v	63	111111	/
Padding		=									

Tableau 25 : Caractères de codage en base 64 (source Wikipédia)

Cette méthode a l'avantage d'être 6 fois plus compacte que la représentation binaire et **elle n'utilise que des caractères imprimables ASCII**. Peut-on faire mieux ?

En Base 256 (ASCII) : Les bits sont regroupés par 8, ce qui donne 256 combinaisons possibles. Les 256 caractères utilisés sont ceux de la table ASCII que l'on trouve facilement sur le web. Cette méthode a l'avantage d'être 8 fois plus compacte que la représentation binaire mais elle a aussi un énorme inconvénient : **de nombreux caractères de cette représentation sont non imprimables** (saut de ligne, espace, EOF, ...) et ne sont donc pas visibles. Cette représentation est donc utile pour le codage de texte, mais inutilisable si l'on veut représenter des données binaires.



➔ **Le meilleur compromis est la base 64. Cette méthode est souvent utilisée pour représenter le Payload de la trame LoRa.**

6.3.3 Exemple de codage en base 64

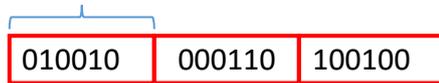
La méthode de représentation en base 64 est donnée au travers de l'exemple suivant : nous souhaitons encoder la valeur hexadécimale 0x4869.

1. Les données hexadécimales sont écrites en binaire.

$$0x4869 = 0100\ 1000\ 0110\ 1001$$

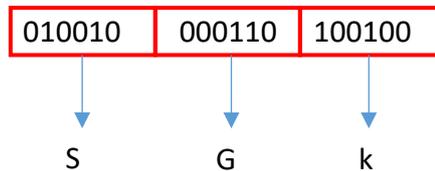
2. Les éléments binaires sont regroupés en blocs de 6 bits. Le nombre de blocs de 6 bits doit être un multiple de 4 (minimum 4 blocs). Si des bits manquent pour former un groupe de 6 bits, des zéros sont ajoutés.

Bloc de 6 bits

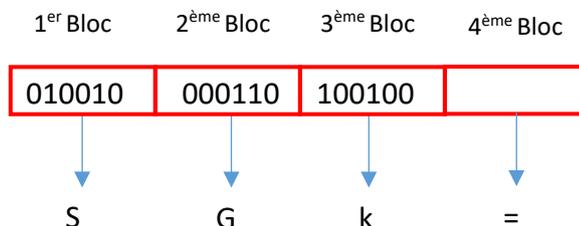


On ajoute des 00 pour avoir 6 bits dans chaque groupe

3. S'il manque des blocs pour obtenir un minimum de 4 blocs, des caractères spéciaux seront ajoutés.
4. Chaque groupe de 6 bits est traduit en utilisant le Tableau 24.



5. Si un bloc de 6 bits est manquant (ils doivent être un multiple de 4), un ou plusieurs sont ajoutés avec le caractère "=".



Résultat : L'encodage de 0x4869 en base 64 est "SGk=".



➔ On veut coder le code ASCII "AA" en base 64. Reprendre la procédure ci-dessus et montrer que le résultat en base 64 est "QUE=".

6.3.4 Trame uplink : du Device LoRaWAN au Network Server

Lorsque le Network Server reçoit une trame IP de la Gateway, plusieurs informations peuvent être facilement extraites : DevAddr, SF, bande passante... mais les données de l'application sont bien sûr cryptées (avec l'**AppSKey**). Sans connaître l'**AppSKey**, il n'est pas possible de comprendre le contenu du message reçu.

Supposons que la trame IP reçue par le Network Server soit la suivante :

```
{
  "gw_id" : "eui-b827ebfffeae26f6",
  "payload" : "QNMaASYABwAP1obuUHQ=",
  "f_cnt" : 7,
  "lora" : {
    "facteur d'étalement" : 7,
    "bande passante" : 125,
    "air_time" : 46336000
  },
  "coding_rate" : "4/5",
  "timestamp" : "2019-03-05T14:00:42.448Z",
  "rssi" : -82,
  "snr" : 9,
  "dev_addr" : "26011AD3",
  "fréquence" : 867300000
}
```

Le Network Server affichera alors les informations suivantes :

time	frequency	mod.	CR	data rate	airtime (ms)	cnt	
▲ 15:00:42	867.3	lora	4/5	SF 7 BW 125	46.3	7	dev addr: 26 01 1AD3 payload size: 14 bytes

Figure 102: Trame reçue sur le Network Server

Nous pouvons trouver les valeurs suivantes fournies par la Gateway :

- Timestamp
- Channel : 867.3 MHz
- Modulation : Lora
- Coding Rate : 4/5
- Data Rate: SF 7 / 125 kHz (DR5)
- Time on Air : 46,3 ms

Si nous voulons plus d'informations, nous pouvons nous plonger dans le PHY Payload. Bien sûr, il y a une partie chiffrée (**Frame Payload**), mais les entêtes sont en clair (voir paragraphe 6.1). Le PHY Payload de notre exemple est "QNMaASYABwAP1obuUHQ=" en base 64 ou "40D31A01260007000FD686EE5074" en hexadécimal. Il a une taille de 14 octets.



Figure 103: PHY Payload d'une trame LoRa

Avec le format hexadécimal du PHY Payload, nous pouvons trouver chaque champ de la trame en utilisant la Figure 91. Nous obtenons le résultat suivant :

PHYPayload = 40D31A01260007000FD686EE5074	
PHYPayload = MAC Header[1 byte] MACPayload[..] MIC[4 bytes]	
MAC Header	= 40 (Unconfirmed data up)
MACPayload	= D31A01260007000FD6
Message Integrity Code	= 86EE5074
MACPayload = Frame Header Frame Port Frame Payload)	
Frame Header	= D31A0126000700
FPort	= 0F
FramePayload	= D6
Frame Header = DevAddr[4] FCtrl[1] FCnt[2] FOpts[0..15]	
DevAddr	= 26011AD3 (Big Endian)
FCtrl (Frame Control)	= 00 (No ACK, No ADR)
FCnt (Frame Counter)	= 0007 (Big Endian)
FOpts (Frame Option)	=

Vous pouvez vérifier le résultat par vous-même en utilisant le [décodeur de trame LoRaWAN 1.0.x](#) en ligne.

LoRaWAN 1.0.x packet decoder

A frontend towards [lora-packet](#).

Base64 or hex-encoded packet

Secret NwkSKey (hex-encoded; optional) Secret AppSKey (hex-encoded; optional)

Figure 104: Décodeur de trame LoRaWAN

6.3.5 Uplink : Network Server vers Application Server

Nous utiliserons le même PHY Payload que celui de la Figure 103. Pour information, les NwkSKey et AppSKey utilisés lors de cette transmission sont :

- NwkSKey : E3D90AFBC36AD479552EFEA2CDA937B9
- AppSKey : F0BC25E9E554B9646F208E1A8E3C7B24

Le Network Server décode l'ensemble de la trame et vérifie la valeur MIC. Si le MIC est correct (authentification de la trame par la **NwkSKey**), le Network Server transmet le contenu du message chiffré (Frame Payload) à l'Application Server. D'après le résultat du décodage du chapitre précédent, le Frame Payload est :

FramePayload	=	D6
--------------	---	----

D6 est le contenu chiffré. Quand il est déchiffré avec l'**AppSKey**, on trouve **01**. Vous pouvez vérifier toutes ces informations avec le [décodeur de trame LoRaWAN 1.0.x](#).

L'Application Server ne recevra les données chiffrées que si le Device LoRaWAN a été enregistré.

time	counter	port	
▲ 15:00:42	7	15	payload: 01

Figure 105: Frame Payload déchiffrée dans l' Application Server

7 Exporter nos données du serveur LoRaWAN

Nous allons voir dans ce chapitre comment extraire nos données du serveur LoRaWAN.

7.1 Les services fournis par la plateforme IoT

Nous avons vu jusqu'à présent comment envoyer des données d'un Device au serveur LoRaWAN. Ces données doivent être transférées sur l'Application utilisateur (plateforme IoT) pour pouvoir être stockées dans une base de données, présentées sous différentes formes (tableaux, graphiques...), et enfin mises à disposition via un service web que l'utilisateur peut interroger. C'est le rôle d'une plateforme IoT.

Toute cette partie est totalement indépendante du protocole LoRa et LoRaWAN que nous avons étudié jusqu'à présent, et n'a aucun lien avec lui. Les explications qui suivent peuvent donc être facilement transposées à tout autre protocole lié à l'Internet des objets.

Sur la Figure 106, nous avons sur le côté gauche la communication entre le Device LoRaWAN et les serveurs LoRaWAN (NS et AS). De l'autre côté, nous avons la communication entre le serveur LoRaWAN et la plateforme IoT. La plateforme IoT sera le lien avec l'utilisateur et devra effectuer les actions suivantes :

- Recevoir des données du serveur LoRaWAN (uplink).
- Transmettre des données au serveur LoRaWAN (downlink).

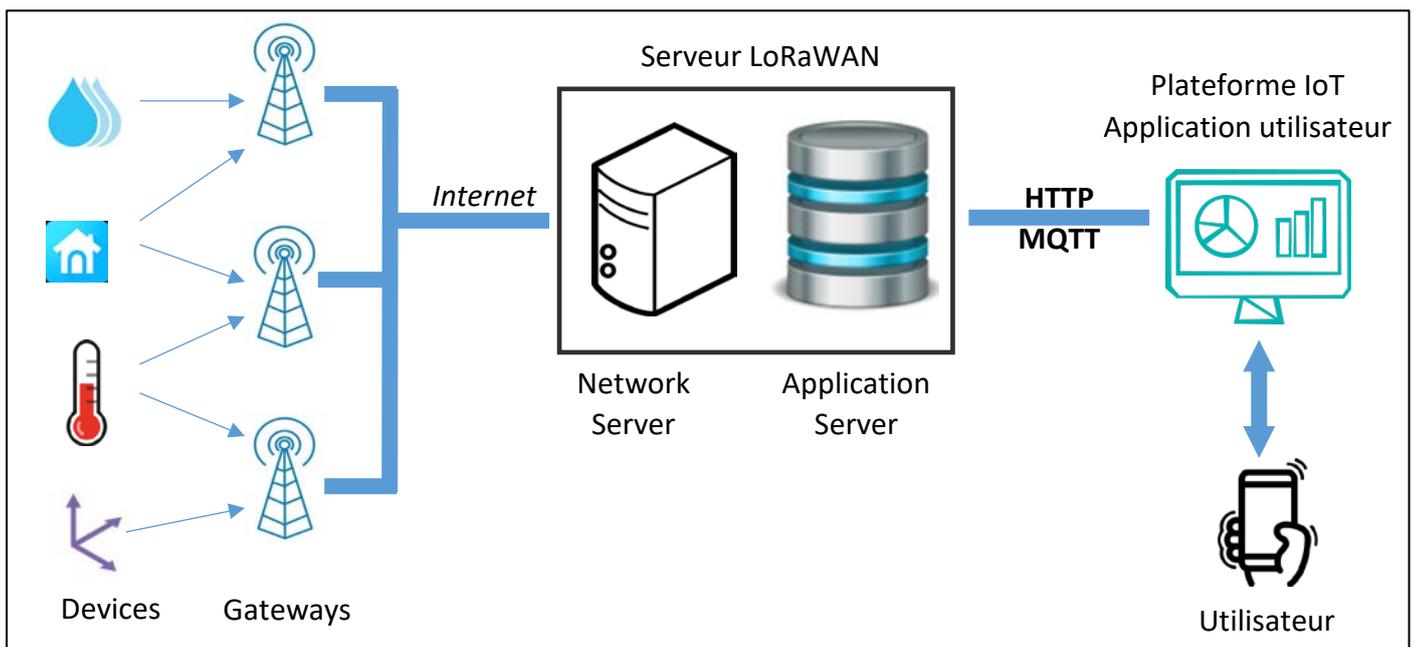


Figure 106: Structure globale d'un réseau LORAWAN

Notez que si votre Network Server fournit une sécurité de bout en bout, vous aurez l'architecture de la Figure 107 avec, dans ce cas, la plateforme IoT qui joue aussi le rôle de l'Application Server . Mais cela ne change pas l'objectif de ce chapitre.

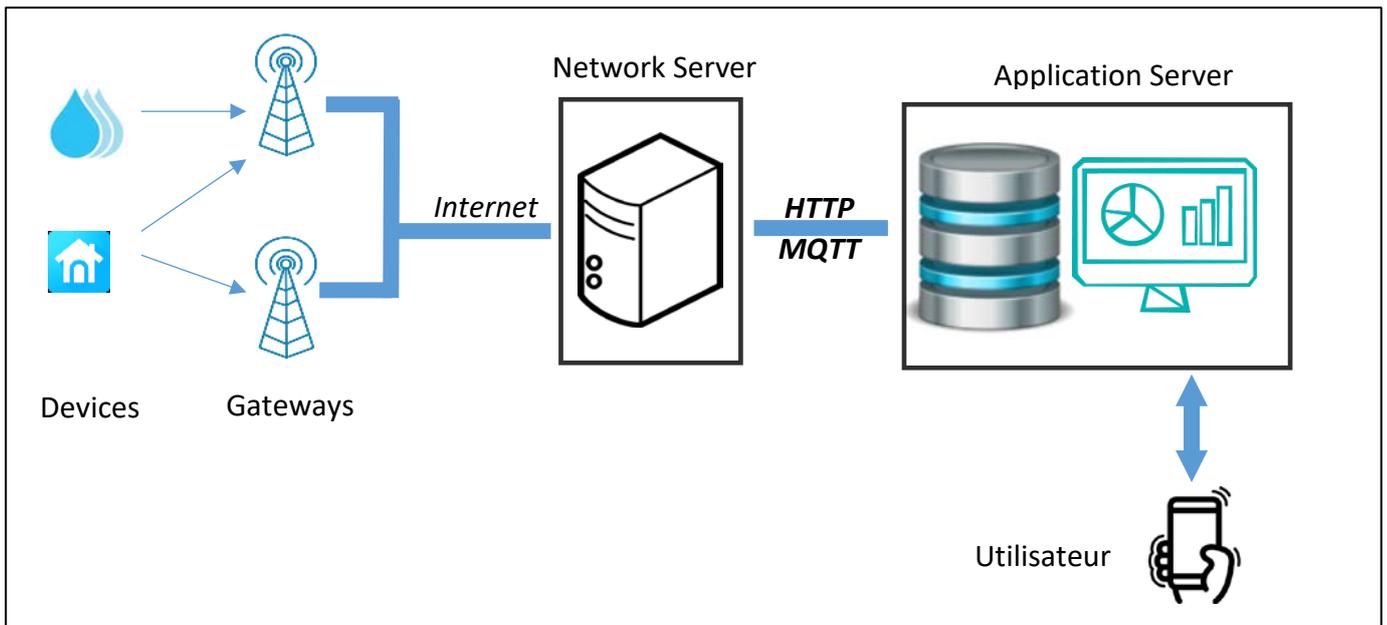


Figure 107: Architecture LoRaWAN globale avec sécurité de bout en bout

Le dialogue entre les serveurs LoRaWAN et la plateforme IoT peut se faire à l'aide de différents protocoles que nous étudierons dans les prochains chapitres.

Pour l'uplink, notre plateforme IoT aura les rôles suivants :

1. **Importation des données (uplink)**
2. Stockage des données (sauvegarde).
3. Mise en forme des données (graphiques, tableaux...).
4. Affichage des données demandées par l'utilisateur (interface web).

Pour le downlink, notre plateforme IoT aura les rôles suivants :

1. Affichage d'une interface utilisateur (bouton, champ de texte, ...).
2. Traitement des demandes des utilisateurs.
3. Stockage de la requête (sauvegarde).
4. **Transmission des données (downlink)**

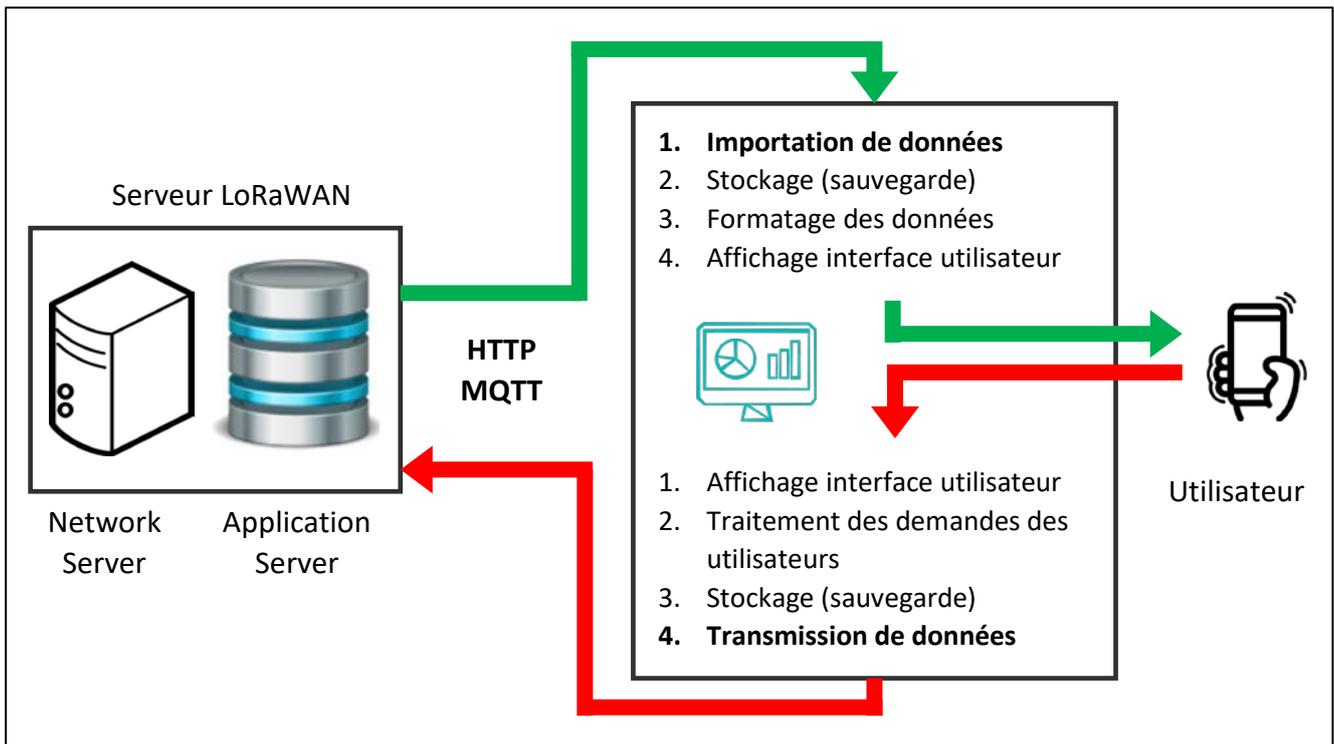


Figure 108: Services de la plateforme IoT

Dans ce chapitre, nous ne traiterons que les deux éléments **en gras** de la liste précédente, à savoir "**Importation de données (uplink)**" et "**Transmission de données (downlink)**".

Nous verrons deux méthodes pour communiquer : le protocole **HTTP** et le protocole **MQTT**.

7.2 Exportation des données avec le protocole HTTP GET

7.2.1 Présentation du principe Client - Serveur

Comme la plupart des protocoles, le protocole HTTP implique le transfert d'informations entre un client et un serveur. Le client et le serveur sont deux entités distantes qui souhaitent communiquer l'une avec l'autre. Le client fait des requêtes et le serveur répond. Le client et le serveur peuvent être de n'importe quel type : email (SMTP), fichiers (FTP)... Ici, nous utilisons un client HTTP et un serveur HTTP.

La Figure 109 montre un client, un serveur et les deux types de messages envoyés : les requêtes et les réponses. Il est très important de savoir qui sera le client et qui sera le serveur. En effet, nous devons attribuer un rôle (client ou serveur) à notre serveur LoRaWAN et à notre plate-forme IoT.

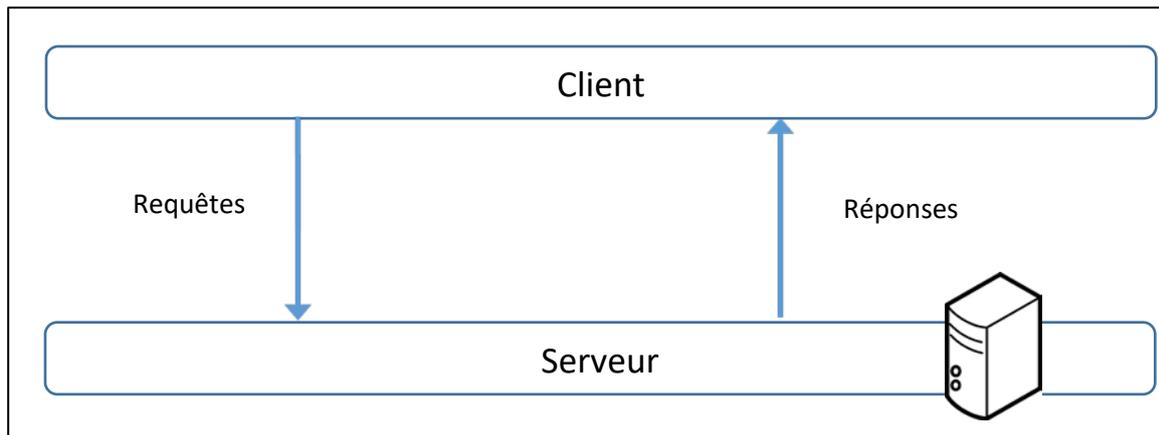


Figure 109: "Client - Serveur" et "Requêtes - Réponses".

7.2.2 Désignation du client et du serveur

Le dialogue a lieu entre le serveur LoRaWAN et l'application utilisateur (plateforme IoT). La Figure 110 représente ces deux entités et les échanges. En haut de la figure, nous avons la serveur LoRaWAN (TTN, Actility, Lorient, Chirpstack, etc...), et en bas, nous avons la plateforme IoT de l'utilisateur. Nous devons maintenant comprendre "qui fait la requête?" et "qui répond?".

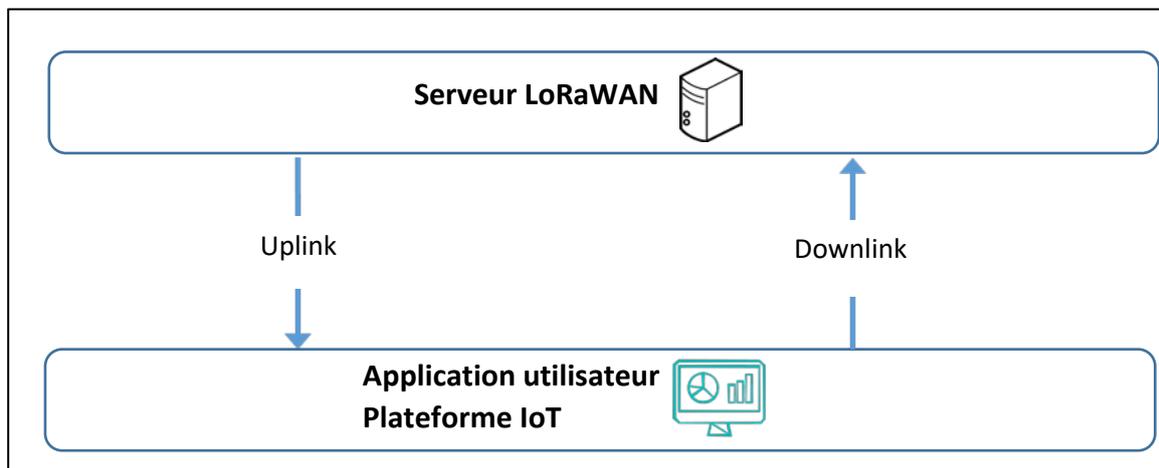


Figure 110: Communication entre le serveur LoRaWAN et la plateforme IoT

Nous allons maintenant étudier les deux situations représentées dans Figure 111:

- Uplink : du serveur LoRaWAN vers la plateforme IoT (application utilisateur).
- Downlink : de la plateforme IoT (application utilisateur) au serveur LoRaWAN.

Commençons par l'uplink qui est la situation la plus courante. Depuis l'application utilisateur, nous voulons récupérer les données qui se trouvent sur le serveur LoRaWAN. La première idée est de faire une requête **(1)** afin d'obtenir ces données. Cette requête s'appelle une requête HTTP GET. Lorsque vous faites une requête HTTP GET à un serveur Web, celui-ci renvoie le contenu de la page HTML qu'il contient. Ici, le serveur LoRaWAN renvoie les données LoRaWAN **(2)**. Dans ce cas :

- L'application utilisateur est le client HTTP.
- Le serveur LoRaWAN est le serveur HTTP.

Maintenant, parlons du downlink. Depuis le serveur LoRaWAN, nous voulons récupérer les données qui se trouvent sur l'application utilisateur. L'idée est de faire une requête **(3)** à l'Application

utilisateur pour obtenir les données et l'application utilisateur renverra ces données au serveur LoRaWAN (4). Dans ce cas :

- Le serveur LoRaWAN est le client HTTP.
- L'application utilisateur est le serveur HTTP.

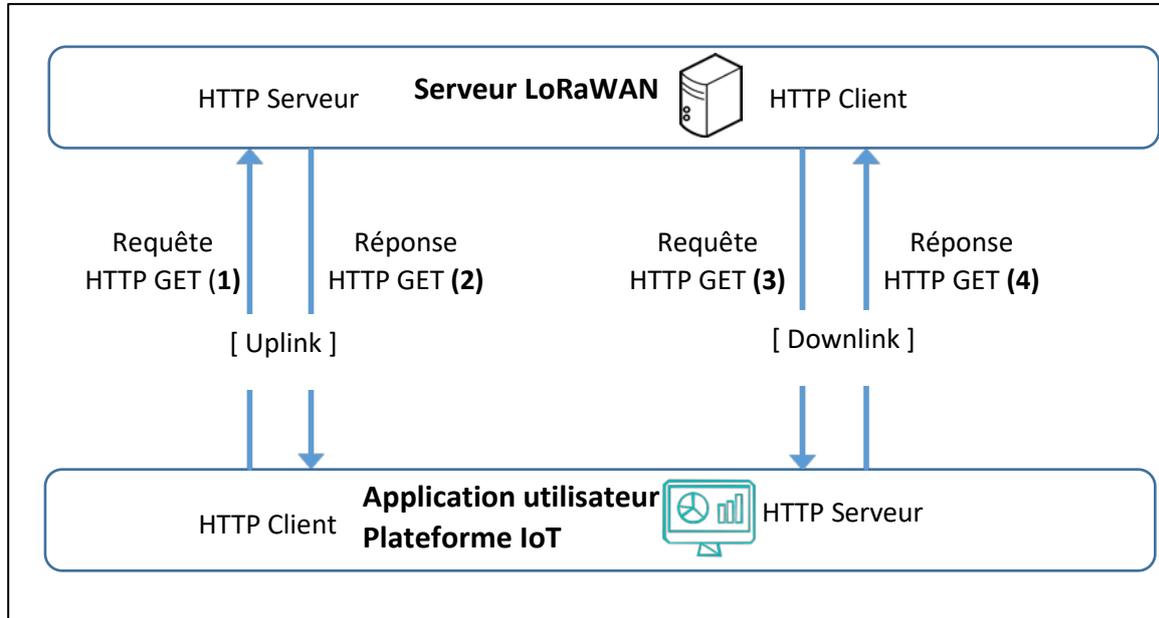


Figure 111: Uplink et downlink (HTTP GET)

Depuis le client, pour effectuer une requête HTTP GET, nous avons besoin :

- De l'URL du serveur avec le format à respecter.
- D'un token (jeton de clé API) pour avoir le droit d'effectuer une requête sur le serveur HTTP.

7.2.3 Configuration du service HTTP GET pour l'uplink

Nous allons configurer un serveur HTTP GET. Dans la fenêtre Figure 111, cela représente les trames uplink (1) et (2). Le serveur HTTP sera sur le serveur LoRaWAN et le client HTTP sur l'application utilisateur.



Pour cet exemple, nous utilisons le serveur LoRaWAN TTN community (v3.15.1). Le service HTTP GET est appelé "Storage Integration".

Nous commençons par configurer le serveur HTTP sur TTN. Pour ce faire, nous devons aller dans notre console **TTN > Applications > Nom de l'application > Integration > Storage Integration > Activate Storage Integration**. Le serveur est alors disponible ainsi qu'une sauvegarde temporaire des données.

Pour connaître les API disponibles pour récupérer les données, consultez la documentation sur "[Storage Integration API](#)".

Nous allons utiliser le logiciel [POSTMAN](#) comme client HTTP qui permet de générer toutes sortes de requêtes HTTP.

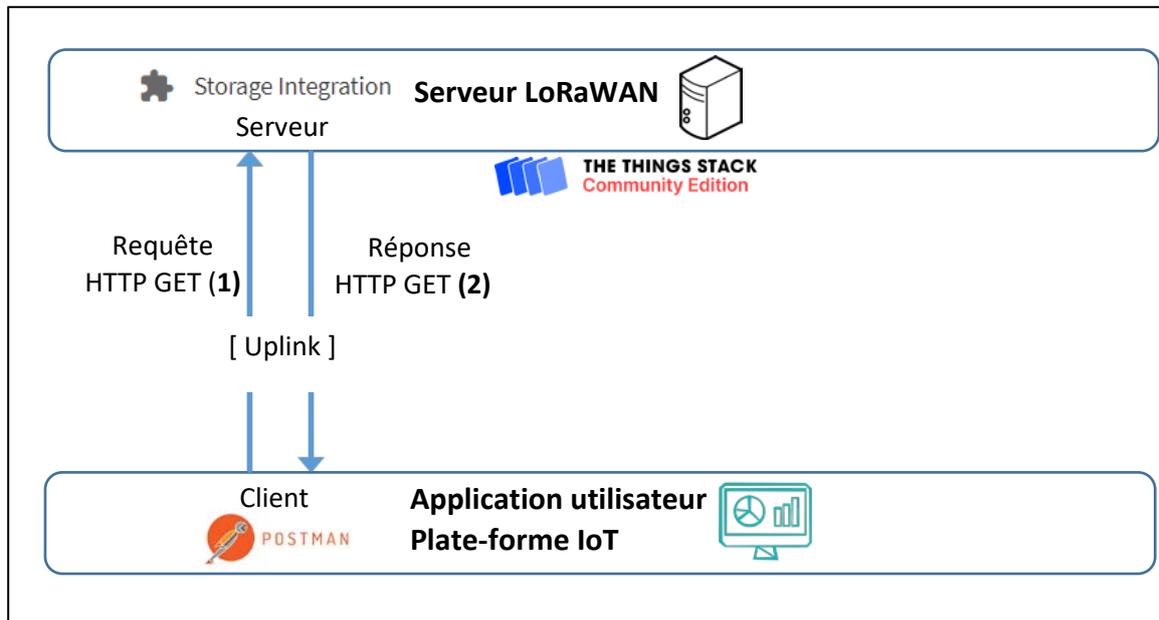


Figure 112: Configuration de l'uplink avec une requête HTTP GET

Dans POSTMAN, nous allons maintenant effectuer une requête HTTP GET. Nous suivons la [documentation](#) et essayons de récupérer tous les messages uplink d'une même application.

Créez une clé API : **TTN > Applications > votre application > API Key > Add API Key > Grant all current and future rights**, puis copiez la clé.

➔ **POSTMAN > Importation > Texte brut**

```
curl -G
"https://eu1.cloud.thethings.network/api/v3/as/applications/app1/p
ackages/storage/uplink_message" \
-H "Autorisation : Bearer $API_KEY" \
-H "Accept : text/event-stream"
```

- ➔ Remplacez "**app1**" par le nom de votre application
- ➔ Changez **\$API_KEY** par votre clé API
- ➔ Importez la requête et envoyez-la.

Vous devriez recevoir une réponse (statut 200 : OK) avec un message texte JSON contenant votre trame de uplink.

7.2.4 Informations sur la méthode HTTP GET

Cette méthode est intéressante pour sa simplicité puisque nous n'avons qu'à envoyer des requêtes HTTP GET chaque fois que nous avons besoin du message de uplink reçu sur le serveur LoRaWAN.

Le premier inconvénient est que nous n'avons travaillé que sur le flux montant. Il n'y a pas de possibilité d'envoyer un message descendant au serveur LoRaWAN. Aucun serveur LoRaWAN n'a la partie droite de la Figure 111 pour mettre en œuvre les trames **(3)** et **(4)**.

Le deuxième inconvénient est que pour le flux uplink, nous passons notre temps à demander des données qui n'existent potentiellement pas. En effet, nous interrogeons le serveur HTTP sans savoir s'il a des informations à nous retourner. Si un capteur émet des valeurs de manière non régulière, alors nous devons faire des requêtes périodiques avec une forte probabilité d'obtenir des réponses vides. Faire des requêtes fréquentes au serveur augmente considérablement la charge du réseau.

Pour le downlink, si nous avons pu installer le client sur le serveur LoRaWAN, le problème serait le même. Nous passerions notre temps à demander des commandes utilisateur alors qu'il y a de fortes chances que l'utilisateur n'en ait pas envoyées.

La solution consiste à réorganiser les clients, les serveurs et les requêtes pour essayer d'optimiser la façon dont nous transférons les données entre le serveur LoRaWAN et notre application utilisateur. Ceci sera possible grâce aux requêtes HTTP POST.

7.3 Exportation de données avec le protocole HTTP POST

Pour l'uplink, nous attribuons les rôles de manière différente en imaginant que l'Application Utilisateur ne demandera pas les informations au serveur LoRaWAN, mais que ce dernier les fournira de lui-même. Le serveur LoRaWAN va donc transmettre à l'Application Utilisateur une requête appelée HTTP POST (1) pour "poster" les données. La réponse (2) est un simple acquittement et ne contient aucune donnée. Dans ce cas :

- Le serveur LoRaWAN est le client HTTP.
- L'application utilisateur est le serveur HTTP.

Pour le downlink, l'utilisateur qui souhaite transmettre des données au serveur LoRaWAN doit fournir une requête HTTP POST (3) et le serveur répondra par un acquittement. Dans ce cas :

- L'application utilisateur est le client HTTP.
- Le serveur LoRaWAN est le serveur HTTP.

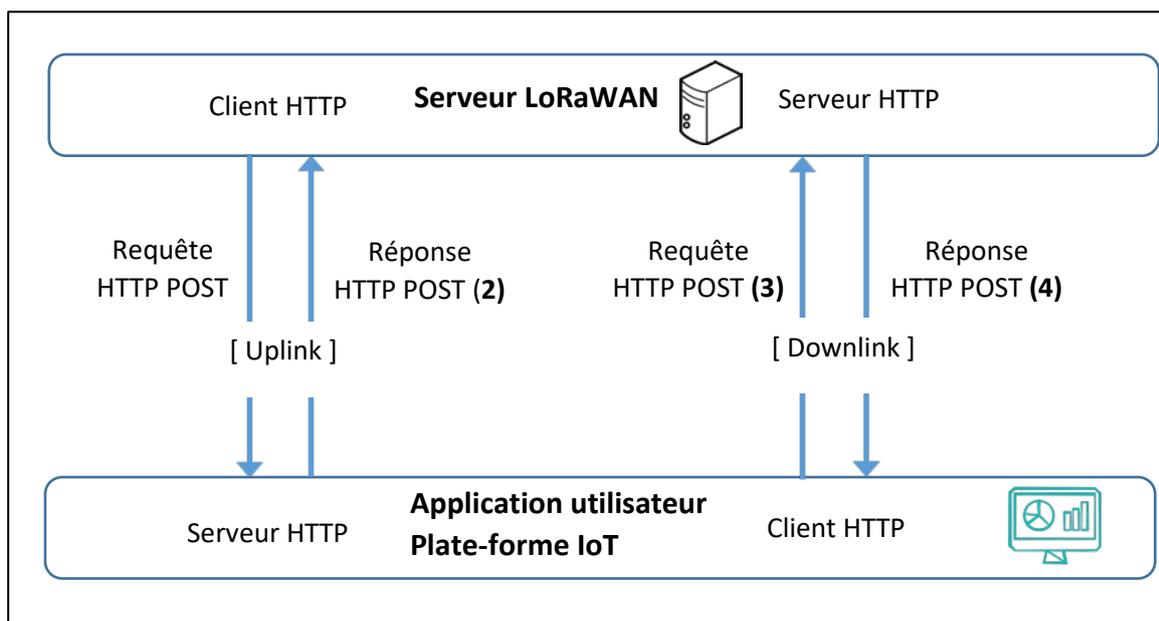


Figure 113: Uplink et downlink - HTTP POST

7.3.1 Configuration des services HTTP POST pour le flux Uplink



Pour cet exemple, nous utilisons le serveur LoRaWAN LORIIOT (v7.0.14). Le service HTTP POST est appelé "HTTP Push Output".

Nous voulons recevoir les données présentes sur les serveurs LoRaWAN. Sur la Figure 113, cela représente les échanges de trames **(1)** et **(2)** (uplink).



La démarche pour les autres Serveurs LoRaWAN est très similaire. Vous trouverez sur notre site web www.univ-smb.fr/lorawan la méthode pour les informations pour les autres Serveurs (Actility, TTN, Chirpstack, LoRa Cloud...).

Nous devons configurer un client HTTP POST sur le serveur de LORIIOT et un serveur HTTP POST sur notre application utilisateur.

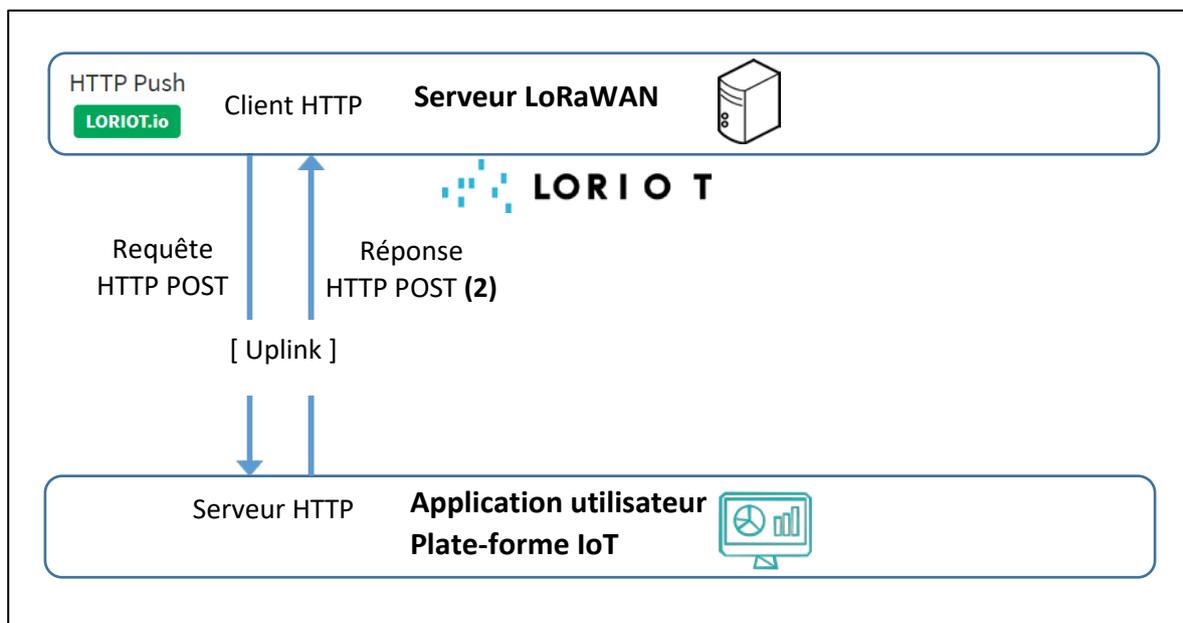


Figure 114: Uplink HTTP POST avec LORIIOT

Nous commençons par configurer le serveur. Nous allons utiliser un serveur HTTP disponible sur le web, gérant les requêtes HTTP POST [<https://rbaskets.in/>] ou [<https://beeceptor.com/>] par exemple.

- ➔ Allez sur <https://rbaskets.in/> et créez un nouveau basket (Endpoint server).
- ➔ Conservez le token si vous souhaitez revenir ultérieurement sur le même serveur et cliquez sur "Open Basket".
- ➔ L'adresse à laquelle vous devez envoyer vos demandes est ensuite spécifiée. C'est celle que nous utiliserons pour configurer le client.

Le serveur HTTP POST est prêt et il attend vos données. Votre "basket" est vide mais c'est là que vous recevrez les données de LORIIOT.

Maintenant, nous devons configurer le client HTTP POST sur le serveur LoRaWAN LORIoT.

- ➔ **LORIoT > Applications > Votre Application > Output > Add New Output > HTTP Push**, et entrez l'adresse de votre serveur sous "Target URL for POSTs".

Output Configuration

Target URL for POSTs

Le client HTTP POST est configuré. Vous pouvez envoyer des données avec votre Device LoRaWAN et les recevoir au format JSON sur votre serveur.

7.3.2 Configuration des services HTTP pour le downlink

Nous allons envoyer les données de l'utilisateur au serveur LoRaWAN. Dans la Figure 113 cela représente les trames downlink (3) et (4). Nous devons configurer un client HTTP sur notre application utilisateur et un serveur HTTP sur LORIoT.

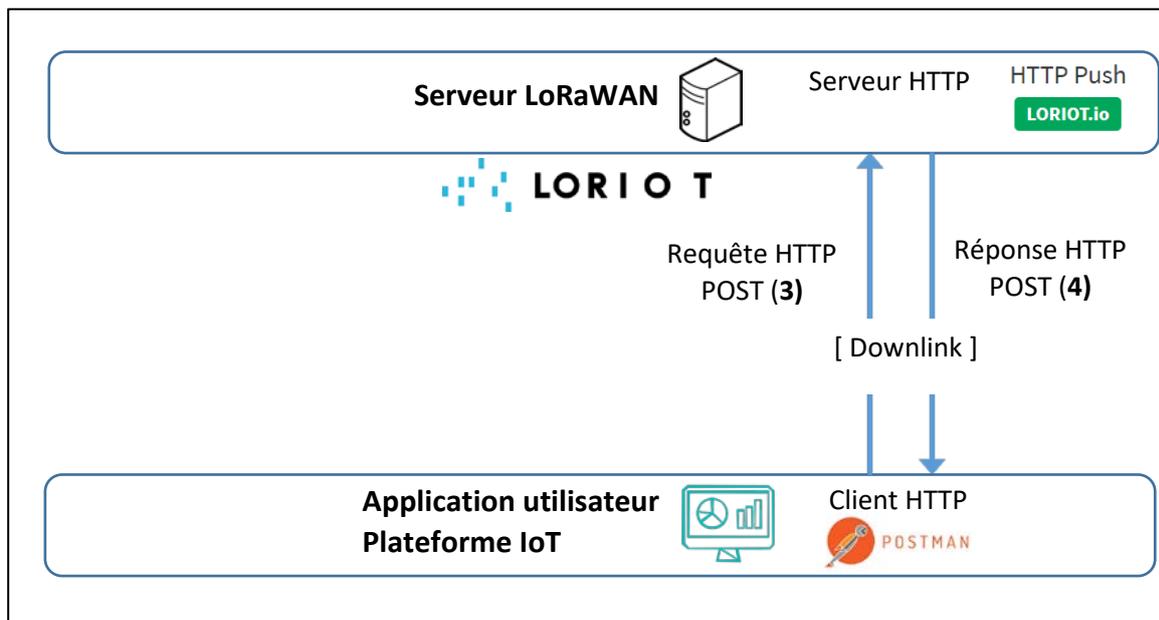


Figure 115: Downlink HTTP POST avec LORIoT

La mise en place du serveur est très simple puisqu'il existe déjà dans tous les serveurs LoRaWAN existants (LORIoT, Actility, TTN, ...) et il est déjà en fonctionnement. Il n'y a donc rien à faire.

Pour le client HTTP POST, nous allons utiliser POSTMAN pour générer la requête. Vous devrez vérifier sur la documentation de votre serveur le format de la requête HTTP POST.



La démarche pour les autres serveurs LoRaWAN est similaire. Vous trouverez sur notre site web www.univ-smb.fr/lorawan le format HTTP POST pour les autres serveurs.

Pour LORIoT :

➔ **POSTMAM > Importation > Brut**

```
curl --location --request POST 'https://eu1.loriot.io/1/rest' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer <API TOKEN>' \
--header 'Content-Type: text/plain' \
--data-raw '{
"cmd": "tx",
"EUI": "XXXXXXXXXXXXXXXXXXXX",
"port": XX,
"confirmed": false,
"data": "ABCDEF",
"appid": "XXXXXXX"
}'
```

Avec les modifications suivantes :

- **<API TOKEN>** : Application > Access Token > Generate authentication token
- **"EUI"** : DevEUI of the Device
- **"appid"** : LORIoT Application ID (4 bytes)
- **"confirmed"**: true or false
- **"data"**: Hexadecimal data to send

7.4 Présentation du protocole MQTT

7.4.1 Aperçu du protocole MQTT

MQTT est un protocole léger conçu pour l'Internet des objets. Plutôt que l'architecture classique client/serveur qui fonctionne avec des requêtes/réponses, MQTT est basé sur un modèle Publisher/Subscriber. La différence est importante, car elle évite de devoir demander (requête) des données dont on ne sait pas quand elles arriveront. Une donnée sera directement transmise aux Subscribers dès qu'elle aura été reçue dans le Broker (serveur central). Pour recevoir les données appartenant à un Topic, le Subscriber doit d'abord souscrire à un Topic.

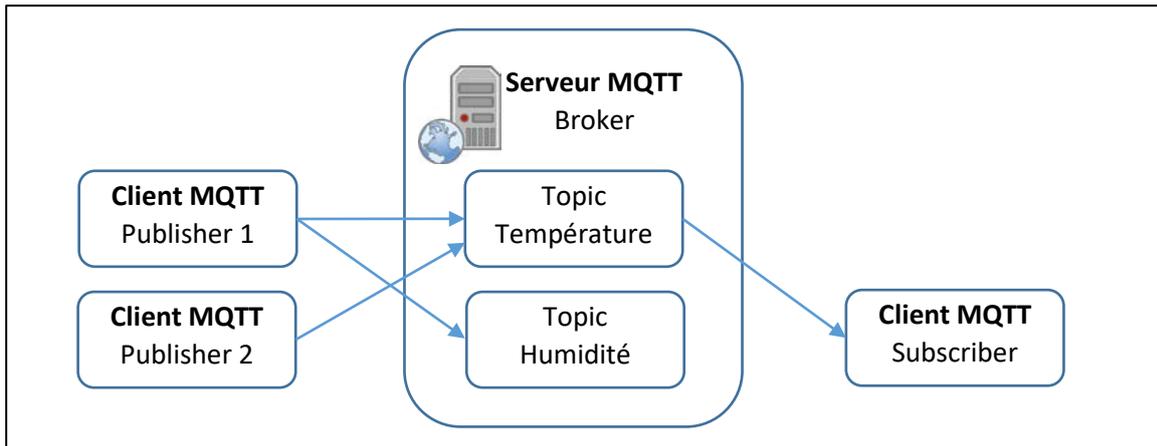


Figure 116: Modèle Publisher/Subscriber du protocole MQTT

MQTT est un protocole basé sur TCP dont la représentation en couche est la suivante :

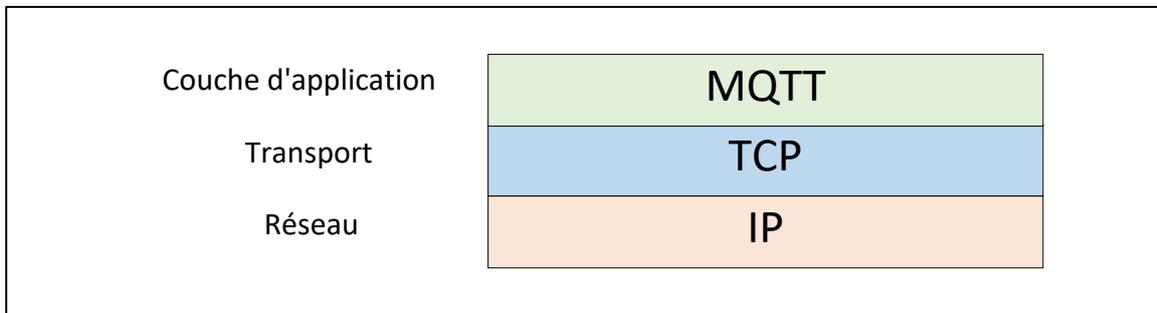


Figure 117: Protocoles utilisés pour la communication en MQTT

Cela peut être vérifié par une capture de trame sur Wireshark.

```
> Ethernet II, Src: Raspberr_f3:03:41 (b8:27:eb:f3:03:41), Dst: Dell_7d:b5:7e (10:65:30:7d:b5:7e)
> Internet Protocol Version 4, Src: 192.168.0.200, Dst: 192.168.0.11
> Transmission Control Protocol, Src Port: 1883, Dst Port: 62454, Seq: 15, Ack: 5, Len: 4
> MQ Telemetry Transport Protocol, Publish Release
```

Figure 118: Capture d'une trame MQTT avec Wireshark

Notez que le port TCP utilisé pour le protocole MQTT (non crypté) est 1883.



Les Publishers et les Subscribers n'ont pas besoin de se connaître.
Les Publishers et les Subscribers ne doivent pas nécessairement fonctionner en même temps.

7.4.2 Connexion au Broker MQTT

Nous allons nous concentrer sur les options de connexion qui permettront de gérer la qualité de service (QoS). Pour se connecter, un client MQTT envoie deux informations importantes au broker:

Un nombre appelé "keepAlive" : Il s'agit de la période la plus longue pendant laquelle le client Publisher ou Subscriber peut rester silencieux. Après cela, il sera considéré comme déconnecté.

Un booléen "cleanSession" : Lorsque le client et le Broker sont brièvement déconnectés (au-delà du keepAlive annoncé), on peut se demander ce qui va se passer lorsque le client se reconnectera.

- cleanSession = True. La connexion est non persistante. Les messages non transmis sont perdus quel que soit le niveau de QoS (**Quality of Service**).
- cleanSession = False. La connexion est persistante. Les messages non transmis seront éventuellement retransmis, en fonction du niveau de QoS. Voir le chapitre 7.4.4.

7.4.3 Qualité de service pendant une connexion unique

Lorsque le client se connecte au Broker, il est possible de choisir le niveau de QoS. Nous parlons ici du cas d'une connexion unique, entre le moment où le client se connecte et le moment où :

- Le client ferme explicitement sa connexion.
- Il n'a montré aucun signe de vie pendant le temps de "keepAlive".

Dans ce cas, nous avons la qualité de service suivante :

QoS 0 "Au plus une fois" : Le niveau de QoS 0 est "sans acquittement". Le Publisher envoie chaque message une seule fois au Broker. Le Broker envoie chaque message une seule fois aux Subscribers. Ce mécanisme **ne garantit pas** la réception des messages MQTT.

QoS 1 "Au moins une fois" : Le niveau de QoS 1 est "avec acquittement". Le Publisher envoie chaque message au Broker et attend sa confirmation. De la même manière, le Broker envoie chaque message à ses Subscribers et attend leur confirmation. Ce mécanisme **garantit** la réception d'au moins un message MQTT.

Cependant, si les accusés de réception n'arrivent pas à temps, ou s'ils sont perdus, la retransmission du message original peut donner lieu à un message en double. Le dernier niveau de QoS permet d'éviter cela.

QoS 2 "Exactement une fois" : Le niveau de QoS 2 est "garanti une fois". Le Publisher envoie un message au Broker et attend la confirmation. Le Publisher donne alors l'ordre de diffuser le message et attend la confirmation. Ce mécanisme **garantit** que, quel que soit le nombre de fois où un message est retransmis, il **ne sera délivré qu'une seule** fois.

La figure ci-dessous montre les trames transmises pour chaque niveau de QoS.

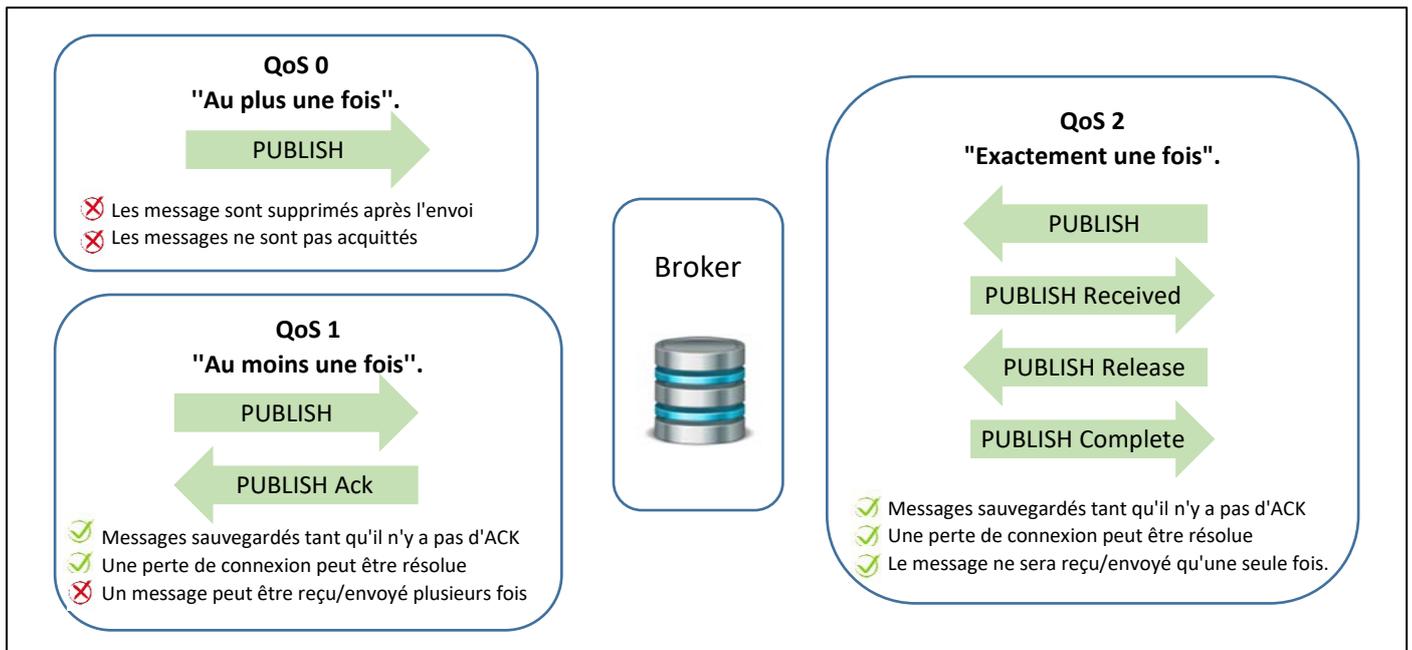


Figure 119: Qualité de service pour le protocole MQTT

La Figure 120 montre les trois trames QoS capturées dans Wireshark.

Source	Destination	Protocol	Length	Info
192.168.0.200	192.168.0.11	MQTT	66	Publish Message [test]

Source	Destination	Protocol	Length	Info
192.168.0.200	192.168.0.11	MQTT	68	Publish Message (id=4) [test]
192.168.0.11	192.168.0.200	MQTT	58	Publish Ack (id=4)

Source	Destination	Protocol	Length	Info
192.168.0.200	192.168.0.11	MQTT	68	Publish Message (id=5) [test]
192.168.0.11	192.168.0.200	MQTT	58	Publish Received (id=5)
192.168.0.200	192.168.0.11	MQTT	60	Publish Release (id=5)
192.168.0.11	192.168.0.200	MQTT	58	Publish Complete (id=5)

Figure 120: Capture de trame avec QoS = 0, puis QoS = 1, puis QoS = 2

Bien entendu, une meilleure QoS augmente la charge du réseau :

- Une trame pour QoS 0.
- Deux trames pour QoS 1.
- Quatre trames pour QoS 2.

7.4.4 Qualité du service après une reconnexion

Que deviennent les messages publiés sur le Broker lorsqu'un ou plusieurs Subscribers sont temporairement injoignables ? Il est possible de sauvegarder les messages qui ont été publiés sur le Broker afin de les retransmettre lors de la prochaine connexion. Cette possibilité de sauvegarde des messages doit être activée lorsque le client se connecte au Broker (cleanSession = 0). La connexion sera alors persistante, c'est-à-dire que le Broker sauvegardera tous les messages qui n'ont pas atteint tous les Subscribers.

Le Tableau 25 résume l'effet du Flag cleanSession et du niveau de QoS lorsqu'il y a une reconnexion du client.

Clean Session Flag	Subscriber QoS	Publisher QoS	Comportement
True (= 1)	0 / 1 / 2	0 / 1 / 2	Messages perdus
False (= 0)	0	0 / 1 / 2	Messages perdus
False (= 0)	0 / 1 / 2	0	Messages perdus
False (= 0)	1 / 2	1 / 2	Tous les messages sont retransmis

Tableau 26 : Valeur QoS et Flag cleanSession

7.4.5 Les Topics du protocole MQTT

Un Topic est une hiérarchie de chaînes de caractères séparées par la barre oblique "/" comme séparateur. La Figure 121 et le Tableau 26 est un exemple de hiérarchie avec la liste des Topics associés.

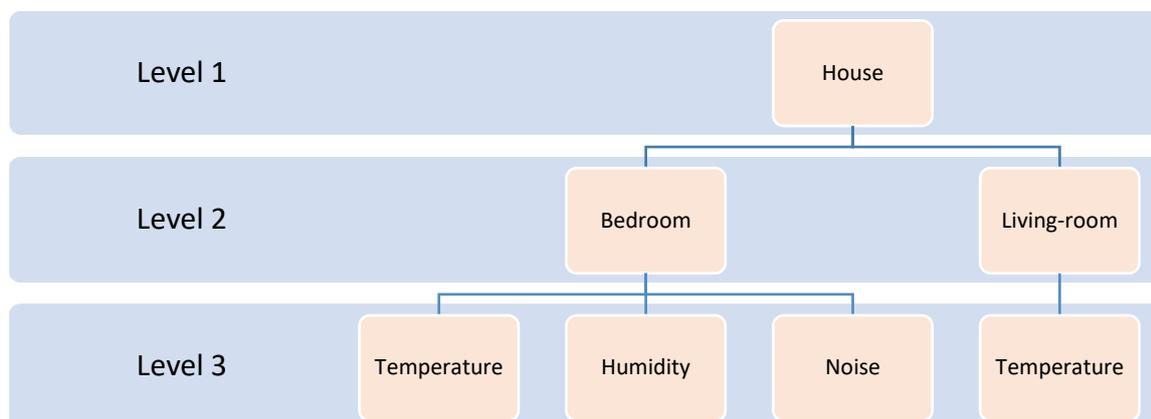


Figure 121: Exemple de hiérarchie de Topics MQTT

Nom du Topic	Détail du Topic
House/Bedroom/Temperature	The Temperature of the Bedroom in the House .
House/ Bedroom /Noise	The Noise of the Bedroom in the House .
Home/Living Room/Temperature	The Temperature of the Living Room in the House .

Tableau 27 : Exemple de Topics

Un client peut souscrire (ou se désabonner) à plusieurs branches de la hiérarchie en utilisant des caractères génériques qui couvrent plusieurs Topics. Il existe deux caractères génériques :

- Le signe plus "+" remplace toute chaîne de caractères du même niveau.
- Le signe dièse "#" remplace toute chaîne de caractères sur tous les niveaux suivants. Il doit être placé à la fin.

Nom du Topic	Détail du Topic
Home/+ /Temperature	The Temperature of all Rooms in the House .
House/#	All measurements of all Rooms in the House .

Table 28: Exemple de Topics utilisant des caractères de remplacement

7.4.6 La configuration d'un Broker MQTT

Pour comprendre le fonctionnement du protocole MQTT, nous mettons en place une infrastructure MQTT simple avec un client Publisher, un Broker et un client Subscriber. Il y a beaucoup de Broker et de clients MQTT disponibles. Pour ce test, nous utilisons :

- Le **Broker** Mosquitto.
- Le client **Subscriber** MQTT Box (pour Windows).

- Le client **Publisher** MQTT Box (pour Windows).

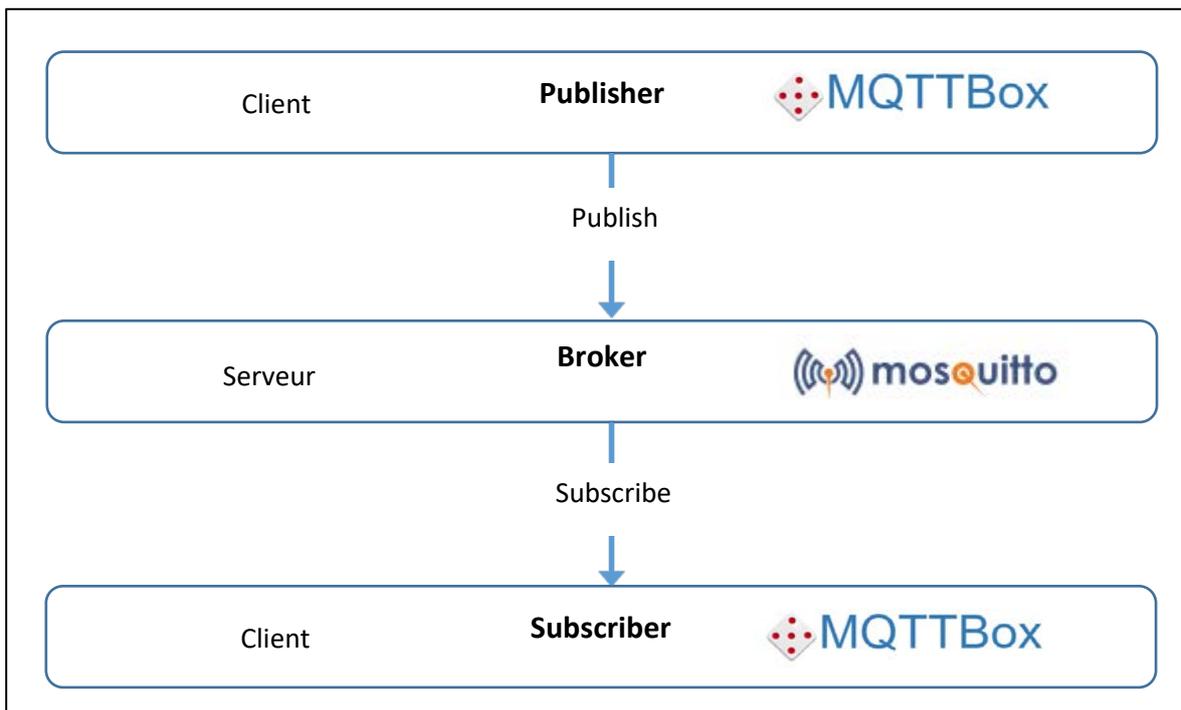


Figure 122: Infrastructure de test du protocole MQTT

Le Broker MQTT est commun à tous. Nous pouvons soit le configurer sur une machine locale, soit utiliser un Broker MQTT public de test. Dans notre cas, nous utiliserons le Broker MQTT public Mosquitto.

7.4.7 Configuration d'un Publisher et d'un Subscriber MQTT

Nous utilisons le client MQTT MQTTBox.

- ➔ Lancez le logiciel MQTTBox.
- ➔ **MQTTBox > Create MQTT Client.**

Un client MQTT doit connaître l'adresse du Broker :

- **Protocole** : mqtt / tcp
- **Host** : test.mosquitto.org

MQTT Client Name <input type="text" value="mosquito public"/>	MQTT Client Id <input type="text" value="5e31ac5f-50ef-4015-8979-de30f6f"/> <input type="button" value="↻"/>
Protocol <input type="text" value="mqtt / tcp"/> ▼	Host <input type="text" value="test.mosquitto.org"/>

Figure 123: Configuration d'un client MQTT dans MQTT Box

Vous pouvez maintenant vous abonner sur un Topic de votre choix et publier sur le même Topic. Le Subscriber devrait recevoir les données. La Figure 124 montrent un Publisher (à gauche) qui envoie

des données "test" sur le Topic "lorawan". Le Subscriber (à droite) reçoit les données "test" car il a souscrit au Topic "lorawan".

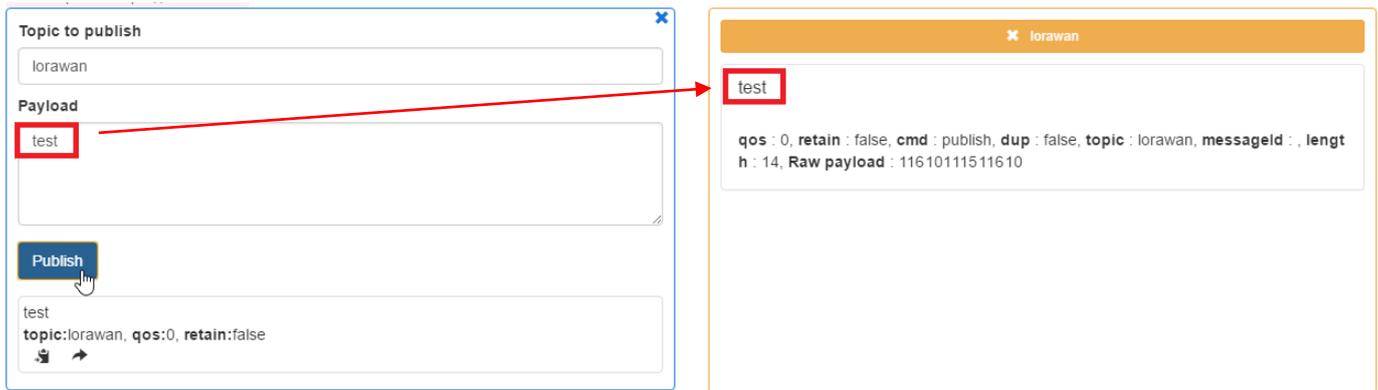


Figure 124: Test du Mosquitto Broker avec le client MQTTBox

7.5 Exportation des données avec le protocole MQTT

Il existe plusieurs possibilités pour utiliser le protocole MQTT avec notre serveur LoRaWAN. Cela dépend de qui est le Broker et de qui est le Publisher/Subscriber. Nous verrons les deux cas mais certains sont plus faciles à mettre en place que d'autres.

7.5.1 Serveur LoRaWAN en tant que Broker MQTT

La première architecture est simple et est représentée sur la Figure 125.

- Le serveur LoRaWAN est le Broker MQTT.
- Vous devez souscrire **(1)** au Topic approprié pour recevoir des données.
- Vous devez publier **(2)** sur le Topic approprié pour envoyer des données.

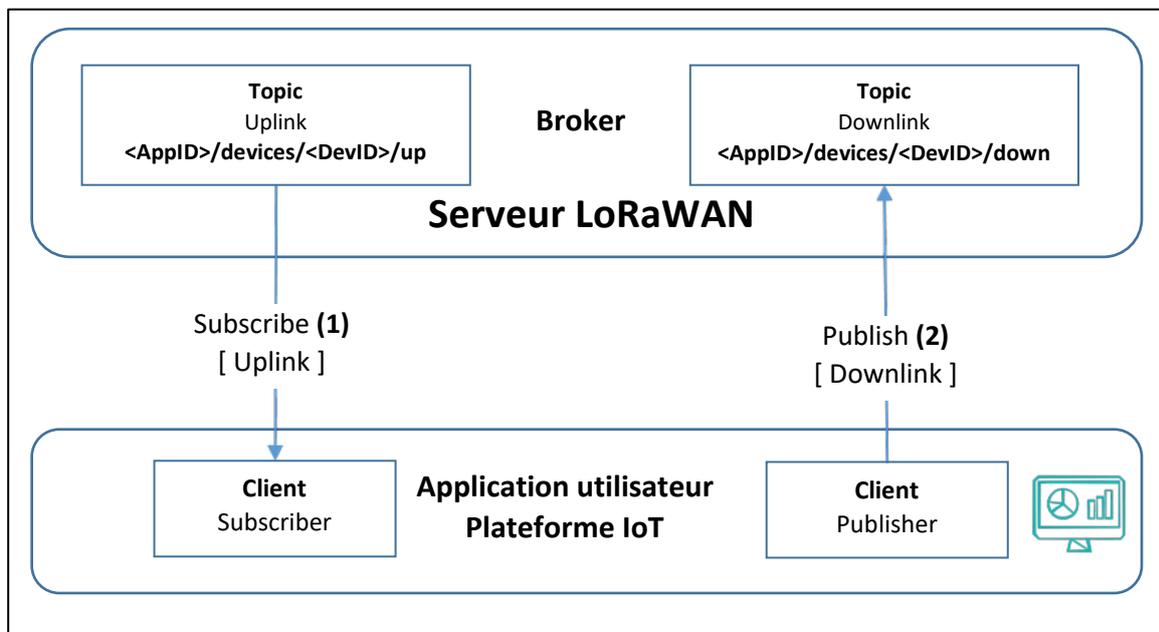


Figure 125: Connexion MQTT avec le serveur LoRaWAN en tant que Broker

Les paramètres de configuration que votre client MQTT doit connaître se trouvent dans la documentation de votre serveur LoRaWAN :

- L'URL du Broker MQTT exposé sur votre serveur LoRaWAN.
- Le nom d'utilisateur.
- Le mot de passe.
- Le Topic à souscrire lorsque vous souhaitez recevoir des données.
- Le Topic auquel il faut publier lorsque vous voulez envoyer des données.

Vous trouverez sur notre site web www.univ-smb.fr/lorawan, les paramètres de configuration pour les serveurs LoRaWAN qui peuvent agir en tant que Broker MQTT.

7.5.2 Le serveur LoRaWAN en tant que client MQTT

Cette deuxième architecture de réseau est représentée à la Figure 126.

- Un Broker MQTT est placé entre le serveur LoRaWAN et l'Application Utilisateur.
- Le serveur LoRaWAN publie **(1)** sur le Topic approprié pour envoyer des données au Broker.
- Vous devez souscrire **(2)** au Topic approprié sur le Broker pour recevoir des données.
- Vous devez publier **(3)** au Topic approprié sur le Broker pour envoyer des données.
- Le serveur LoRaWAN souscrit **(4)** au Topic approprié pour recevoir les données du Broker.

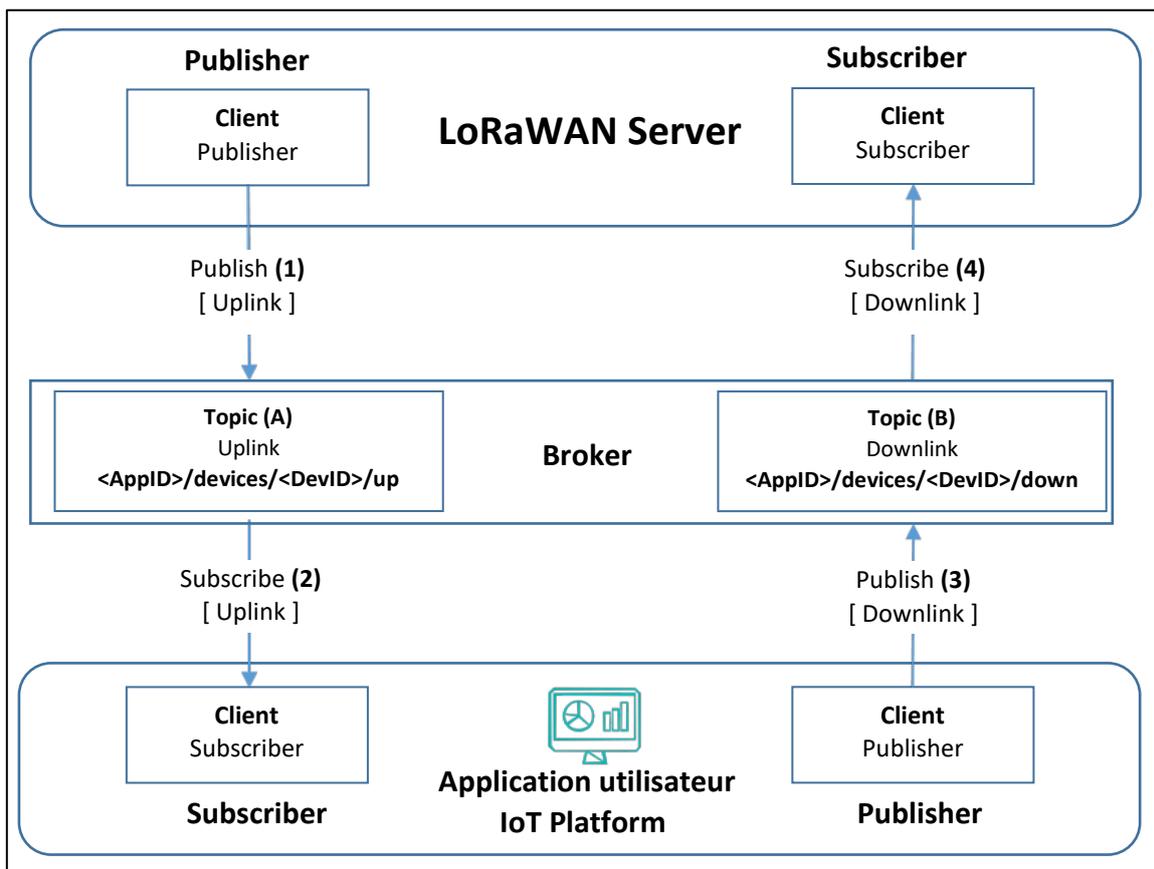


Figure 126: Connexion MQTT avec le serveur LoRaWAN comme client MQTT.

Les paramètres de configuration pour le client MQTT "LoRaWAN Server" et "Application Utilisateur" dépendent du Broker choisi, mais vous avez toujours besoin des informations suivantes :

- L'URL du Broker MQTT.

- Le nom d'utilisateur pour se connecter à votre Broker.
- Le mot de passe pour se connecter à votre Broker.
- Le Topic que vous choisissez pour la réception des données [**(A)** dans la Figure 126].
- Le Topic sur lequel il faut publier pour envoyer des données [**(B)** dans la Figure 126].



Dans cet exemple, nous utilisons le serveur LoRaWAN d'Activity (v3.6.0), le Broker public "HiveMQ" comme intermédiaire et le client MQTTBox pour simuler l'Application Utilisateur.

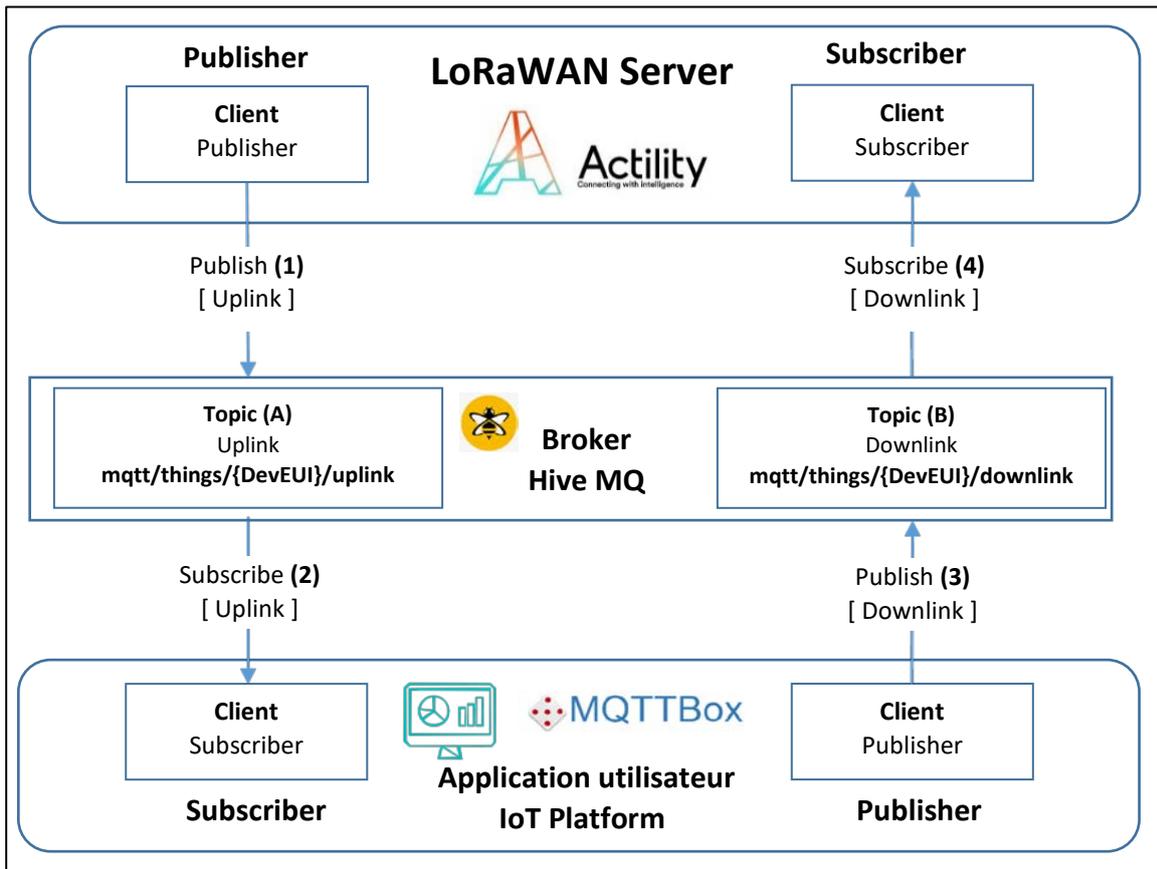


Figure 127: Test du serveur LoRaWAN en tant que client MQTT

Dans Activity, nous devons d'abord créer et configurer le client MQTT : **Activity > Connexions > TPX > MQTT** . Puis vous entrez les paramètres suivants :

- Hostname: **broker.hivemq.com:1883**
- MQTT Username: test
- MQTT Password: test
- Uplink topic pattern: `mqtt/things/{DevEUI}/uplink`
- Downlink topic pattern: `mqtt/things/{DevEUI}/downlink`

Vous devez changer **{DevEUI}** par votre Device EUI.

- ➔ Nous allons maintenant attribuer cette nouvelle connexion à votre appareil : **Activity > Device > List > Your Device > Connexions > La connexion MQTT que vous venez de créer.**

Nous créons maintenant le client MQTT.

- ➔ Avec MQTT Box, créez un client avec les mêmes paramètres que ceux utilisés ci-dessus.
- ➔ Créez un Subscriber sur le Topic "mqtt/things/{DevEUI}/uplink". Vous devriez recevoir des données arrivant en uplink depuis votre Device.
- ➔ Créez un Publisher sur le Topic "mqtt/things/{DevEUI}/downlink". Puis entrez le message suivant pour envoyer une trame de downlink :

```
{
  "DevEUI_downlink" : {
    "Heure" : "2021-12-12T15:38:46.882+02:00",
    "DevEUI" : "Your-Device-EUI",
    "FPort" : "1",
    "payload_hex" : "9e1c4852512000220020e3831071"
  }
}
```

Vous devriez recevoir le message de downlink sur votre Device LoRaWAN.

7.6 Utilisation d'une plateforme IoT

Une plateforme IoT n'est pas spécifique au protocole LoRaWAN. Au contraire, elle combine plusieurs protocoles et des réseaux hétérogènes en un seul endroit afin d'aider les entreprises à gérer leurs actifs et à créer une application pour l'utilisateur final.

Il existe des centaines de plateformes IoT, qui présentent toutes des avantages spécifiques et ciblent des marchés différents. Les connexions entre le serveur LoRaWAN et la plateforme IoT sont souvent faciles et la plupart du temps, elles supportent HTTP POST et MQTT. Certaines plateformes IoT ont établi un partenariat avec le serveur LoRaWAN afin de faciliter la connexion. Vous pouvez voir sur la figure ci-dessous quelques-uns des connecteurs disponibles dans ACTILITY.

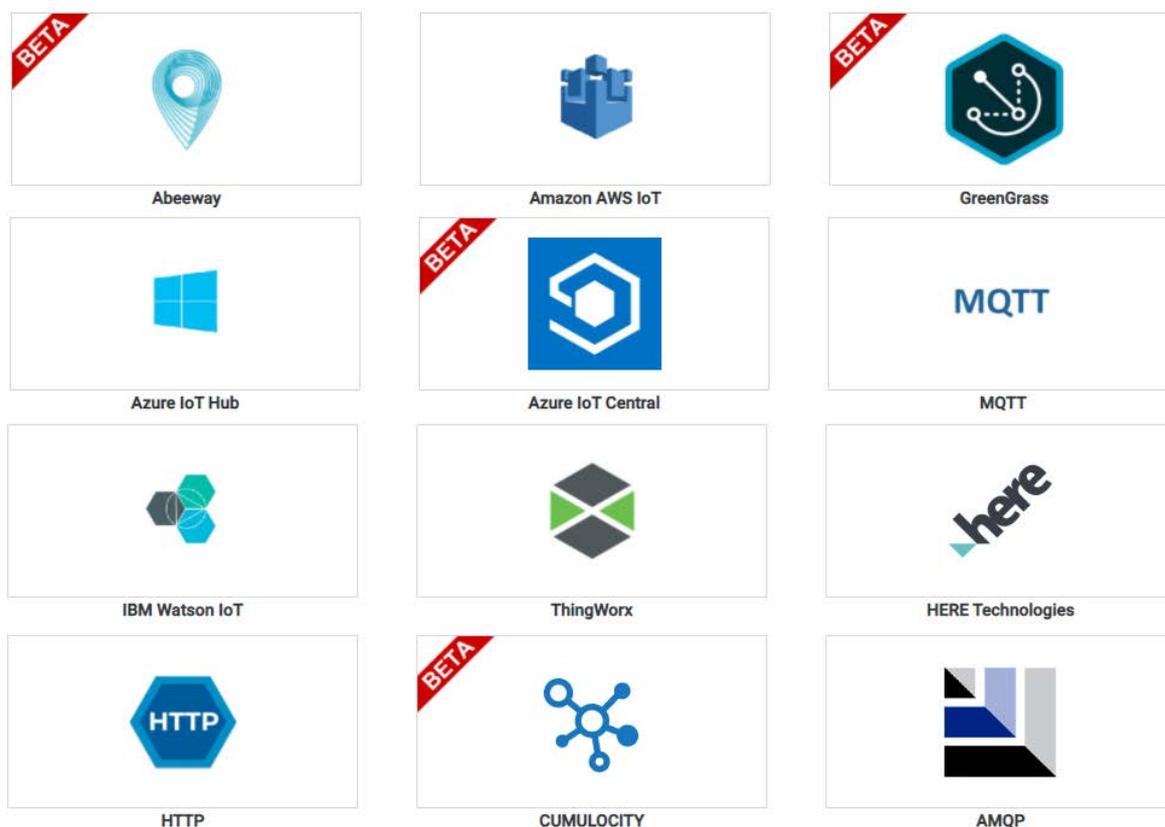


Figure 128: Connexions d'ACTILITY à des plateformes IoT

Voici quelques exemples de plateformes IoT :

- Vertical M2M - [[CommonSenseIoT Platform](#)] : Editeur de logiciels indépendant spécialisé dans les solutions IoT industrielles et B2B avec plus de 13 ans d'expérience sur le marché de l'IoT.
- ioThink - [[KHEIRON IoT suite](#)] : KHEIRON est une solution IoT flexible et personnalisée, destinée aux intégrateurs de systèmes, aux fabricants de Devices, aux constructeurs de machines et aux opérateurs de réseaux.



Les plateformes IoT sont généralement disponibles sous forme de service cloud ou installées sur les serveurs de la société.

Nous allons essayer Vertical M2M (CommonSense IoT Platform) et le connecter à notre serveur LoRaWAN. CommonSense IoT Platform permet aux territoires, aux services publics, aux opérateurs télécoms et aux clients industriels de déployer des projets IoT à grande échelle en résolvant les 3 problèmes critiques auxquels ils sont confrontés :

- Gérer l'hétérogénéité des Devices et des technologies IoT : LoRaWAN mais aussi beaucoup d'autres telles que Sigfox, NB-IoT, LTE-M, cellulaire 2G-3G-4G-5G...
- Gérer en toute sécurité le parc de Devices : fonctions avancées de supervision, d'alertes, de commande, de rapport et de gestion.
- Connecter facilement toutes les données et tous les Devices IoT aux applications commerciales des clients : cela est possible grâce à une solution lowcode/nocode (appelée module IoT APP STUDIO) permettant de concevoir et de créer des applications IoT entièrement personnalisables pour des marchés verticaux tels que smartcity, smartwater ou smartbuilding.

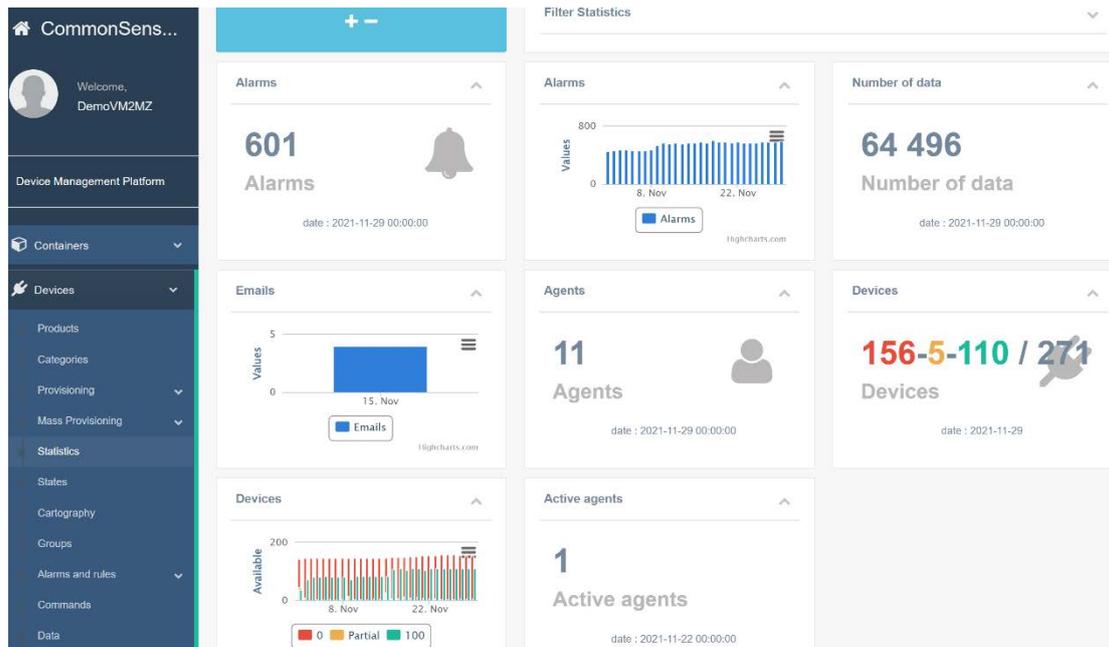


Figure 129: Plateforme IoT de CommonSense

8 Concevoir votre propre Device LoRaWAN

Dans ce chapitre, nous allons énumérer toutes les architectures de Devices LoRaWAN. Bien sûr, chacune d'entre elles a ses propres avantages et inconvénients. Dans la dernière section, nous récapitulerons toutes les architectures vues dans ce chapitre.

Un Device LoRaWAN a besoin :

1. D'un programme pour l'application utilisateur.

Ce programme s'occupe des interfaces avec les capteurs, du calcul, du réveil et de la transition vers les modes basses consommation, de l'interface utilisateur (afficheur, bouton poussoir, Led...) et de tout autre élément répondant au besoin du client. Celui-ci n'est pas lié au protocole LoRaWAN.

2. D'une Stack de protocole LoRaWAN.

C'est là que tout le protocole LoRaWAN exécute la routine séquentielle de la couche LoRa MAC pour effectuer une transmission de données. Si le Device est certifié LoRaWAN, il doit respecter les spécifications pour garantir une transaction correcte et sécurisée. La Stack de protocole est paramétrable en fonction de la région où le Device va opérer.

3. Une interface radio LoRa.

Un Transceiver génère le signal Radiofréquence en respectant la modulation LoRa. Le Transceiver peut couvrir plusieurs régions du monde en fonction des fréquences qu'il est capable de générer, mais la conception de l'antenne est spécifique à chaque bande.

8.1 Les Stacks LoRaWAN disponibles

Si nous utilisons un Device qui intègre déjà une Stack LoRaWAN, alors nous n'avons pas à nous soucier de son intégration dans notre composant. En revanche, pour toute autre conception, nous devons l'intégrer nous-mêmes. Voici une brève description des Stacks LoRaWAN disponibles :

1. La Stack la plus connue est développée par SEMTECH. Cette Stack s'appelle [LoRa MAC Node](#) et est disponible pour tous les microcontrôleurs. Elle est très bien maintenue et suit la spécification LoRaWAN.
2. L'autre Stack bien connue est [LMIC \(LoRa Mac In C\)](#). C'est la Stack préférée lorsqu'on utilise des cartes Arduino.

D'autres Stacks sont disponibles. Par exemple, ST propose sa propre Stack, qui est basée sur le "LoRa MAC Node" de SEMTECH, mais qui a été améliorée pour mieux correspondre aux spécificités des microcontrôleurs STM32. Arm MBED propose également une Stack LoRaWAN, mais elle n'est pas open source.

8.2 Architecture "Microcontrôleur + Transceiver"

8.2.1 Présentation de cette architecture

Dans ce type d'architecture, un microcontrôleur gère à la fois la Stack LoRaWAN et le programme de l'application utilisateur. Cela nécessite de disposer d'une Stack LoRaWAN.

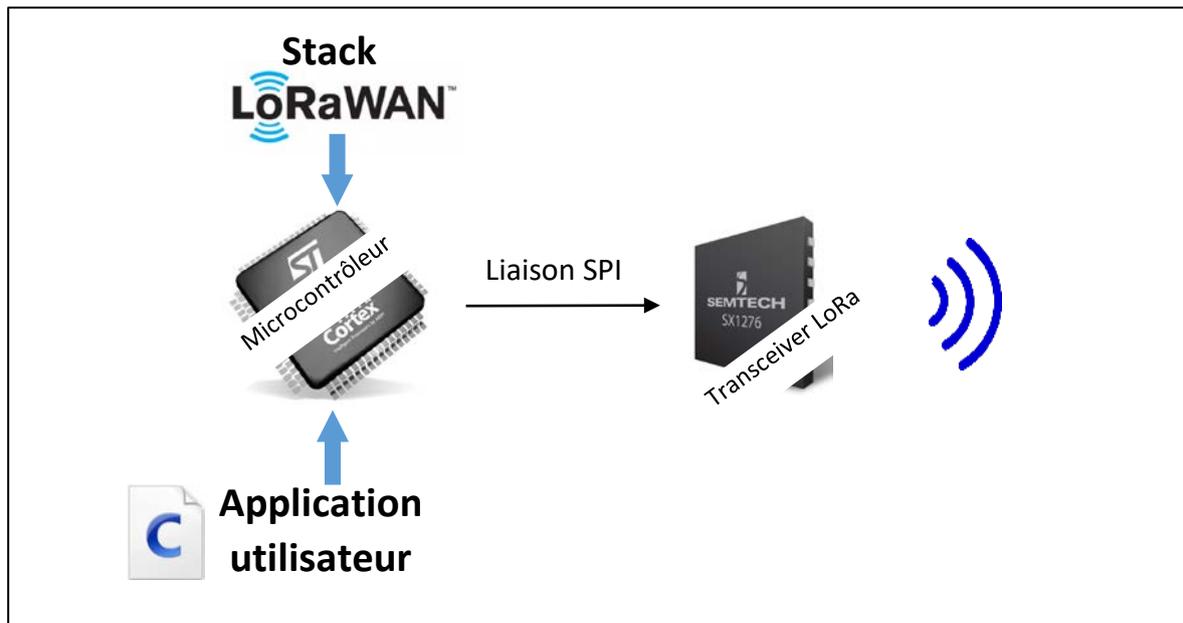


Figure 130: Device LoRaWAN avec Microcontrôleur + Transceiver

Le SX1262 (SEMTECH) est un exemple de Transceiver LoRa. Il gère la partie physique du protocole : Modulation, détection de préambule.... La gestion complète du protocole LoRaWAN est réalisée par la Stack logicielle implémentée dans le microcontrôleur. Le microcontrôleur pilote le Transceiver (par exemple le SX1262) avec un bus SPI.

Ce choix est plutôt intéressant et optimisé en termes de consommation électrique, car il n'y a qu'un seul microcontrôleur qui gère tout. En revanche, c'est une solution plus complexe car il est nécessaire de se plonger dans la Stack LoRaWAN. En effet, même si l'application utilisateur et la Stack sont bien séparées dans le code, c'est un véritable défi d'être certain que l'une n'interfère pas avec l'autre car elles tournent sur le même MCU en même temps. Il faut toujours faire très attention à ce que l'Application ne prenne pas de ressource nécessaire au fonctionnement du protocole LoRaWAN.

8.2.2 Exemple d'une carte de développement

On peut trouver de nombreuses cartes de développement pour les transceivers SEMTECH sur leur site web. En voici une proposée par ST. La carte de développement P-NUCLEO-LRWAN1 associe un microcontrôleur STM32L073 (Cortex M0+) à un transceiver SX1272.

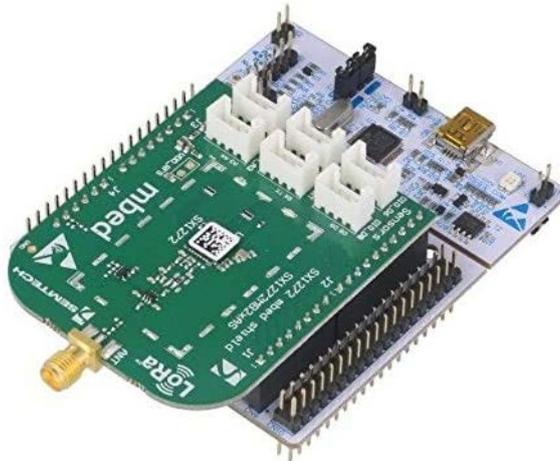


Figure 131: Boîtier P-NUCLEO-LRWAN1

8.2.3 Les solutions de mise en œuvre

Après la période de prototypage, vous pouvez passer à la réalisation de votre propre circuit imprimé. Si vous voulez faire les soudures vous-même, vous aurez besoin d'un atelier raisonnablement équipé. Une solution intéressante est d'utiliser des "breakout boards", qui contiennent déjà le Transceiver et quelques composants utiles.



Figure 132: Exemple d'une breakout board pour le Transceiver SX1276 (NiceRF)

8.3 Architecture "Module LoRaWAN autonome"

8.3.1 Présentation de cette architecture

Dans ce type d'architecture, nous utilisons un module autonome qui comprend les deux :

- Un microcontrôleur (qui contient le programme de l'application + la Stack LoRaWAN).
- Un Transceiver.

Il y a plusieurs composants dans le module. Cependant, le Transceiver n'est pas intégré dans le microcontrôleur comme nous le verrons dans la section 8.4. Du point de vue du programmeur, c'est presque la même chose.

Cette solution a l'avantage de simplifier la partie matérielle de la solution précédente. En revanche, nous avons toujours la proximité du programme de l'application et de la Stack LoRaWAN à gérer. Le module dispose d'un certain nombre de périphériques (ADC, I2C, UART, GPIO...) pour réaliser la connexion du capteur pour l'application utilisateur.

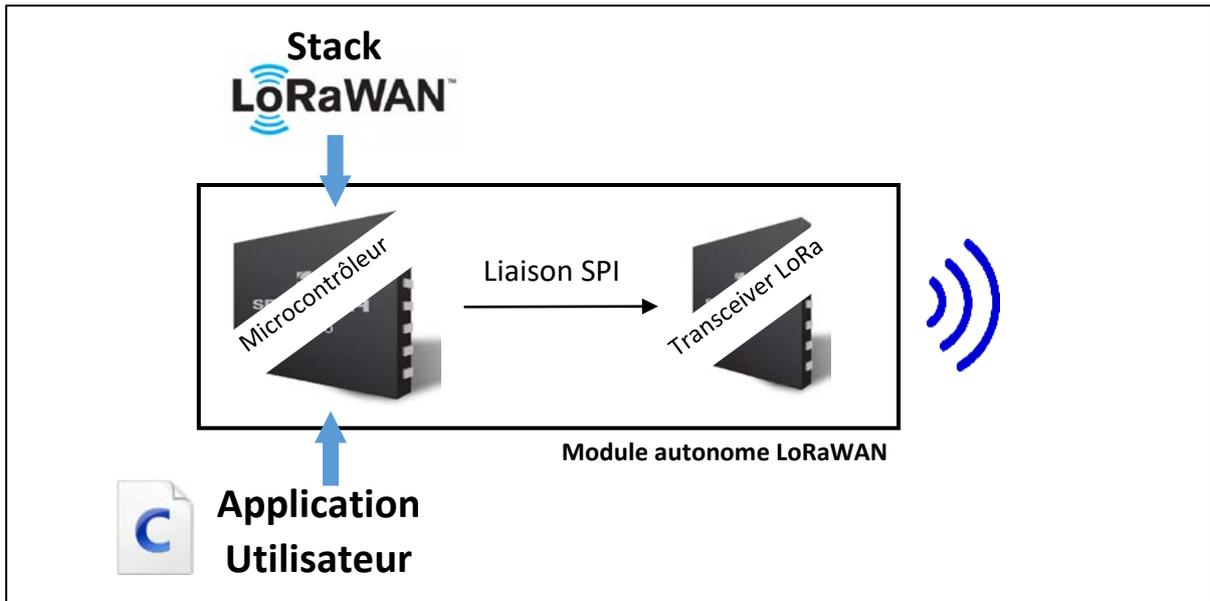


Figure 133: Module autonome LoRaWAN

8.3.2 Exemple de module

Le Murata CMWX1ZZABZ peut fonctionner en mode autonome. Il n'aura besoin d'aucun autre composant pour fonctionner, à l'exception de l'antenne et du circuit d'adaptation d'impédance.



Figure 134: Module intégrant une Stack LoRaWAN et un transceiver LoRa

La Figure 135 montre le schéma du module Murata.

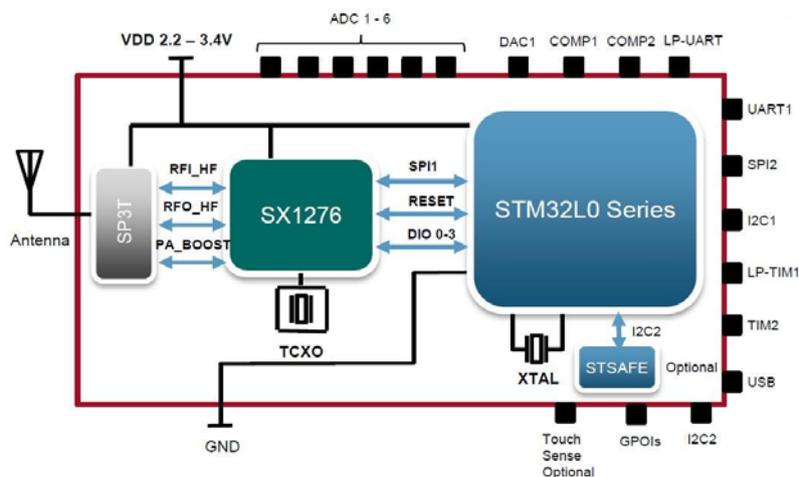


Figure 135: Intérieur du module Murata ABZ

D'autres modules peuvent intégrer un connecteur SMA sur le circuit pour faciliter l'interface avec l'antenne.

8.3.3 Exemple d'une carte de développement

ST dispose d'une carte de développement pour le composant CMWX1ZZABZ. Il s'agit de la carte Discovery B-L072Z-LRWAN1.



Figure 136: Carte Discovery de ST B-L072Z-LRWAN1

8.4 Architecture "Microcontrôleur + LoRaWAN module"

8.4.1 Présentation de cette architecture

Dans ce type d'architecture, le microcontrôleur ne gère que le programme de l'application utilisateur. Un module externe, piloté par une liaison série (RX-TX), gère l'ensemble du protocole LoRaWAN. Ce module est exactement du même type que celui que nous avons vu dans le chapitre précédent 8.3.

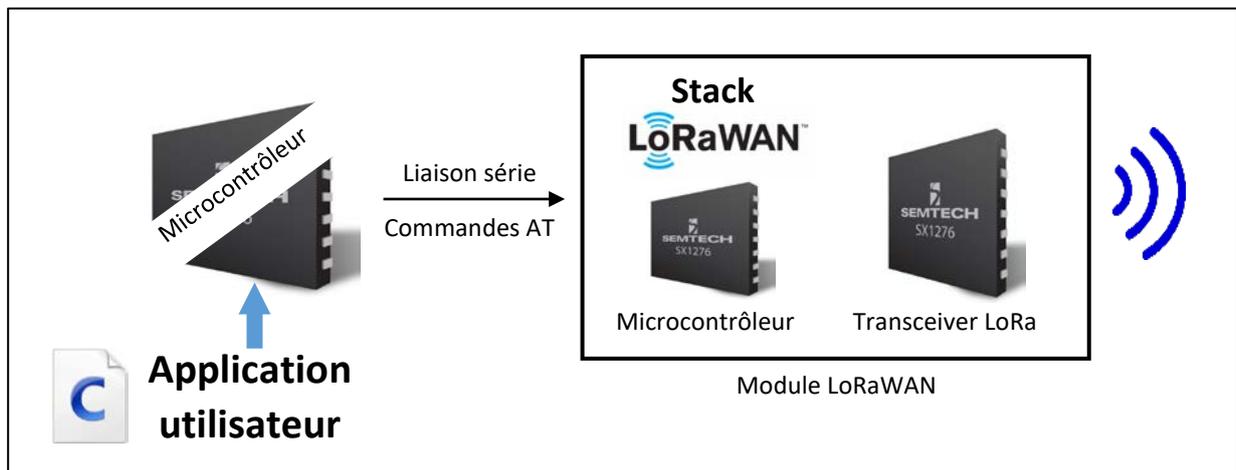


Figure 137: Device LoRaWAN avec microcontrôleur et module LoRaWAN

Nous pouvons choisir n'importe quel microcontrôleur (même 8 bits) car la Stack LoRaWAN s'exécute sur le module et ne prend donc aucune ressource sur le MCU de l'application utilisateur.

8.4.2 Exemple de module LoRaWAN

Voici deux modules couramment utilisés :

- RAK Wireless : RAK3172
- Murata : CMWX1ZZABZ : Il s'agit du même module que celui vu au paragraphe 8.2 mais au lieu d'être autonome, le firmware répond à un ensemble de commandes AT envoyées par un maître en liaison série.



Figure 138: Modules intégrant la Stack LoRaWAN + un Transceiver

Il existe des bibliothèques permettant d'utiliser les commandes AT de façon efficace.

Ce choix a l'avantage considérable de sa simplicité. Nous n'avons pas à gérer quoi que ce soit du protocole LoRaWAN car tout est fait dans le module. En envoyant une simple commande AT, le concepteur du Device n'a plus qu'à se concentrer sur l'application utilisateur.

La contrepartie est évidemment le fait que le système global comporte deux microcontrôleurs : un pour le programme de l'application utilisateur et un pour la gestion du module LoRaWAN. Cela influencera le prix de l'ensemble du système, et bien sûr, sa consommation.

8.4.3 Exemple de carte de développement

Voici un exemple de carte de développement couramment utilisée.



Figure 139: Arduino MKRWAN 1310 : Microcontrôleur ATMEL 32 bits + Module CMWX1ZZABZ

8.5 Architecture "Microcontrôleur LoRaWAN"

8.5.1 Présentation de cette architecture

STMicroelectronics a développé le premier microcontrôleur avec Transceiver LoRa intégré : le STM32WL. Il existe en deux versions : simple cœur (STM32WLEx) ou double cœur (STM32WL5x). Dans cette architecture, la Stack LoRaWAN et le programme de l'application utilisateur sont dans le même microcontrôleur mais peuvent être séparés si nous utilisons la solution double cœur. Il est important de dire que ce composant est construit sur la même puce de silicium.

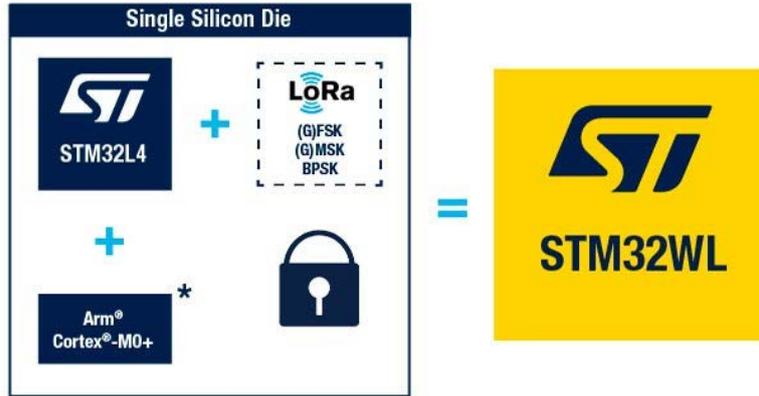


Figure 140: Microcontrôleur double cœur STM32WL

C'est la solution la plus intéressante en termes de coût, de consommation électrique et d'encombrement.

8.5.2 Exemple de carte de développement

STMicroelectronics propose sa propre carte Nucleo pour ce microcontrôleur.

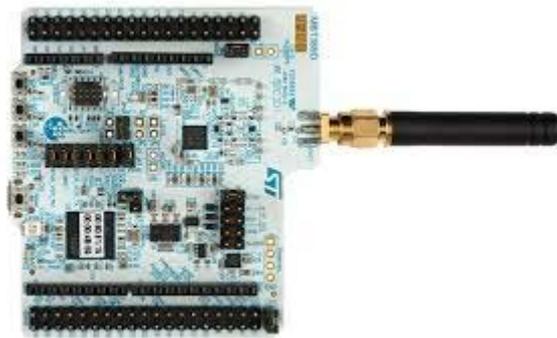


Figure 141: Carte STM32WL Nucleo

A noter que le RAK3172 (vue plus haut) est en fait un STM32WL intégré dans un module pour faciliter son utilisation.

8.6 Résumé des architectures

Un résumé comparatif des solutions est donné dans le Tableau 28. Les évaluations fournies ne sont pas quantifiées.

	Microcontrôleur + Transceiver	Module LoRaWAN autonome	Microcontrôleur + Module LoRaWAN	Microcontrôleur LoRaWAN
Taille	Moyen	Faible	Haut	Très faible
Coût	Haut	Moyen	Très élevé	Faible
Complexité du code	Haut	Haut	Faible	Haut

Tableau 29 : Avantages et inconvénients des différentes architectures

9 Configurer votre propre serveur LoRaWAN

Le réseau LoRaWAN sur lequel nous avons travaillé jusqu'à présent était un réseau hybride (voir chapitre 5.1.4) : nous utilisons notre propre Gateway mais les serveurs LoRaWAN ne nous appartenaient pas. Nous allons maintenant installer notre propre serveur LoRaWAN pour créer un réseau totalement privé (voir chapitre 5.1.2).



Des démonstrations de l'installation sont disponibles sur notre site web www.univ-smb.fr/lorawan pour [ChirpStack](#) et [The Things Stack](#).



Il est également possible de choisir d'autres serveurs LoRaWAN privé non open source dans votre entreprise. Le prix dépend du niveau de la prestation demandée.

9.1 Informations préliminaires

9.1.1 Commander votre propre Gateway

Le passage à un réseau privé n'est possible que si vous disposez de vos propres Gateways (une minimum). Vous devez les reconfigurer pour qu'elles pointent vers les serveurs LoRaWAN que vous avez mis en place.

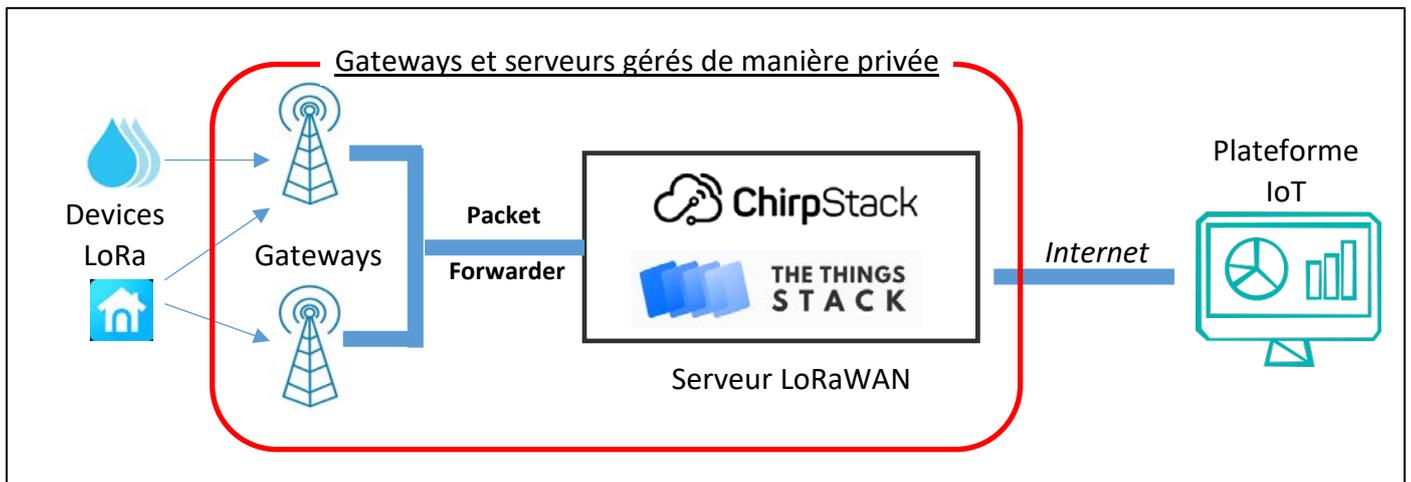


Figure 142: Infrastructure d'un réseau LoRaWAN privé

9.1.2 Vérifier le Packet Forwarder

Certains serveurs LoRaWAN imposent l'utilisation d'un Packet Forwarder spécifique. Pour ChirpStack et The Things Stack, Basic Station est le choix privilégié pour le déploiement. Pour le développement, SEMTECH UDP Packet Forwarder est toujours disponible.

9.1.3 Emplacement de la Gateway et du serveur LoRaWAN

La Gateway doit échanger des données avec votre serveur LoRaWAN. Il existe de nombreuses possibilités :

- La Gateway et le serveur LoRaWAN se trouvent sur l'Internet public. Dans ce cas, ils ont tous deux une adresse IP publique et ils peuvent facilement se connecter ensemble.

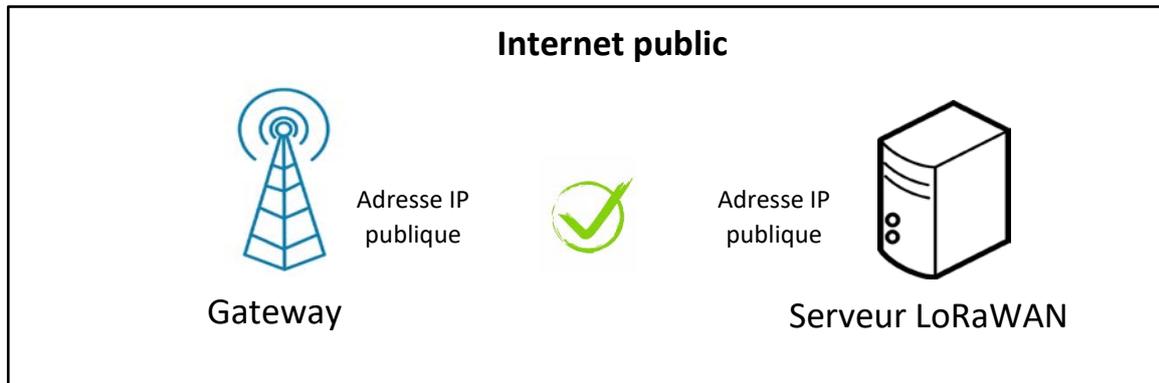


Figure 143: Gateway et serveur LoRaWAN sur l'Internet public

- La Gateway et le serveur LoRaWAN se trouvent dans le même réseau privé. Dans ce cas, ils ont tous deux une adresse réseau commune et ils peuvent facilement se connecter ensemble.

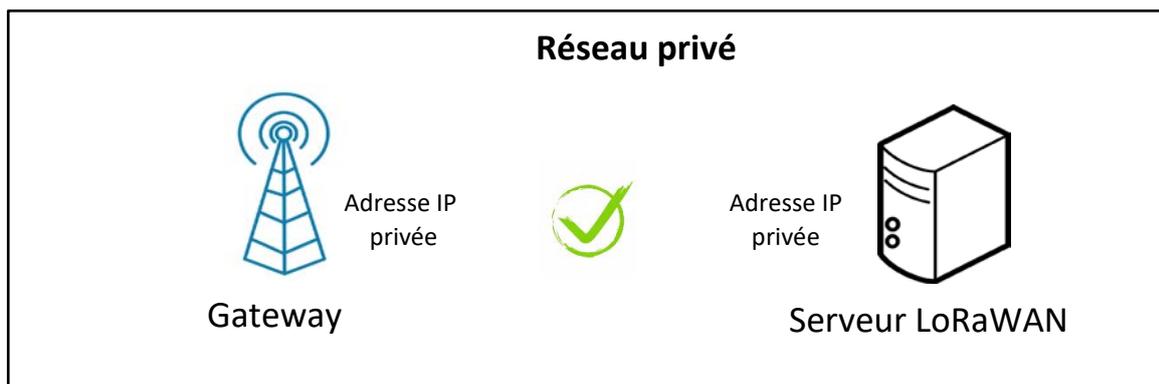


Figure 144: La Gateway et le serveur LoRaWAN sont dans le même réseau privé

- La Gateway est dans un réseau privé et le serveur LoRaWAN est sur l'Internet public. Dans ce cas, le processus de translation de ports du routeur permettra à la Gateway de se connecter au serveur. Une fois que la translation est active sur le routeur, le serveur LoRaWAN peut également envoyer des données à la Gateway. Tout fonctionne.

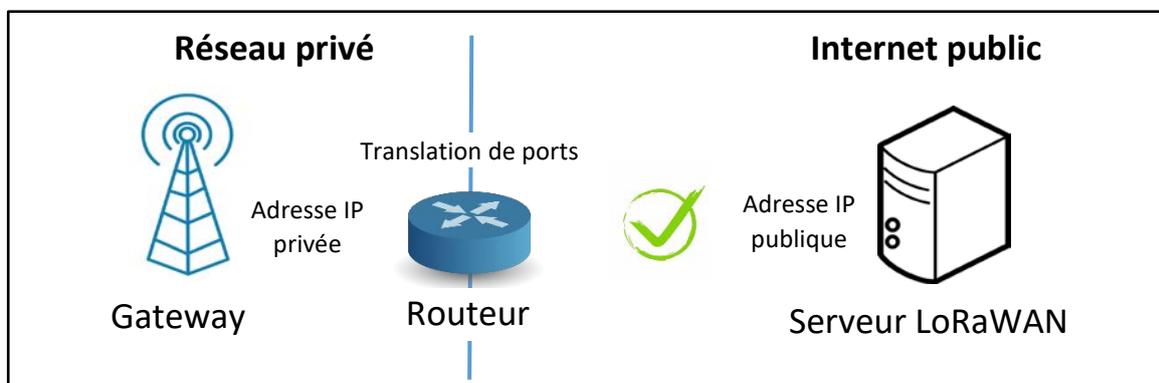


Figure 145: Gateway dans le réseau privé et serveur LoRaWAN sur l'Internet public

- La Gateway est sur l'Internet public et le serveur LoRaWAN est dans un réseau privé. Dans ce cas, **la Gateway ne pourra pas se connecter au serveur si aucune translation de port statique n'est configurée manuellement dans le routeur**. La même situation se produit si la Gateway et le serveur LoRaWAN se trouvent dans deux réseaux privés différents.

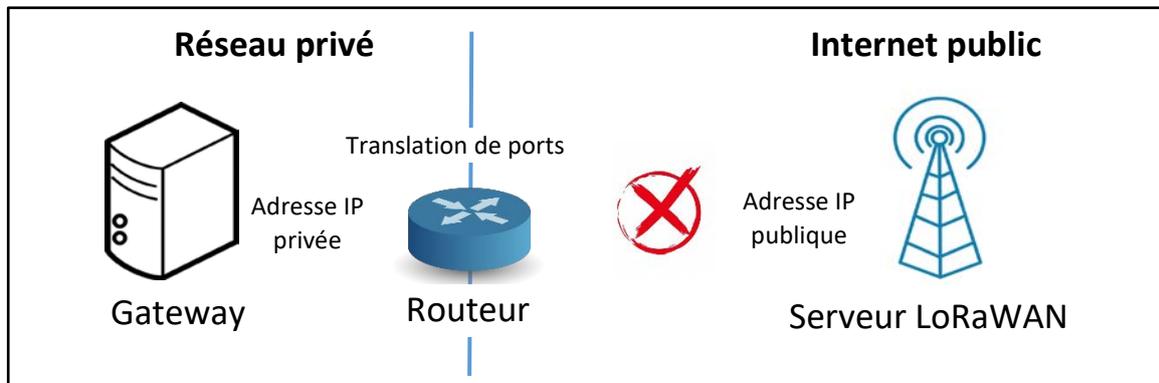


Figure 146: Gateway sur l'Internet public et serveur LoRaWAN dans un réseau privé

9.1.4 Hôte du serveur LoRaWAN

Pour installer notre serveur LoRaWAN, nous avons besoin d'un hôte connecté à Internet et accessible depuis la Gateway. Cela peut être :

- Sur la Gateway elle-même.
- Votre propre PC local (Windows, Linux, MacOS).
- Un ordinateur externe (Raspberry PI, ...).
- Une machine virtuelle (Virtual Box, VMware, ...).
- Un serveur d'un fournisseur (OVH, AWS...).

La diversité des architectures rend l'installation assez difficile à documenter. Il faut donc standardiser le processus. Docker et Docker Compose ont été développés dans ce but car ils permettent d'isoler les services en utilisant des conteneurs, ce qui les rend plus polyvalents :

- Indépendant du système d'exploitation hôte (Linux, MacOS, Windows).
- Indépendant du matériel (architecture ARM, X86).
- Indépendant des autres services installés sur la machine.

Docker permet d'effectuer des installations de manière extrêmement simple, sans avoir à gérer les dépendances et les bibliothèques spécifiques à chaque système d'exploitation (Windows, Linux, MacOS) et aux processeurs utilisés (ARM, X86). Il agira comme une couche d'abstraction qui isolera le service installé du reste du système.

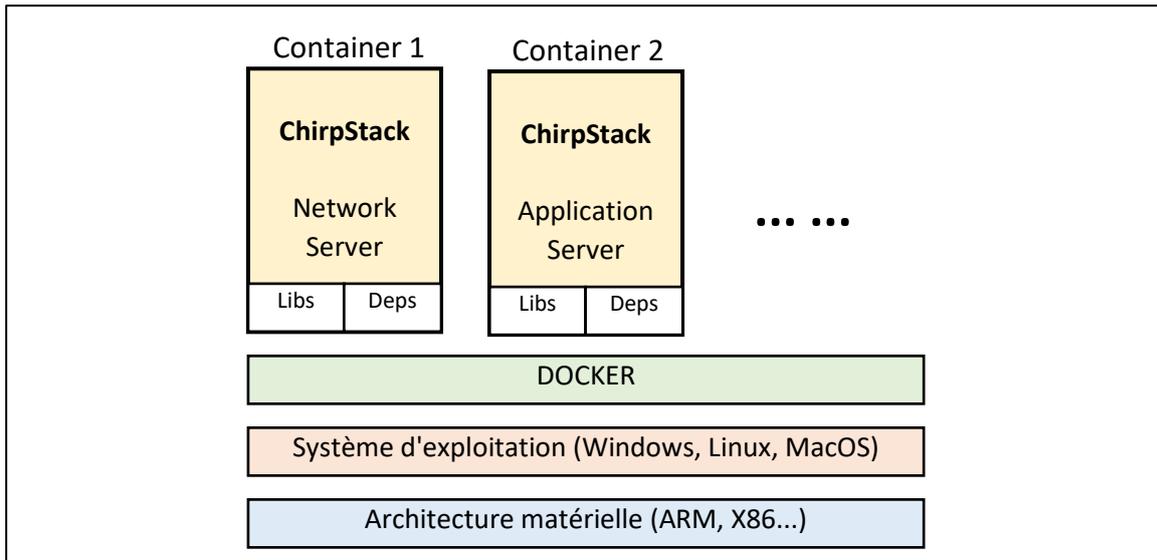


Figure 147: Utilisation de Docker et Docker Compose pour installer un serveur LoRaWAN

L'installation de Docker et de Docker Compose dépend de la plateforme et se fait comme suit :

- ➔ Installation de **Docker** : <https://docs.docker.com/engine/install/>
- ➔ Installer **Docker Compose** : <https://docs.docker.com/compose/install/>

9.2 Serveur LoRaWAN ChirpStack

9.2.1 Le projet ChirpStack

ChirpStack est un projet open-source avec l'architecture simplifiée suivante.

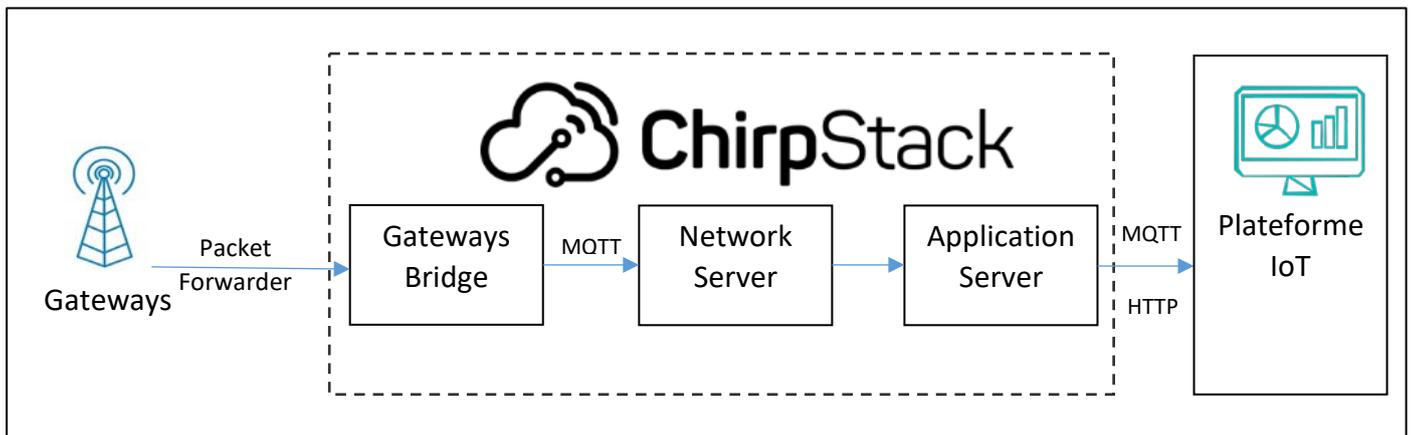


Figure 148: Architecture de ChirpStack

ChirpStack dispose des deux services habituels : **Network Server** et **Application Server**. Il existe également un autre service appelé **Gateway Bridge**. Grâce au Gateway Bridge, ChirpStack est capable de communiquer avec plusieurs Packet Forwarder indépendamment du Network Server. Le Gateway Bridge convertit les différents formats des Packet Forwarder en un protocole MQTT commun utilisé par le Network Server de ChirpStack.

Le Gateway Bridge de ChirpStack peut s'interfacer avec le **SEMTECH UDP Packet Forwarder** ou **Basic Station**.

9.2.2 Configuration de ChirpStack avec Docker

Vous pouvez utiliser Docker si vous installez ChirpStack sur un Raspberry PI, sur un serveur cloud ou sur votre PC local avec n'importe quelle distribution d'OS et n'importe quel processeur. Il n'y a rien de spécial à part le fait de suivre la documentation de ChirpStack : [Documentation ChirpStack](https://www.univ-smb.fr/lorawan).



La démonstration de l'installation de ChirpStack en utilisant Docker est disponible ici en vidéo : www.univ-smb.fr/lorawan.

La Figure 149 montre les services (conteneur Docker) générés avec Docker Desktop sur Windows.

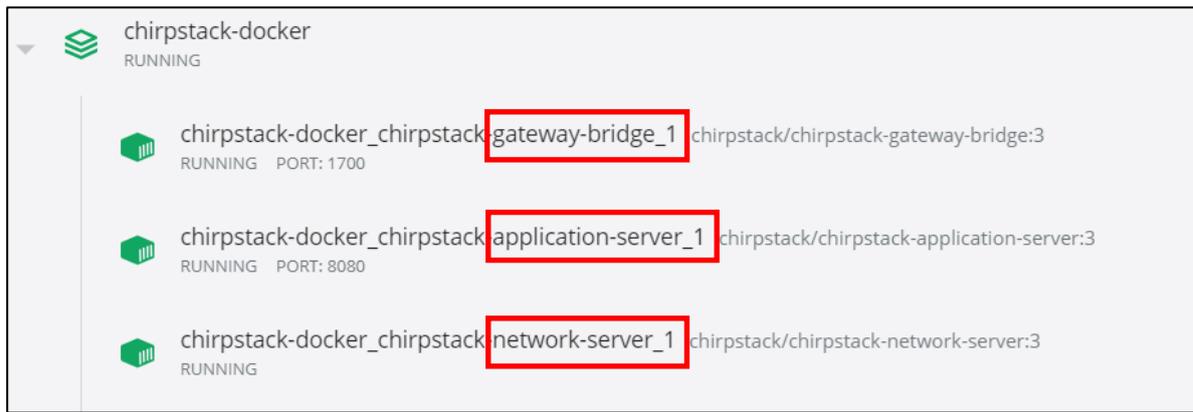


Figure 149: Conteneurs Docker de ChirpStack (Docker Windows Desktop)

La Figure 150 montre les services (conteneur Docker) générés avec Docker sur une distribution Linux.

NOMS PORTS

```
chirpstack_chirpstack-application-server_1 0.0.0.0:8080->8080/tcp
chirpstack_chirpstack-network-server_1
chirpstack_chirpstack-gateway-bridge_1 0.0.0.0:1700->1700/udp
```

Figure 150: Conteneurs Docker de ChirpStack (Docker sur une distribution Linux)

Nous pouvons remarquer que le service **Gateway Bridge** écoute sur le port 1700/udp parce que nous utilisons le "Semtech UDP Packet forwarder". C'est le port sur lequel notre Gateway doit envoyer ses données. L'Application Server écoute sur le port 8080/tcp. C'est l'accès à l'interface web pour configurer le serveur LoRaWAN.

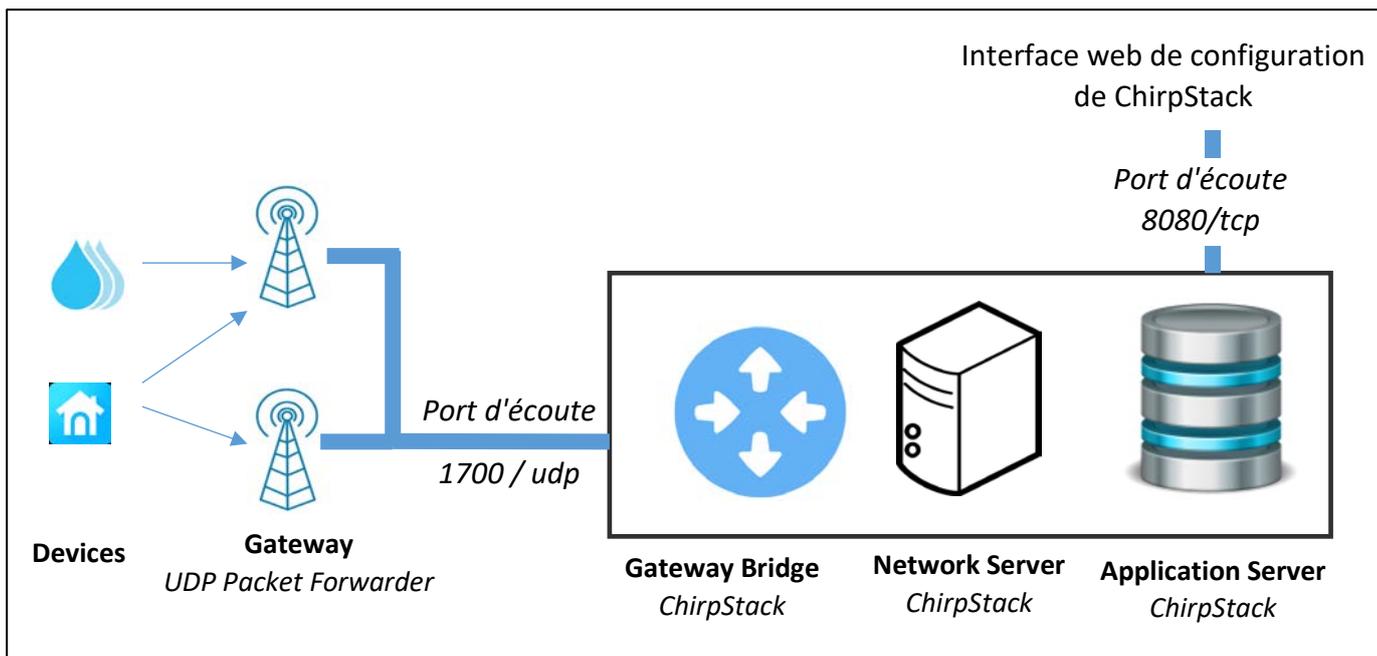


Figure 151: Architecture globale après l'installation de ChirpStack

9.2.3 Configuration de ChirpStack sur une Gateway Raspberry Pi

Une Raspberry Pi est souvent utilisée pour construire une Gateway et il est possible d'installer un serveur LoRaWAN intégré à celle-ci. La Gateway et le serveur LoRaWAN sont donc dans le même système. ChirpStack fournit une image appelée **chirpstack-gateway-os-full**. Cette image est prête à fonctionner et comprend :

- Un système d'exploitation embarqué basé sur Linux qui peut faire fonctionner diverses Gateways LoRaWAN (SEMTECH SX1301 CoreCell, IMST, RAK, RisingHF...).
- Une installation du Gateway Bridge, du Network Server et de l'Application Server.

Il s'agit d'une solution tout-en-un pour collecter et exporter les données IoT rapidement et facilement.

9.3 Configuration de ChirpStack

Nous configurons ChirpStack à partir d'une interface web graphique disponible via le port 8080/tcp : <http://@IP-ChirpStack:8080>

Les noms d'utilisateur et les mots de passe par défaut sont les suivants :

- Nom d'utilisateur : admin
- Mot de passe : admin



La démonstration de la configuration de ChirpStack est disponible ici en vidéo : www.univ-smb.fr/lorawan .

L'Application Server ChirpStack peut se connecter à une ou plusieurs instances du Network Server ChirpStack. Les administrateurs peuvent ajouter de nouveaux Network Servers à l'installation.

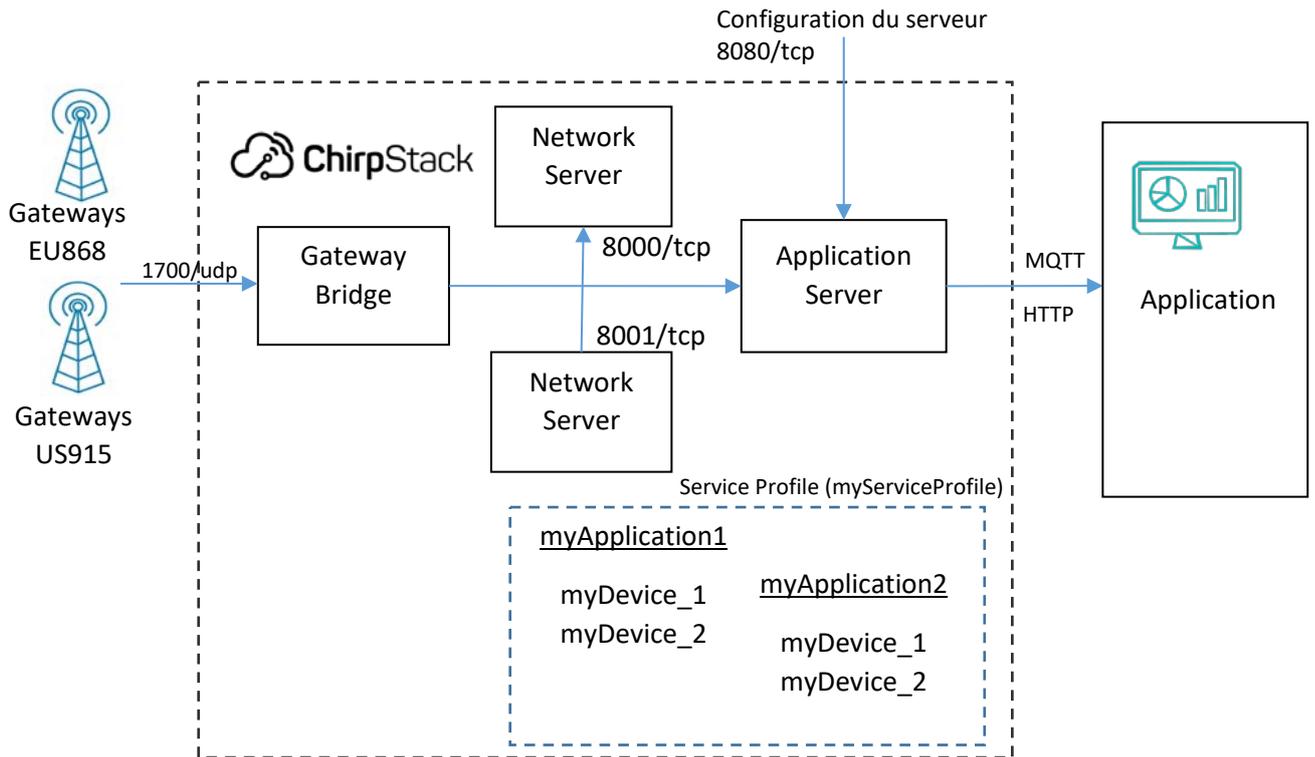


Figure 152: ChirpStack avec plusieurs Network Servers

Dans l'onglet **Application > monApplication > Devices > monDevice > FRAMES LORAWAN**, nous pouvons voir les trames LORAWAN.

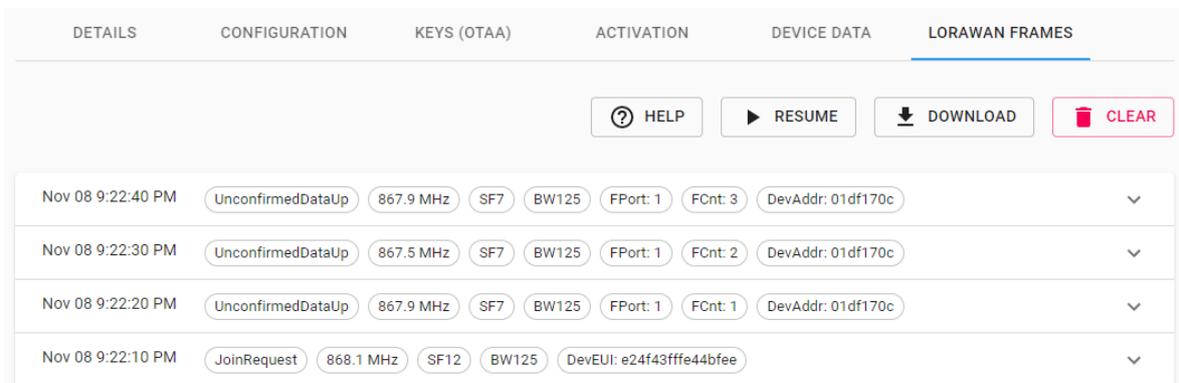


Figure 153: Trames LoRaWAN reçues dans ChirpStack

9.3.1 Exportation des données utilisateur

Vous pouvez trouver sur notre site web www.univ-smb.fr/lorawan :

- Le format de la requête HTTP POST du client pour le downlink.
- Le Topic MQTT à souscrire pour l'uplink.
- Le Topic MQTT à publier pour le downlink.



La démonstration de l'exportation des données utilisateur à l'aide de HTTP POST et MQTT en uplink et downlink est disponible ici en vidéo : www.univ-smb.fr/lorawan.

10 Installer votre propre application utilisateur- Plateforme IoT

10.1 Aperçu des choix

Dans le chapitre 7 nous avons expliqué comment exporter des données depuis notre serveur LoRaWAN (uplink) et comment fournir des données au serveur LoRaWAN (downlink). Nous allons maintenant examiner l'application utilisateur dans son ensemble et présenter plusieurs architectures fonctionnelles complètes. Notre application utilisateur doit importer, stocker, traiter et afficher ces données sur une page web. C'est ce que fait une plateforme IoT.

Le Tableau 29 liste les différents choix technologiques disponibles pour le prototypage de cette application utilisateur. Cette liste est loin d'être exhaustive, mais elle nous permet de comprendre les différentes fonctionnalités que nous devons construire.

Que voulons-nous ?	Comment pouvons-nous le faire ?	Quelques choix technologiques possibles			
<i>Une page Web disponible pour l'utilisateur</i>	<i>Un serveur Web Une interface utilisateur</i>				 
<i>Graphiques, tableaux, jauges, tableaux...</i>	<i>Librairies pour Dashboard</i>				  
<i>Sauvegarde des données</i>	<i>Une base de données</i>				  
<i>Importation de données</i>	<i>Un endpoint HTTP Subscriber et publisher MQTT</i>				  HTTP / MQTT

Tableau 30 : Choix technologiques pour construire une plateforme IoT

10.2 Construire une plateforme IoT

Un choix très courant pour construire une plateforme IoT est d'utiliser la stack open source **TICK** proposée par Influxdata [www.influxdata.com].

- **Telegraf** : Importation de données.
- **InfluxDB** : Stockage (sauvegarde).
- **Chronograf** : Formatage des données et affichage du tableau de bord.
- [**Kapacitor** : Nous n'utilisons pas ce service dans cette démonstration].

Un autre service, célèbre pour la création de tableaux de bord est Grafana. Il a le même objectif que Chronograf. Nous allons donc inclure Grafana dans notre test. Une fois encore, nous utilisons Docker et Docker Compose pour mettre en place les services de notre application utilisateur. Voici la liste des conteneurs :

- **Telegraf**.
- **InfluxDB** (connexion via le port 8680/tcp).
- **Grafana** (connexion via le port 3000/tcp).
- **Chronograf** (connexion via le port 8888/tcp).

Lorsque tous les conteneurs sont actifs, ils sont capables de communiquer entre eux en utilisant des adresses IP connues au sein de l'infrastructure Docker. Ces adresses IP peuvent être résolues par le nom du conteneur (influxdb, telegraf, ...).

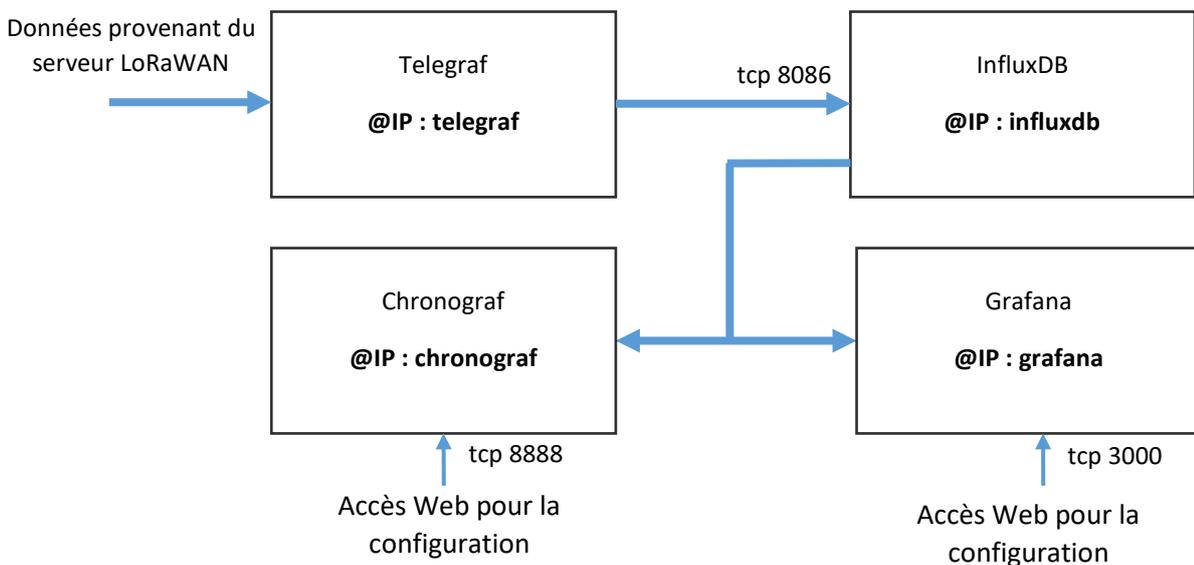


Figure 154: Conteneurs Docker pour notre application utilisateur

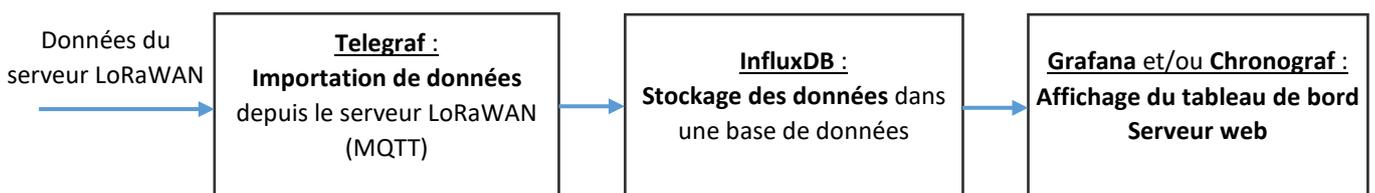


Figure 155: Telegraf - InfluxDB - Chronograf - Grafana



Toute cette mise en place est disponible sur Github au lien suivant :

<https://github.com/SylvainMontagny/dashboard-lorawan>



La démonstration de la mise en place de l'ensemble de cette plateforme IoT est disponible ici en vidéo : www.univ-smb.fr/lorawan .

11 LoRaWAN avancé

Nous avons maintenant une vision globale du protocole LoRaWAN. Cela devrait être suffisant pour une application simple. Cependant, certains sujets ont été volontairement occultés afin de rendre les choses plus faciles à comprendre. Ce chapitre a pour but d'entrer dans les détails et d'apprendre de nouvelles fonctionnalités.

11.1 Le Join Serveur

11.1.1 Le rôle du Join Server

Nous savons que le protocole LoRaWAN 1.0.x nécessite deux clés pour fonctionner :

- La NwSKey pour l'authentification.
- L'AppSKey pour le cryptage.

Le mode d'activation ABP fournit ces clés directement au Device. Cela simplifie le processus mais ce n'est pas la manière la plus sûre de le provisionner. Le mode d'activation recommandé est donc OTAA afin qu'un nouveau jeu de NwSKey et AppSKey soit généré à chaque nouvelle session grâce à la procédure de Join. Mais qui gère exactement cette procédure de Join?

Jusqu'à présent, nous considérons que cette phase d'activation était gérée par le Network Server. En réalité, c'est une entité spécifique appelée Join Server qui la gère. Le Join Server est connecté à la fois au Network Server et à l'Application Server et possède un identifiant unique de 8 octets (EUI) appelé JoinEUI. Le JoinEUI permet donc de définir le Join-Server qui sera utilisé pendant la phase d'activation.

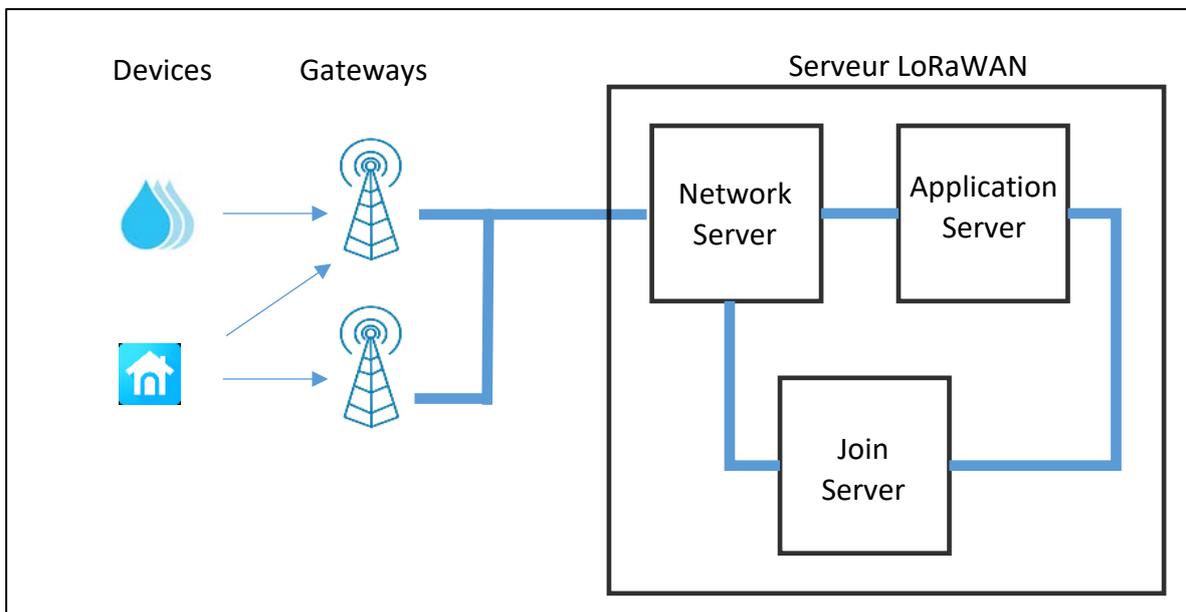


Figure 156: Le Join Server

Ces trois entités se trouvent souvent au même endroit, mais elles peuvent être sur des instances séparées et même appartenir à des fournisseurs de service différents. Par exemple, dans TTN, vous pouvez choisir pour chaque Device où se trouvent l'Application Server et le Join Server associé.

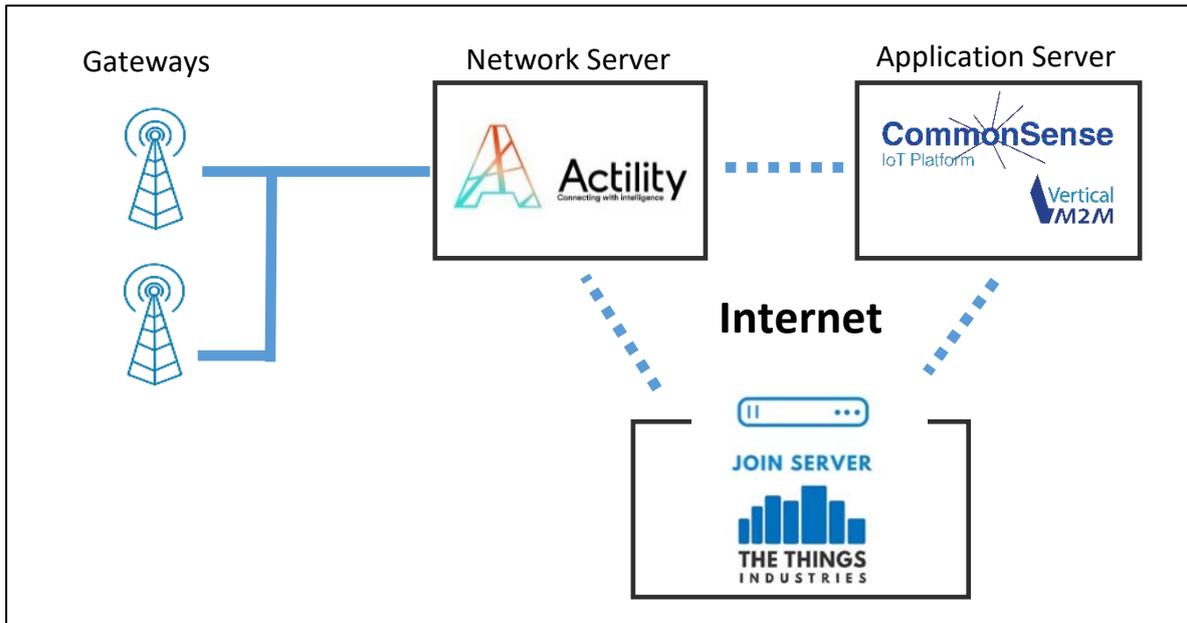


Figure 159: Network Server – Join Server - Application Server

11.1.2 La procédure d'activation en OTAA

Dans un chapitre précédent, la Figure 49 présentait la procédure de Join. On rappelle que la trame Join-Request est composée d'un JoinEUI, d'un DevEUI et d'un DevNonce comme le montre la Figure 160.

Size (octets)	8	8	2
Join-Request payload	JoinEUI	DevEUI	DevNonce

Figure 160: Trame Join-Request

Le DevNonce est une valeur importante pour le Join-Request car il protège des attaques par Replay, exactement comme nous l'avons vu pour le Frame Counter au chapitre 4.5.3. Un DevNonce ne peut pas être utilisé deux fois. Le nombre maximum de Join-Request est donc de 65536, ce qui ne devrait jamais arriver dans un fonctionnement normal car un Device LoRaWAN génère rarement des Join-Request. La spécification LoRaWAN propose une gestion différente du DevNonce en fonction de la version du protocole.

- **Jusqu'à LoRaWAN 1.0.3**, le Join Server accepte un Join-Request que si le DevNonce n'a jamais été utilisé. Jusqu'à LoRaWAN 1.0.3, le Device LoRaWAN choisissait aléatoirement un DevNonce parmi les 65536 disponibles. Si le Device n'a pas sauvegardé les valeurs utilisées précédemment pour le DevNonce, ce n'est pas très grave car lors de la prochaine Join-Request, il générera une nouvelle valeur aléatoire qui aura très probablement jamais été utilisée.
- **A partir de LoRaWAN 1.0.4**, le Join Server accepte un Join-Request uniquement si la valeur utilisée est supérieure à la valeur précédente. Depuis LoRaWAN 1.0.4, le Device utilise un compteur pour le DevNonce. Si le Device redémarre, il doit conserver la dernière valeur du DevNonce dans une mémoire non volatile. Sinon, la demande d'activation sera rejetée.

Lorsque le Join-Request est accepté, le Join Server renvoie un Join-Accept avec les informations indiquées sur la Figure 161. Parmi celles-ci figurent le DevAddr, les informations de configuration du réseau, ainsi que tout ce dont le Device a besoin pour générer la NwkSKey et l'AppSKey de son côté.

Size (octets)	3	3	4	1	1	(16) optional
Join-Accept payload	JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

Figure 161: Format d'une trame Join-Accept

Le but du Join Server est de :

- Stocker en toute sécurité les clés racine (AppKey).
- Répondre à la demande d'activation (Join-Request) émise par les Devices autorisés.
- Stocker et transmettre en toute sécurité la NwSKey au Network Server.
- Stocker et transmettre en toute sécurité l'AppSKey à l'Application Server.

Le Join Server contient les informations suivantes pour chaque Device sous son contrôle :

- Le DevEUI
- L'AppKey
- L'identifiant du réseau : NetID
- L'identifiant de l'Application Server
- La version LoRaWAN du Device

On peut se demander quel est l'intérêt d'une telle structure puisque la gestion des clés est simplement déportée sur un autre serveur. Pour répondre à cette question, il faut comprendre que le Join Server n'est pas censé être un service interne connecté à un seul Network Server et à un seul Application Server. La Figure 162 représente les multiples connexions possibles entre eux.

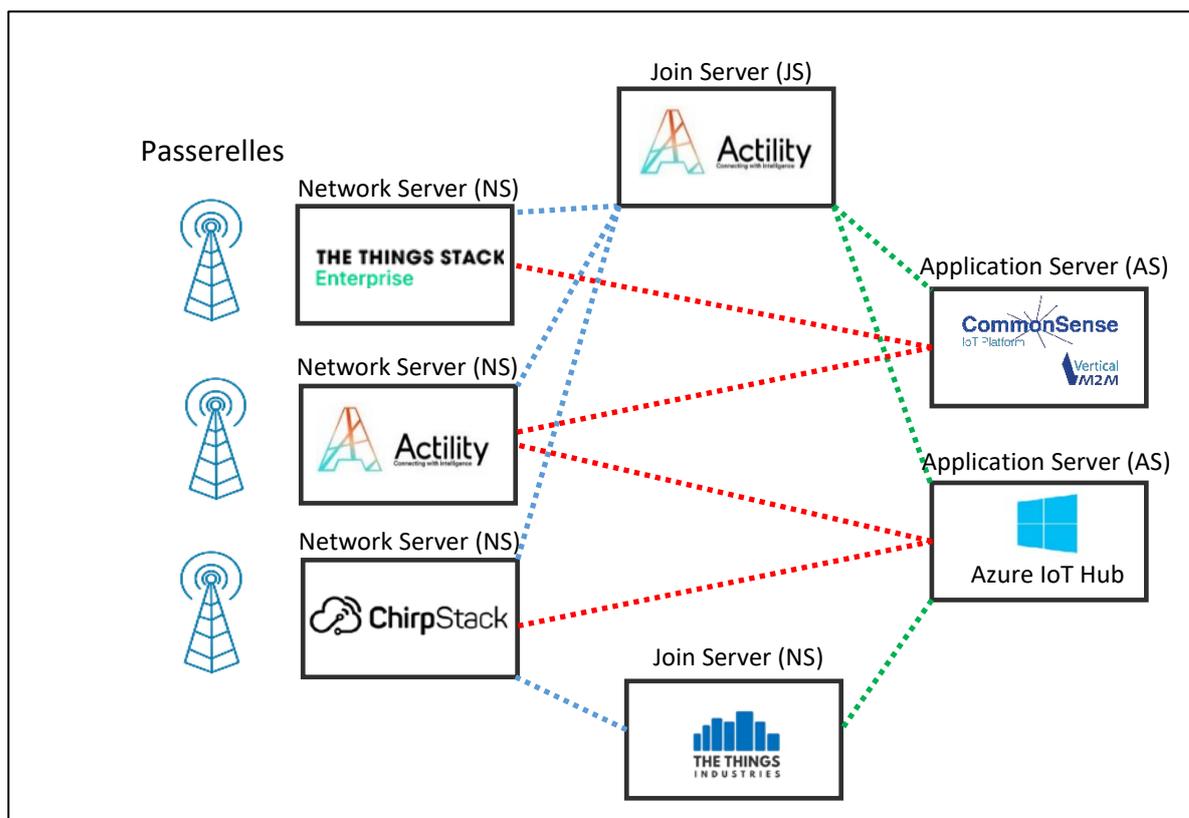


Figure 162: Connexions des Network Servers, des Join Servers et des Application Servers

- Un Network Server (NS) peut être connecté à plusieurs Join Server (JS). Par exemple, ChirpStack peut se connecter au Join Server d'Actility et au Join Server de TTI. Le Network Server d'Actility sait quel JS il doit solliciter grâce au JoinEUI envoyé par le Device pendant la procédure de Join. Un NS peut également se connecter à plusieurs AS.
- Un JS peut connecter plusieurs NS et donc opérer sur plusieurs réseaux. Il peut aussi travailler avec plusieurs AS pour l'échange de l'AppSKey.

Une telle structure présente de nombreux avantages du point de vue de la sécurité et de l'interopérabilité des Devices au sein de différents réseaux.

11.1.3 Keys provisioning

Lorsque le processus d'activation OTAA s'exécute, nous supposons que les clés racine sont déjà stockées dans le Device d'une part et dans le serveur d'autre part. Il faut donc qu'auparavant, le fabricant du Device et le Join Server aient échangé ces informations avant la vente de l'objet.

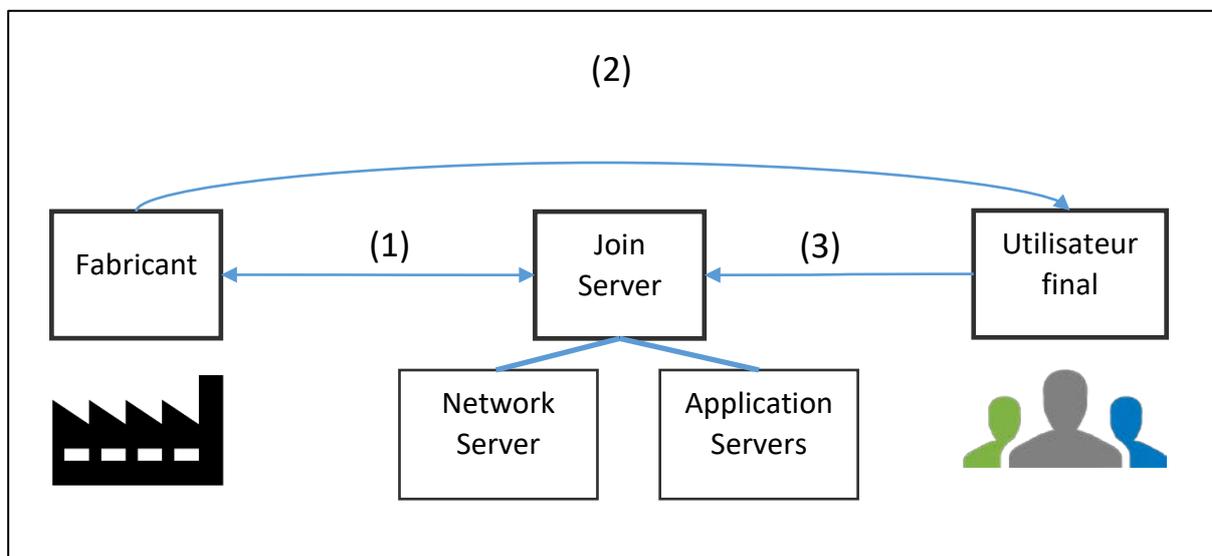


Figure 163: Le fabricant du Device, l'utilisateur final et le Join Server

(1) Au cours de l'industrialisation, le fabricant enregistrera dans le Device les clés (AppKey dans LoRaWAN 1.0.x) ainsi que les JoinEUI et DevEUI. Le DevEUI et l'AppKey doivent également être stockés sur le Join Server. Lorsqu'ils partagent des clés, ils doivent convenir d'un moyen sûr de garantir qu'elles ne soient pas exposées. Ces clés doivent ensuite être stockées de manière très sécurisée sur les deux entités. C'est ce que l'on appelle la phase de **provisioning**.

(2) Ensuite, le Device est vendu à l'utilisateur final.

(3) Enfin, l'utilisateur final doit prouver sa propriété du Device au Join Server. Cette action permet d'autoriser le Join Server à créer les clés de session (NwSKey et AppSKey) lors de la prochaine réception d'un Join-Request.

11.1.4 Sécuriser les clés

La sécurité du protocole LoRaWAN dépend de la manière dont les clés racine sont partagées et stockées (AppKey dans LoRaWAN 1.0.x).

Les défis sont les suivants :

- **Comment provisionner la même clé racine (AppKey) dans le Device et dans le Join Server sans les exposer ?** Cette opération est compliquée car les entreprises qui fabriquent le

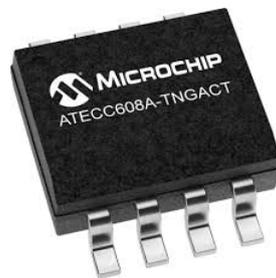
Device LoRaWAN ne sont pas forcément des entreprises de confiance, vous ne pouvez donc pas forcément leur fournir la liste des clés à intégrer dans vos composants sans précaution.

- **Comment réagir en cas d'intrusion ?** L'accès physique aux mémoires stockant les clés doit être le plus limité possible. Une détection d'intrusion doit conduire à l'autodestruction des clés.
- **Comment éviter de donner des indices pour trouver les clés ?** Les calculs effectués pendant le chiffrement ne doivent laisser aucun indice, comme des variations de tension ou des consommations de courant qui pourraient aider à retrouver les valeurs utilisées.

Pour sécuriser le stockage de ces clés et effectuer toutes les opérations de chiffrement, des modules matériels spécifiques sont donc nécessaires des deux côtés (Device et Join Server).

- Sur le Device, nous utilisons des **SE** (Secure Element). Il s'agit de très petits composants proches du microcontrôleur.

Par exemple, le composant ATECC608B-TNGACT possède toutes ces caractéristiques et est déjà provisionné, c'est-à-dire qu'il contient déjà les clés racine pour le serveur LoRaWAN ACTILITY.



- Du côté du Join Server, nous utilisons un **HSM** (Hardware Security Module). Il est connectés au réseau et possède les mêmes caractéristiques de sécurité que coté Device, mais bien sûr avec beaucoup plus de puissance et de fonctionnalités.

11.1.5 Preuve de la propriété d'un Device : "onboarding"

Lorsque le Device est fabriqué, il est approvisionné avec ses clés. Ces mêmes clés sont stockées sur un Join Server, mais il n'est actuellement pas autorisé à répondre à une procédure de Join car aucun utilisateur final n'a prouvé sa propriété. La preuve de propriété peut être faite par un QR code fourni par le vendeur ou par les informations sur le Device (DevEUI) associées à un jeton (token) (voir Figure 164).

Le processus de validation d'un propriétaire, ainsi que l'autorisation d'utiliser un réseau est appelé "onboarding". Ces appareils peuvent être retirés (offboarding) lorsqu'un utilisateur ne souhaite plus être responsable du Device, lors d'une vente par exemple.

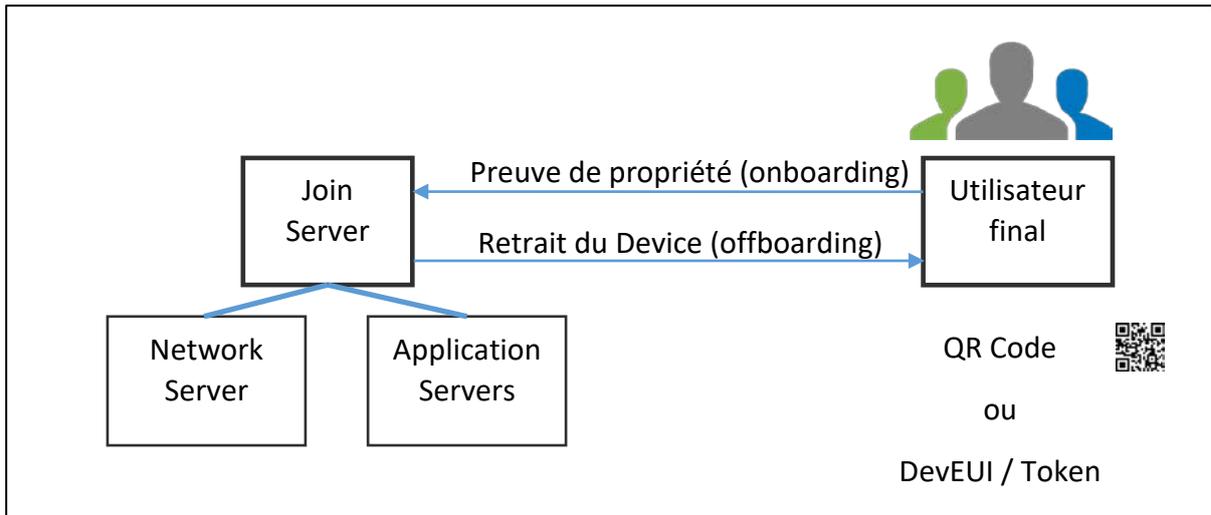


Figure 164: Réclamation du Device

Si la demande est réussie, le serveur Join autorise le Join-Request pour le Device correspondant. Les clés de session seront ensuite distribuées au Network Server (NwkSKey) et à l'Application Server (AppSKey).

Si une nouvelle demande est effectuée, par exemple par un nouveau propriétaire, le Device devra refaire la procédure de Join. En effet, la réclamation d'un Device ne fait que transférer la propriété de celui-ci mais ne transfère pas les clés de session.

Versions du document

Version 1.0 : Janvier 2022

- Version initiale.

Version 1.1 : Mars 2022

- Nombreux changements et corrections techniques mineures.
- Relecture complète et correction orthographique.
- Correction paragraphe sur "les plans de fréquences".
- Ajout du plan de fréquence pour le Roaming (Guidelines de la LoRa Alliance)