Réseau LoRa

Table des matières

INTRODUCTION :

Le réseau LoRa (Long Range) est utilisé pour recevoir et envoyés des données aux Objets Connecté (IoT).

Ce type de réseau se nomme **LPWAN = Low Power Wide Area Network**

L'architecture du réseau LoRa repose sur 3 éléments clés :

- **Objets connectés (capteurs)** : Qui mesurent et collectent des données (Température, humidité etc...)

- **Passerelles (Gateway)** : Elles reçoivent les données transmises et les acheminent vers les serveurs centraux

- Serveurs de réseau (Network Server) : Ils gèrent le traitement, le stockage et l'analyse des données reçues, permettant leur exploitation.

La communication entre les capteurs des IoT et les passerelles utilise la modulation LoRa, qui fonctionne sur des bandes de fréquences libres, notamment la bande des 868 MHz en Europe.

A savoir que cette bande n'est utilisable que a 1% du temps on appelle cela le **duty cycle**.

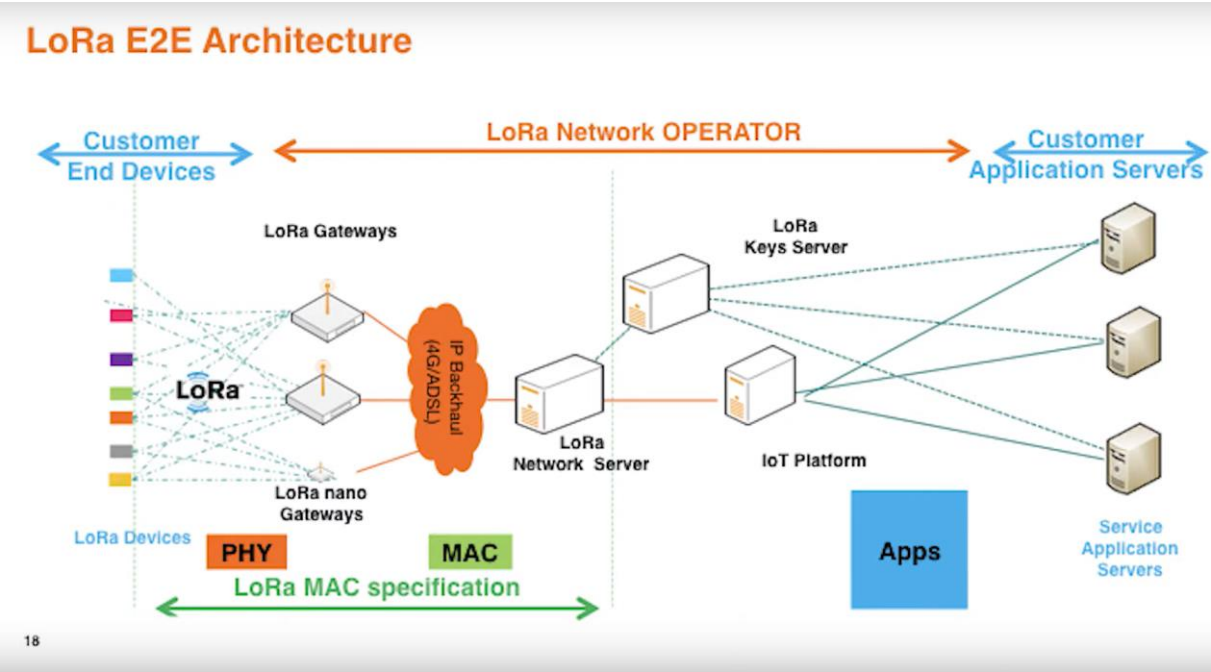
Duty cycle ce qui correspond au fait de ne pas « parler » plus de 1% du temps ce qui correspond un peu près à 36 seconde par heure et pour nos données cela correspond à 1 message (Envoie de donnée) **une fois toutes les 10 mins** si il y a besoin de plus cela vas être compliqué d'utilisé le réseau LoRa(Il faut utilise lte-m).

Il faut bien distinguer :

LoRa : Désigne la couche physique, c'est-à-dire la technologie de modulation radio qui permet la transmission des données.

LoRaWAN : Représente le protocole de communication réseau qui définit la manière dont les objets connectés communiquent via le réseau LoRa

Schéma Architecture LoRa :



Tout cet ensemble représente le protocole LoRaWAN qui est au-dessus de la modulation LoRa qui elle est entre l'appareil connecté et la passerelle.

Les gateway LoRa écoutent tous les canaux et Spreading factor en même temps qui sont renvoyé vers un serveur appelé LoRa Network Server, ce serveur s'occupe de l'authenticité, la confidentialité et l'intégrité des communications.

Authenticité :

- Le LNS gère l'authentification des objets connectés en vérifiant leur identité avant d'autoriser leur accès au réseau.
- L'activation des dispositifs peut se faire via **OTAA (Over-The-Air Activation)** ou **ABP (Activation By Personalization)**.

Confidentialité :

- Le protocole LoRaWAN utilise un **chiffrement AES-128** pour sécuriser les communications entre les objets connectés et le serveur d'application.

- Chaque message est protégé pour empêcher toute écoute indésirable.

Intégrité :

- Chaque paquet de données comprend un **Message Integrity Code (MIC)**, permettant de vérifier qu'il n'a pas été altéré pendant son transit.
- Le LNS vérifie cette intégrité avant de transmettre les données aux applications backend.

Cette architecture peut être réalisé en local au lieu de faire appel à un opérateur (Public).

On peut aussi faire un réseau hybride ou on fournit La Gateway LoRa et la partie serveur est laissé à l'opérateur.

Donc Trois réseaux possibles :
Public
Privé
Hybride

Les trois réseaux auront toujours la même architecture.

Modulation LoRa :

Modulation Lora = **4 paramètres** pour le receveur et le transmetteur, Donc ils doivent avoir le même **Channel**, la même **bande passante**, le même **coding rate** (Plus il est élevé plus il permet de détecter et même corriger des erreurs dans la transmission sans avoir à retransmettre une trame ou cela s'est mal passé). Et le **spreading factor**.

Channel : 868.1 MHz

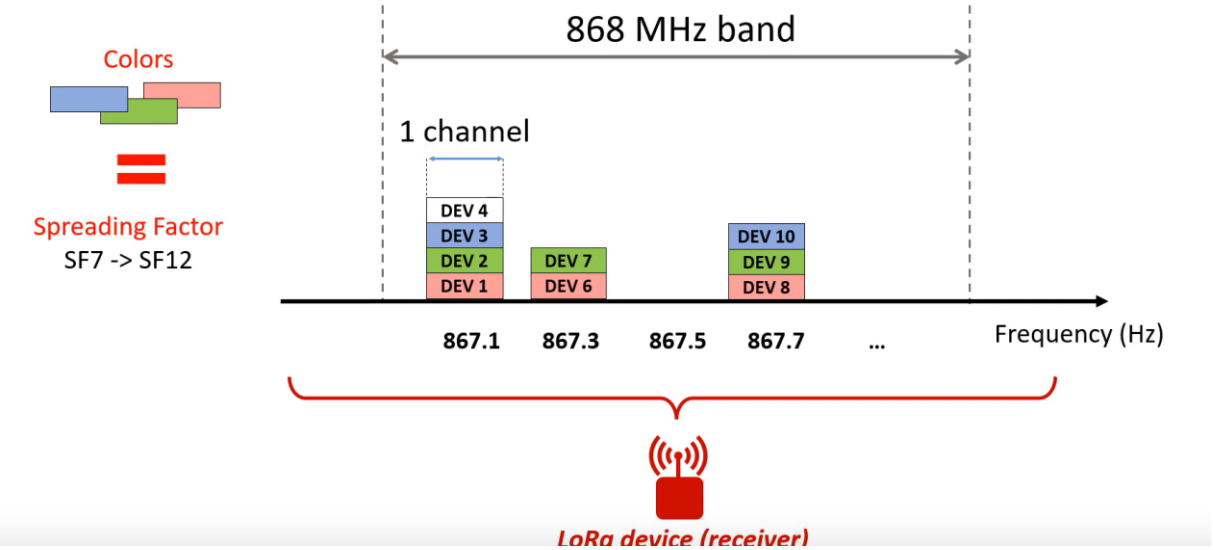
Bandwidth : 125 kHz

Spreading Factor : SF10

Coding Rate : 4/5

Sync Word : 0x34

Dans un même canal on peut transmettre des données de différents appareil en utilisant un spreading factor différents pour chaque appareil
Spreading Factor SF7 à SF12



Il permet aussi de déterminer la puissance du signal.

La gestion du Spreading factor est assuré de manière automatique par le réseau.

Cette manière de de déterminer automatiquement les 4 paramètres de la modulation LoRa est appeler ADR.

La bonne pratique lors de l'attache initial (un join request) est de le configurer en spreading factor 12 (message très lent et puissance maximum) pour être sûr d'être entendu par le réseau, le réseau vas répondre et lui dire de passer en spreading factor 9 ce qui vas permettre d'établir une communication optimisée (Moins de consommation de batterie et de latence).

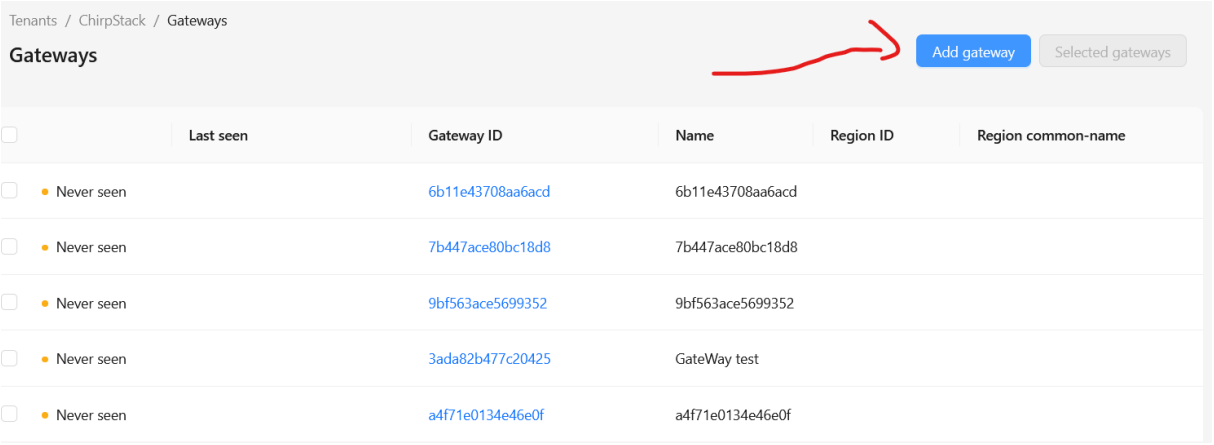
A ne pas faire pour des objets qui se déplace car l'appareil change de zone de réseau, les conditions de réception ne sont donc jamais les mêmes et il y a un risque que des réseaux ne l'entendent pas. Il faudrait donc pour un

objet en mouvement utiliser un spreading factor fixe à définir en fonction de la chance d'être entendu et la consommation énergétique (Souvent le spreading factor adapté à ce genre de situation est le 10).

Enregistrer un appareil connecté :

Il faut avant tout enregistrer la **passerelle** dans le serveur réseau pour pouvoir recevoir les demandes d'upload des appareils connectés.

Sur Chirpstack il faut par exemple rajouter un serveur



Il suffit de préciser un nom et surtout l'ID de la Gateway qui correspond a l'adresse mac mais sous un format **EUI-64 donc par exemple MAC AA:BB:CC:DD:EE:FF, son Gateway ID sera AABBCCDDEEFF.**

Pour enregistrer un **appareil connecté (end-device)** dans le réseau LoRa et pouvoir l'authentifié :

- Il faudra pour l'appareil connecté et l'application Server : **Une Application Session Key (AppSKey**) pour la **confidentialité** des échanges.

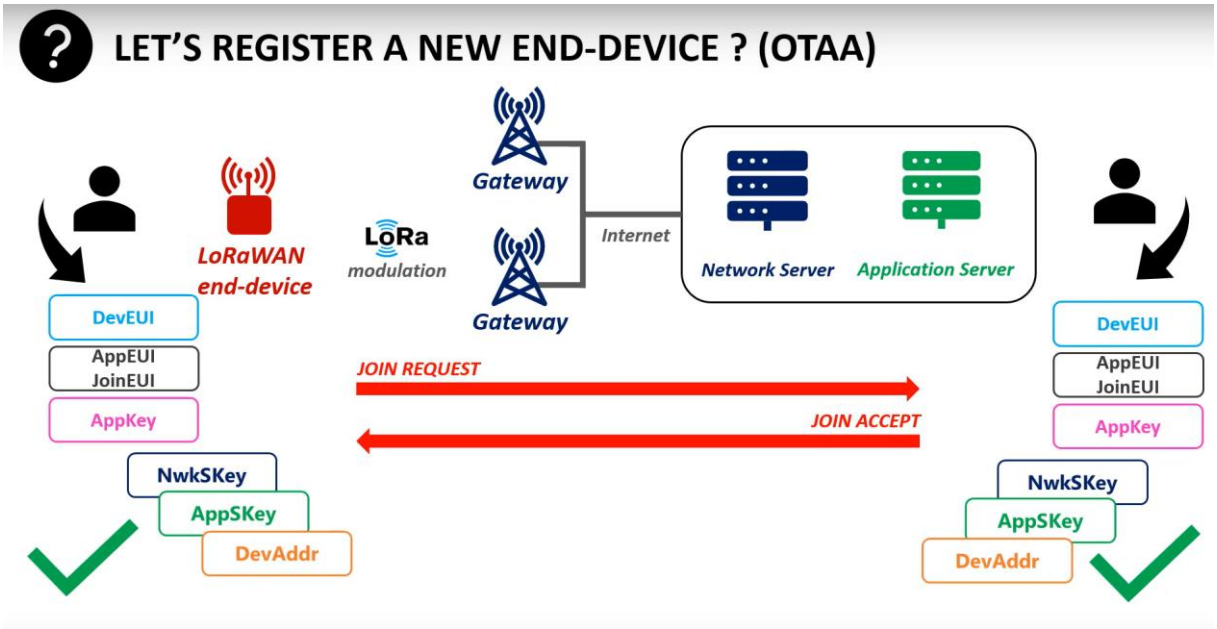
- Ensuite pour gérer l'**authentification** il faudra une clé commune dans le **Network server et l'appareil connecté** qui s'appelle une **Network Session Key (NwkSKey).**

- Et pour **Identifié L'appareil** il faut que l'appareil connecté est une **Device Address (DevAddr)** et le **Network Server** doit avoir la DevAddr de l'appareil.

Tout ceci se nomme **ABP (Activation By Personalization) Cette manière de faire est obsolète.**

Inconvénient : Le end device (IoT) n'as aucun information sur les canaux qu'il a sa disposition , il ne connait pas le délais ou encore le spreading factor a utilisé. Ce qui peut poser un problème pour certaine fonctionnalité. (PAS ADR)

Il y a donc une autre solution **OTAA** :



Le principe de **AppSKey, NwkSkey et DevAddr** est le même mais ils sont ici générés automatiquement par le LNS (Server LoRaWAN/Serveur Réseau).

Il faut rentrer trois autres informations dans l'end-device et le Network Server pour rendre ceci possible. **DevEUI, AppEUI/JoinEUI et AppKey**.

DevEUI (Device EUI)

📌 **Qu'est-ce que c'est ?**

- **Identifiant unique mondial** de l'appareil.
- Attribué par le fabricant (comme une adresse MAC pour une carte réseau mais dans **le format EUI-64**).
- **Utilisé pour identifier de manière unique l'appareil sur n'importe quel réseau LoRaWAN**.

AppEUI / JoinEUI

📌 **Qu'est-ce que c'est ?**

- **Identifiant de l'Application Server auquel l'appareil doit envoyer ses données**.

- L'AppEUI est utilisé pour savoir **quel service ou réseau doit recevoir les données de l'appareil**.
- Dans **LoRaWAN 1.0.4**, l'AppEUI est remplacé par **JoinEUI** (même fonction, juste un changement de nom).

AppKey (Application Key)

📌 **Qu'est-ce que c'est ?**

- Une **clé secrète de 128 bits** utilisée lors de l'activation OTAA (Over-The-Air Activation).
- Sert à **générer les clés de session** :
  - **AppSKey (Application Session Key)** → Chiffre les données entre l'appareil et l'Application Server.
  - **NwkSKey (Network Session Key)** → Sécurise les échanges entre l'appareil et le Network Server.

Paramètre	Rôle	Stocké dans l'appareil ?	Stocké dans le Network Server ?	Stocké dans l'Application Server ?
DevEUI	Identifiant unique de l'appareil	✅ Oui	✅ Oui	❌ Non
AppEUI / JoinEUI	Identifiant de l'application cible	✅ Oui	✅ Oui	❌ Non
AppKey	Clé secrète pour générer les clés de session	✅ Oui	❌ Non	✅ Oui

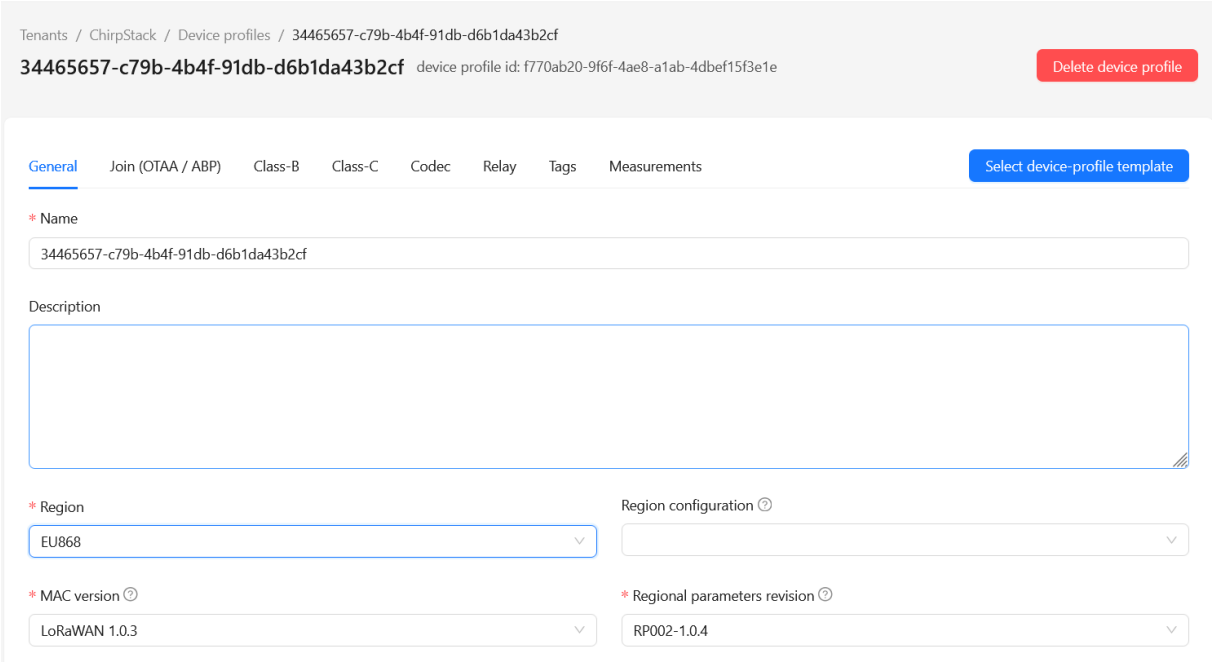
📌 **Exemple de scénario avec OTAA**

1) L'appareil LoRaWAN **(DevEUI = A1B2C3D4E5F6G7H8)** envoie une **Join Request** au Network Server.
2) Le Network Server vérifie si le **DevEUI et le JoinEUI sont valides**.
3) L'Application Server utilisent **l'AppKey pour générer AppSKey et NwkSKey**.
4) Le Network Server répond avec un **Join Accept**, contenant une **DevAddr, AppSKey et NwkSKey**.
5) L'appareil commence à envoyer des données, **chiffrées avec l'AppSKey**,

et le Network Server **vérifie leur intégrité avec la NwkSKey**.

Le Join Accept contient aussi l'ensemble des informations qui vont permettre au devices de récupéré les paramètres LoRaWAN, donc toutes les fréquences utilisées par l'appareil connecté pour transmettre à la passerelle, le délai de réception ou encore le Spreading Factor (ADR).

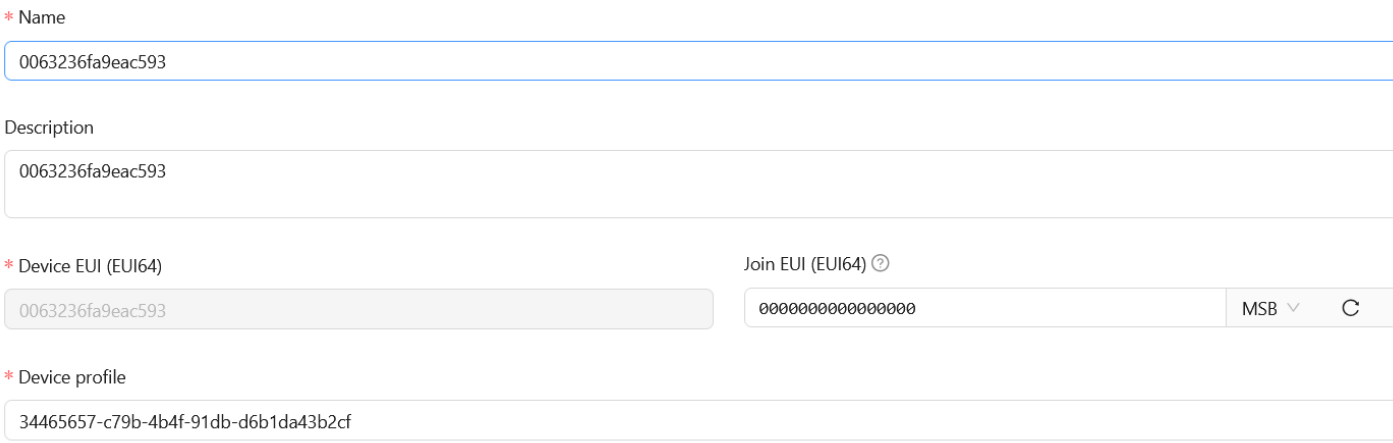
Sur Chirpstack il faut d'abord créer une Template précisant les spécifications des appareils qui vont se connecter :



On précise la **région et la fréquence utilisé EU868**, **la version de LoRaWAN** supporter par le ou les appareils et ensuite la **version des paramètres régionaux LoRaWAN** que ton appareil utilise.

On précise si on utilise **OTAA** ou non (Si en prod obligatoire de l'utilisé pour la sécurité) et on peux préciser **la classe des appareils** (Expliquer plus bas dans le document).

Une fois notre modèle configurer, on peux passer a l'enregistrement individuel des appareils avec **OTAA** :

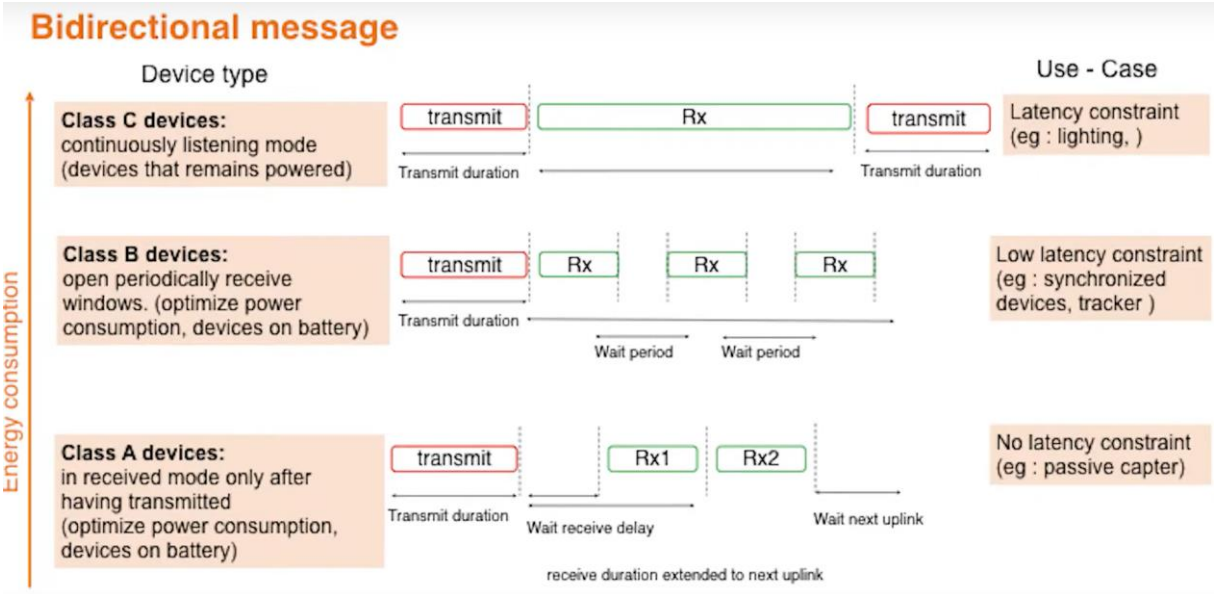


Il y a deux paramètres a spécifié le **DevEUI** donc l'adresse MAC de l'appareil connecté au format **EUI64** et le **template/modèle** définit plus tôt.

Une fois ceci réalisé, il suffit d'attendre que la **passerelle** reçoive les données de l'appareil. Elle les redirigera ensuite **vers le serveur réseau**, qui vérifiera si elles proviennent d'un **appareil enregistré**. Si c'est le cas, il les **stockera** ou les **transmettra** à la partie applicative.

Tout ceci peut se faire automatiquement grâce a L'API REST de Chirpstack.

Classe des appareils connectés :



Class A :

L'appareil émet (UPLINK) et ensuite une fenêtre de réception s'ouvre pendant 2 secondes 1 slot (RX1) 1 seconde et 2 secondes (RX2), passé ce délai il faudra attendre la prochaine transmission (UPLINK) de l'appareil pour pouvoir envoyer des données à l'appareil (DOWNLINK). Très peu de consommation d'énergie.

Class B :

On peut rajouter autant d'emplacement que l'on veut pour la réception (Downlink) et le rythme de slot entre deux UPLINK. Consomme un peu plus d'énergie

Class C :

Entre deux UPlink la réception est tout le temps ouverte (DOWNlink) donc disponible tout le temps. En permanence branché car consomme beaucoup d'énergie.

LoRaWAN FUOTA permettant de mettre à jour des appareil connectés (utilise la fragmentation des données)

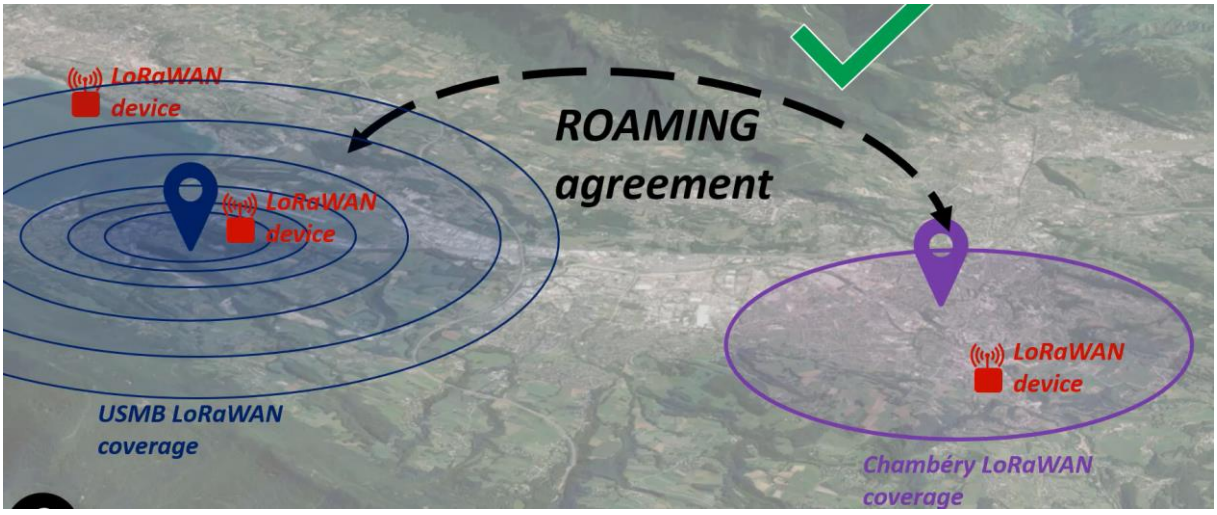
Le roaming :

Le **roaming**, ou **itinérance** en français, est une fonctionnalité des réseaux de téléphonie mobile qui permet à un abonné d'utiliser son téléphone pour passer des appels, envoyer des SMS ou accéder à Internet lorsqu'il se trouve en dehors de la zone de couverture de son opérateur habituel. Cette capacité est rendue possible grâce à des accords entre différents opérateurs, permettant à un utilisateur de se connecter au réseau d'un autre opérateur lorsque le sien n'est pas disponible.

La LoRa Alliance a défini plusieurs méthodes de roaming :

- Hand-over roaming : comme pour les smartphones, l'objet est connecté au réseau de l'opérateur visité, mais reste géré par l'opérateur d'origine. Ce type de roaming n'est pas compatible avec la v.1.0 de LoRaWAN qu'utilise la quasi-totalité des objets déployés

- Passive Roaming : l'objet reste connecté à son opérateur d'origine, vers lequel un ou plusieurs réseaux du pays visité transfèrent les données. La transmission peut être directe et sans contrôle (stateless) ou selon autorisation (stateful). C'est le mode stateless qui a été utilisé pour les tests de roaming entre KPN et Orange par exemple.



Géolocalisation LoRa :

Qu'est-ce que la Macro-Géolocalisation LoRaWAN® ?

C'est une technologie qui vous permet d'estimer la position de vos appareils IoT sans GPS (Donc moins de consommation), en analysant la puissance des signaux reçus par

les antennes du réseau LoRaWAN Orange.

Faille de sécurité :

Une des principales failles de sécurité est **l'attaque du serveur de réseau**, permettant donc de voir et manipuler le réseau. Ce serveur doit être très sécuriser car c'est un point d'entré très important dans le réseau

Une autre faille serait **d'envoyer énormément de requête(DDOS)** sur l'appareil connecté (IoT), ce qui résulterait a l'épuisement de la batterie plus vite que l'anormal.

Et pour finir une autre faille pourrait être le **brouillage des communications** radios ce qui bloquerait toute communication de l'appareil connecté vers la passerelle LoRa.

Toutes ces failles sont présentes lorsque l'on utilise **L'OTAA** donc génération de clé de façon dynamique. Si on utilise le **ABP** encore plus de faille de sécurité apparaissent :

- **Interception et décryptage des messages (Plaintext Recovery)** donc on récupère assez de trame pour effectuer une **analyse cryptographique** et récupérer l'AppKey.

- Si la clé a été intercepté il peut faire de **l'injection et modification de messages (Message Forgery)** et donc envoyé des fausses données au serveur.

- **Usurpation d'identité (Device Cloning)**

Si un attaquant récupère les clés de session (NwkSKey et AppSKey), il peut **répliquer l'identité d'un capteur** et envoyer de fausses données à la place du capteur légitime.

Il peut aussi **prendre le contrôle du capteur** et envoyer des commandes malveillantes sur le serveur réseau.

Application Concrète :

Agriculture intelligente 🌱

- **Surveillance des cultures** : Les capteurs LoRa sont déployés dans les champs pour mesurer des variables comme l'humidité du sol, la température, la luminosité et la qualité de l'air. Ces données permettent aux agriculteurs de prendre des décisions basées sur des informations précises, réduisant ainsi la consommation d'eau et optimisant la récolte.
- **Gestion des ressources** : Les capteurs LoRa peuvent être utilisés pour surveiller le niveau d'eau dans les réservoirs et contrôler l'irrigation, assurant une utilisation efficace des ressources.

2. Gestion de la ville intelligente 🏙️

- **Éclairage public intelligent** : Les lampadaires sont équipés de capteurs LoRa pour ajuster l'intensité lumineuse en fonction de la luminosité ambiante ou de la présence de personnes, réduisant ainsi la consommation d'énergie.
- **Gestion des déchets** : Les capteurs LoRa installés dans les poubelles de la ville peuvent surveiller leur niveau de remplissage, permettant aux services de gestion des déchets de planifier plus efficacement les tournées de collecte.
- **Suivi de la qualité de l'air** : Des capteurs LoRa peuvent être utilisés pour mesurer les niveaux de pollution de l'air et fournir des données en temps réel aux autorités publiques, contribuant à des actions ciblées contre la pollution.

3. Santé connectée 🩺

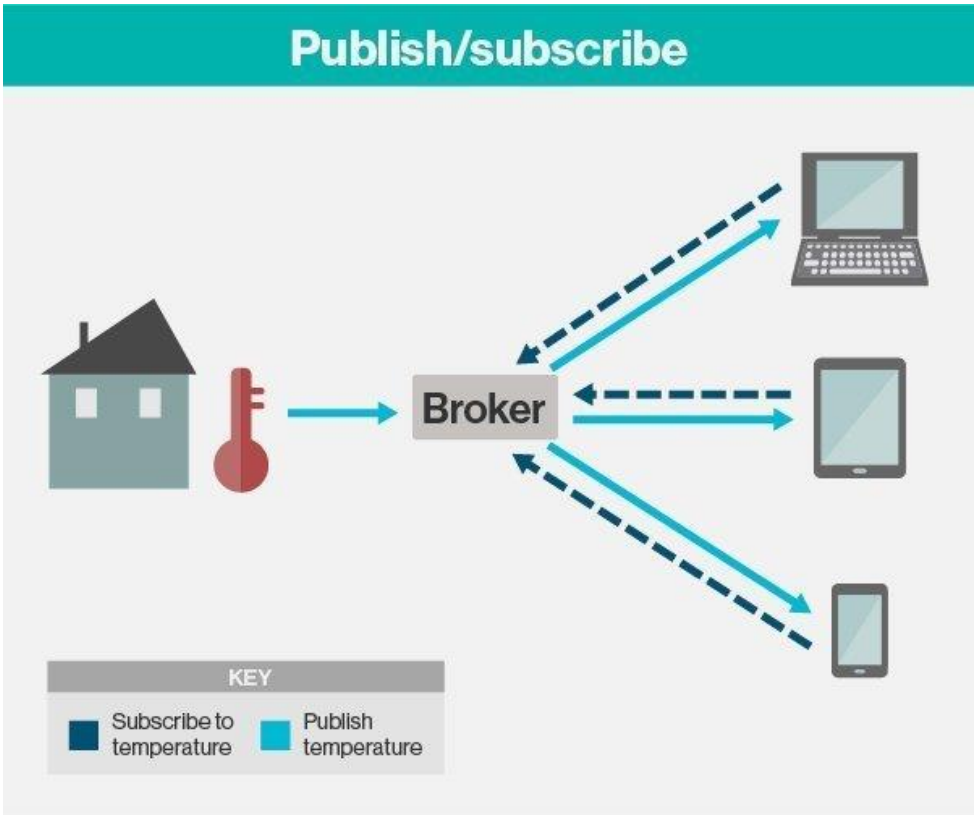
- **Surveillance des patients à distance** : Les dispositifs médicaux portables utilisant LoRa peuvent surveiller des paramètres vitaux tels que la fréquence cardiaque, la pression artérielle ou la température corporelle. Ces données sont envoyées à une station centrale pour un suivi continu.
- **Suivi des médicaments** : Les capteurs LoRa peuvent suivre la prise des médicaments par les patients, en envoyant des alertes si une dose est manquée.

4. Gestion de la chaîne d'approvisionnement 🚚

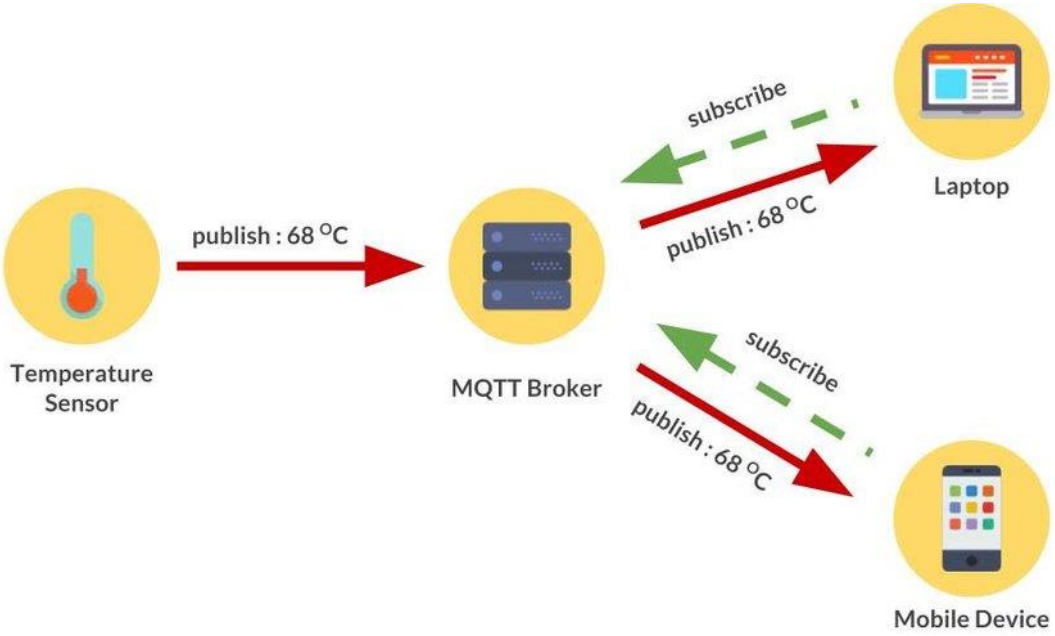
- **Suivi des marchandises** : LoRa permet de suivre en temps réel l’emplacement et l'état des marchandises (température, humidité) pendant le transport, notamment pour les produits périssables.
- **Gestion des stocks** : Des capteurs peuvent surveiller le niveau des stocks dans un entrepôt, réduisant le besoin d'interventions humaines et garantissant une gestion optimisée des inventaires.

MQTT

Schéma d’utilisation d’un serveur MQTT (Mosquito par exemple)



Une fois les données récupérées grâce au processus LoRa décrit ci-dessus, il est nécessaire de les transmettre vers la couche applicative, comme une base de données, une plateforme IoT ou un autre système. Pour cela, on utilise un serveur MQTT, un protocole de messagerie léger conçu pour les communications M2M (Machine-to-Machine) et IoT.



MQTT fonctionne selon un modèle **publish/subscribe** : les capteurs et passerelles **publient des messages sur des topics** (sujets), et les applications qui ont besoin des données **s'abonnent à ces topics** pour les recevoir en temps réel. Les données, souvent au format **JSON**, peuvent ensuite être transformées et stockées dans une base de données ou envoyées vers d'autres services applicatifs.

Ce protocole est particulièrement adapté aux environnements IoT grâce à sa faible consommation de bande passante et sa gestion efficace des connexions instables.