



Projet IOT

Rédacteur : MARQUES Acacio

Version : VP2024 - 08



P2

Introduction

Organisation - définition - demain

P12

LORA

Etalement de spectre – modulation LoRa

P24

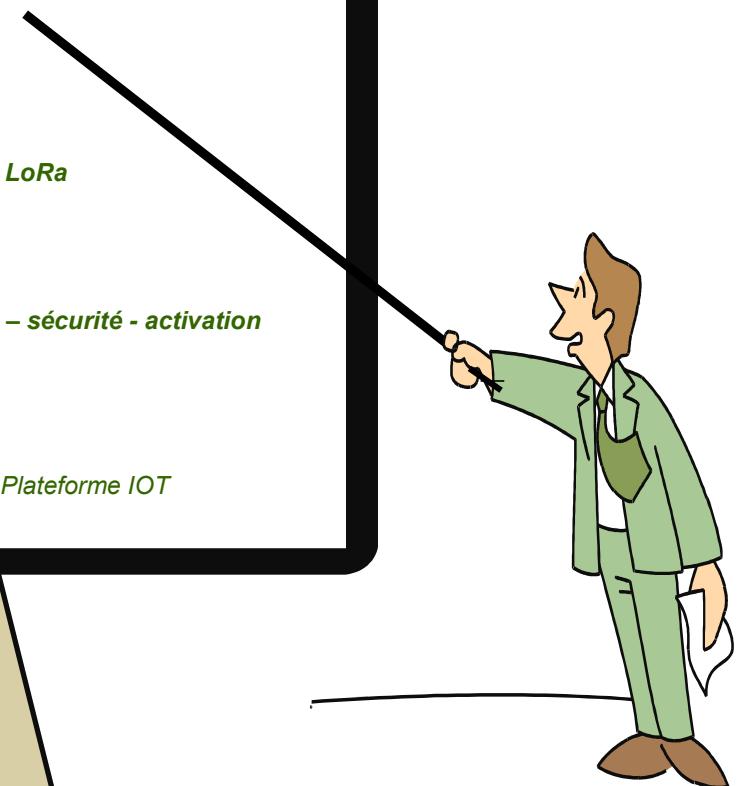
LORAWAN

LORAWAN : architecture – classes – sécurité - activation

P44

Application

réseau – objet – passerelle – TTN - Plateforme IOT





□ Se familiariser avec

- La couche physique LoRa
- Le protocole LoRaWan
- Le serveur de réseau public TTN
- Plateforme IOT d'affichage
- Mise en pratique



□ Métriques

- Soutenance Projet

Il s'agit de se familiariser avec la couche physique et le protocole LoRa dans l'optique de comprendre les paramètres de configurations rencontrés dans les logiciels que l'on va manipuler. La seconde partie sera plus pratique, il s'agira de mettre en place un réseau LoRa en utilisant les outils de développement aujourd'hui à notre disposition. Ce support fourni les briques de base permettant de réaliser par la suite votre projet IOT.

notes

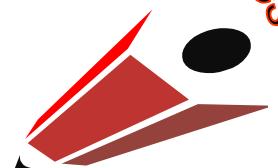


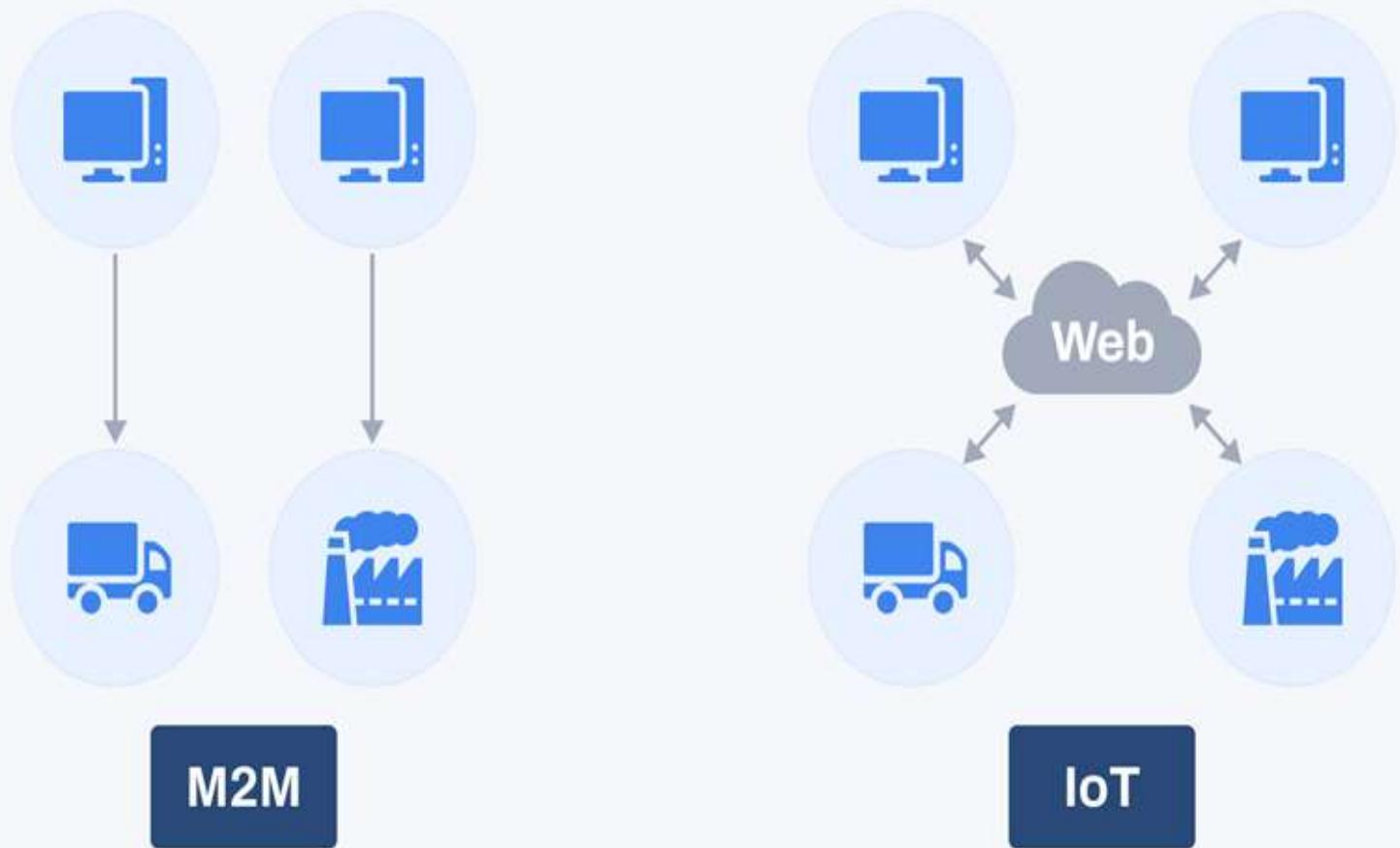


- <https://www.thethingsnetwork.org/>
- <https://lora.readthedocs.io/en/latest/>
- https://www.mobilefish.com/developer/lorawan/lorawan_quickguide_tutorial
- <https://www.frugalprototype.com/technologie-lora-reseau-lorawan/>
- <http://www.linuxembedded.fr/2017/12/introduction-a-lora/>
- **LoRa – LoRaWAN et l'Internet des Objets de Sylvain MONTAGNY**

Quelques liens qui peuvent vous aider à comprendre plus en détail les principes théoriques et les outils que nous allons utiliser.

notes





L'IoT est une évolution du M2M, qui intègre la connectivité Internet pour permettre aux objets de communiquer entre eux et avec les systèmes informatiques.

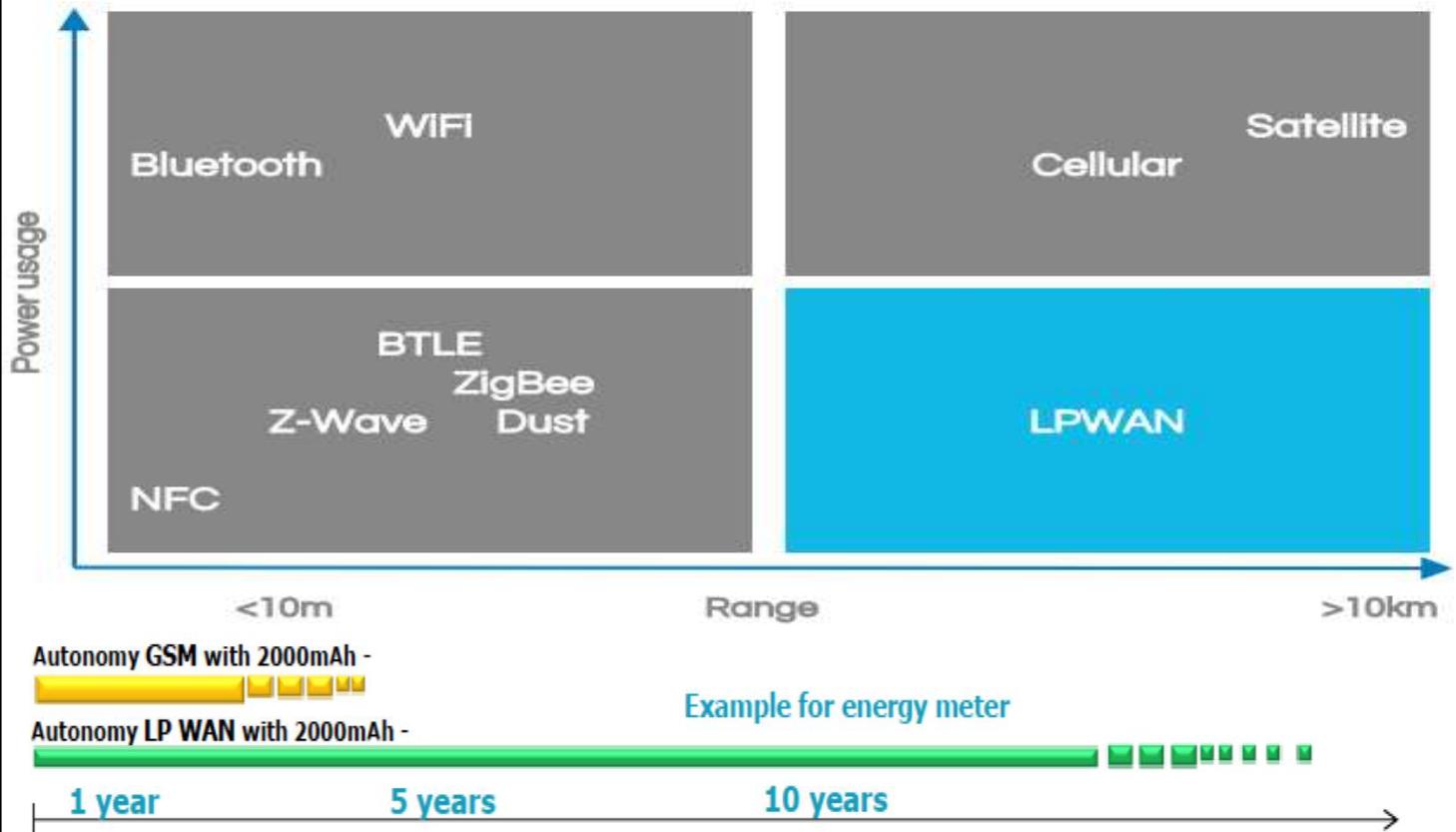
Le M2M est une technologie de communication machine-à-machine basée sur des protocoles propriétaires, tandis que l'IoT utilise des normes ouvertes et des protocoles de communication standard pour permettre aux appareils de communiquer de manière transparente entre eux et avec les systèmes informatiques. L'IoT est une vision plus large de la connectivité des objets, qui englobe une gamme d'applications et de services au-delà de la simple transmission de données.

notes



▪ Une consommation très faible

▪ Une communication très longue portée



Dans le monde de demain, tous les équipements électroniques seront connectés. Un réseau émergent dédié à l'IoT fait son essor depuis quelques années, il s'agit du réseau LPWAN (Low Power Wide Area Network).

Les réseaux LPWAN, comme le laisse deviner l'acronyme, sont des réseaux sans fils basse consommation, bas débit et longue portée, optimisés pour les équipements aux ressources limitées pour lesquels une autonomie de plusieurs années est requise. Ces réseaux conviennent particulièrement aux applications qui n'exigent pas un débit élevé.

Les LPWAN utilisent les bandes de fréquences à usage libre – sans licence – ISM (Industriel, Scientifique et Médical) disponibles mondialement, contrairement aux opérateurs mobiles qui utilisent des bandes sous licence pour lesquelles ils payent l'attribution des centaines de millions d'euros. Il est à noter que l'utilisation des bandes ISM implique le partage des ressources avec les concurrents et avec les autres technologies (RFID, WiFi, Bluetooth, ZigBee, etc.). Toutefois, ce n'est pas la jungle non plus : ces bandes de fréquences sont régulées par des autorités organisatrices et il est tenu de respecter des règles d'utilisation.

Le réseau LPWAN est encore peu connu, mais derrière lui se cachent des technologies plus médiatisées tels que LoRaWan et SigFox. Cette technologie permet d'émettre et recevoir des messages de très petites tailles, sur de très longues portées (de 5km à 40km), avec pour avantage majeur que les composants utilisés pour émettre ces messages sont très peu coûteux et très peu énergivores (il est donc possible avec une simple batterie, d'émettre quelques messages par jours pendant 10 ans).

Nous voyons émerger des modèles économiques alternatifs et attractifs avec des abonnements selon volume de 1 € à 20 € par an et sans coût data. Certes, ces solutions permettent de faire transiter qu'un faible volume de données par message, mais couvre largement plus de 80 % des besoins M2M et IoT. On parle de débits de l'ordre de quelques dizaines voire centaines de kilos octets par message. Grâce à ces technologies, nous sommes capables de transmettre des données issues d'objets divers sur de très longues distances (plusieurs kilomètres en champs libres).

notes



Sigfox

- Startup française fondée en 2009 (Toulouse)
- Protocole bidirectionnelle et une offre de service
- Débit (Data rate): 100 bits/s
- Portée 30 à 40 km (antenne de 2 m)
- 140 messages de 12 octets (*bytes*) par jour reçus du périphérique (*offre Platinum*)
- 4 messages par jour envoyés au périphérique
- ISM : 868 Mhz en Europe, 915 Mhz aux US
- Durée de transmission (30s /h) soit en 6 messages maximum par heure.

LoRa

- Longue portée (2 – 15 km)
- Robuste au bruit
- Taille des messages
- Faible consommation
- Débit adaptatif
- Communication bi directionnelle
- Fonctionne en mobilité rapide
- Supporte la géolocalisation
- Echanges de données sécurisés



Commençons les présentations avec Sigfox, une société pionnière sur le marché des LPWAN, arrivée en 2009 avec une technologie basée sur la transmission de signaux sur une bande ultra étroite (UNB pour Ultra Narrow Band) d'une centaine de Hz contre quelques centaines de kHz pour les réseaux cellulaires traditionnels.

Sigfox est un opérateur qui fournit sa technologie de connectivité bas débit au travers de son propre réseau cellulaire. Le réseau couvre la grande majorité du territoire français métropolitain et est déployé dans [plusieurs pays](#) d'Europe.

Un capteur, équipé d'un module et d'un abonnement au réseau Sigfox (1 à 15€ par an et par appareil), peut envoyer jusqu'à 140 messages par jour. Chaque message pouvant contenir jusqu'à 12 octets de données réelles de charge utile (l'identifiant du périphérique est transmis par le protocole), ce qui est largement suffisant pour la majorité des applications de monitoring.

Avec une portée comprise entre 30 et 50 Km dans les zones rurales et entre 3 et 10 km dans les zones urbaines, Sigfox offre un débit de 100 bits par seconde.

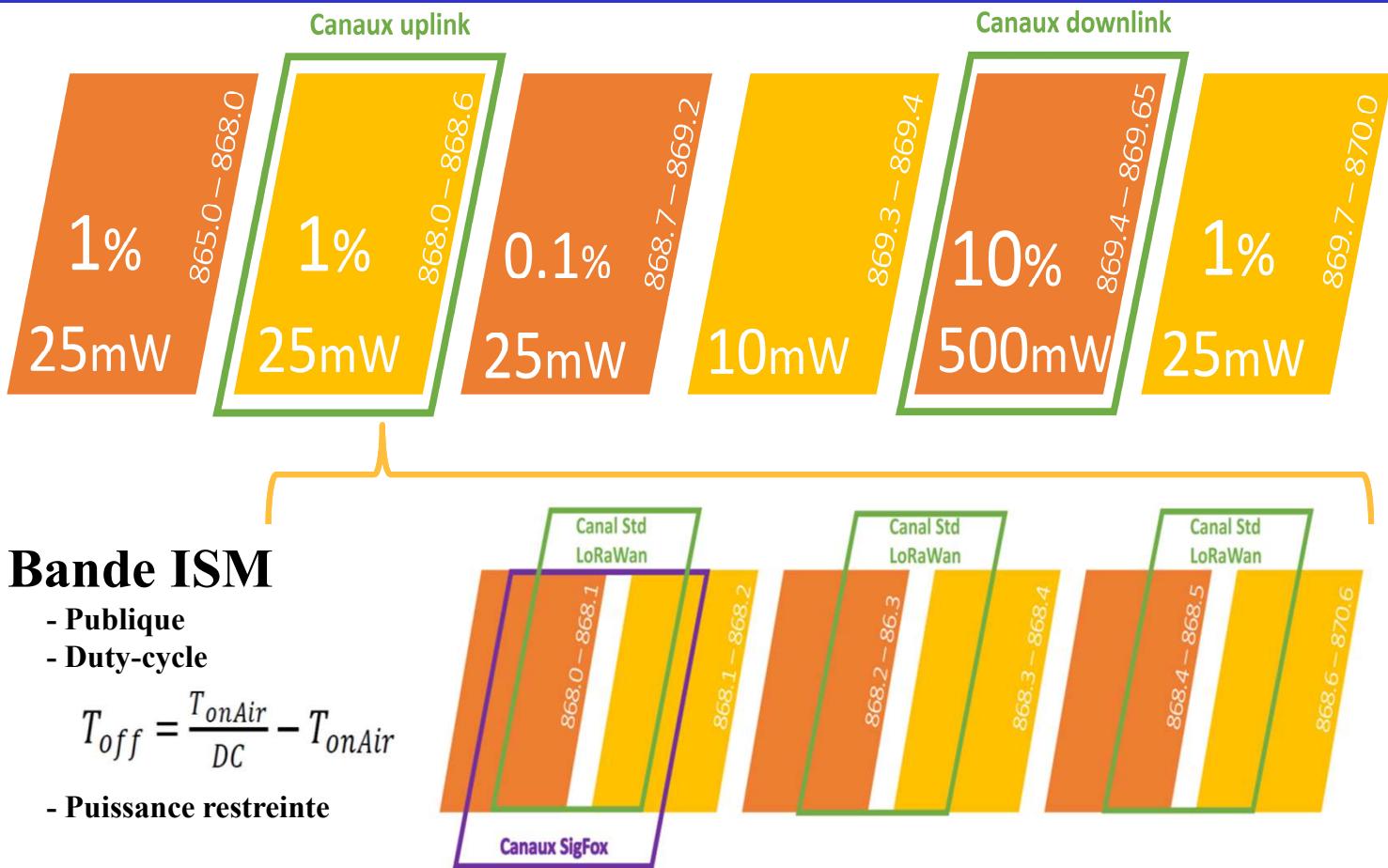
LoRaWan (Long Range Radio Wide Area Network) est un réseau LPWAN basé sur la technologie radio LoRa.

Développée par Cycleo en 2009 – rachetée 3 ans plus tard par l'américain Semtech – LoRa utilise une technique d'étalement de spectre pour la transmission des signaux radio.

La technologie LoRa à travers le réseau LoRaWan est poussée par un consortium d'industriels et d'opérateurs nommé LoRa Alliance regroupant entre autres IBM, Cisco, Bouygues Télécom, etc...

Sur un réseau LoRaWan, les données émises par les équipements sont centralisées par des gateway (des concentrateurs) qui transmettent les données à leur tour vers le serveur de gestion en ligne. La liaison entre les gateways et le serveur s'appuie sur des technologies très haut débit (Ethernet, Réseaux mobiles). LoRaWan revendique un débit adaptatif compris entre 0,3 et 50 kbits par seconde et une communication bidirectionnelle moins limitée que son concurrent direct SigFox.

notes



En Europe la bande Industriel Scientifique et Médicale (Sub-GHz) est dite la bande des 868 Mhz de 865 à 870 Mhz.

Les fréquences radio ISM (Sub-Ghz) sont publique, libre d'usage mais toutefois réglementées pour permettre à chacun de pouvoir les utiliser. La notion de coefficient d'utilisation limite horaire (aussi appelé Duty Cycle) vient restreindre le temps de parole de chaque objet, dans la majeure partie des cas 1%.

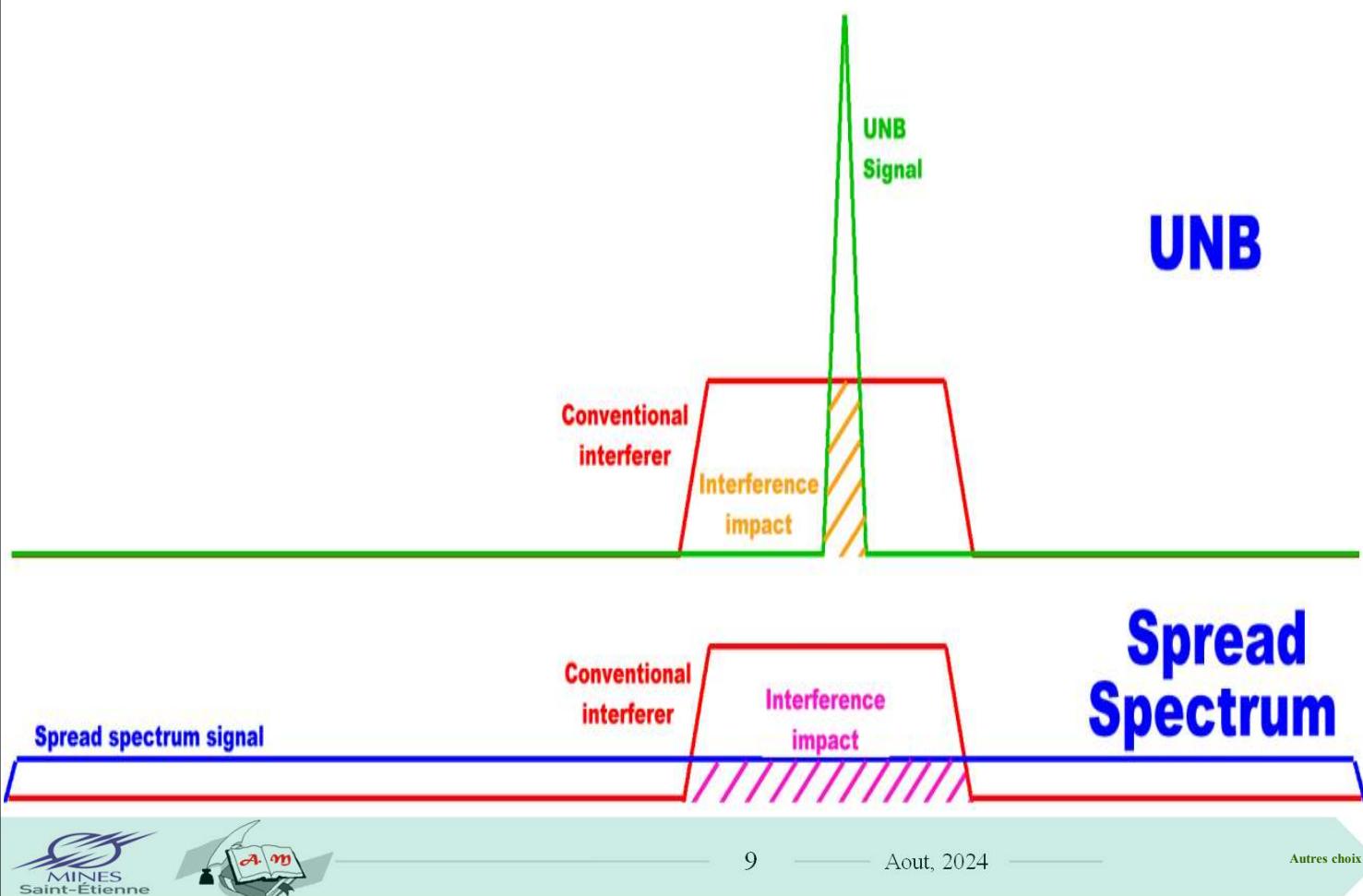
Un duty cycle de 1% signifie que si votre device communique pendant 1s, il devra se taire (low-power) pendant les 99s suivantes.

Notez la puissance autorisée dans cette bande de fréquence pour l'object en uplink : 25 mW de puissance d'émission. En downlink (de la passerelle vers l'objet) 500 mw sont autorisé.

Les Devices utilisent des canaux de fréquence pour séparer leurs transmissions, c'est ce qu'on appelle le FDM (Frequency Division Multiplexing). 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. Sigfox utilise 200 khz de bande passante dans laquelle il utilise 2000 canaux de 100 hz chacun. LoraWan utilise dans cette bande de fréquence 3 à 8 canaux de 125 khz chacun. Donc au maximum seul 8 devices peuvent transmettre en même temps. C'est bien sûr trop peu mais compte tenu du duty-cycle imposé de 1% les dispositifs émettent par intermittence et donc laisse les canaux libres pour les autres devices. Ce TDM (Time Division Multiplexing) augmente donc de 100 la capacité de transmission: 800 devices peuvent donc transmettre en même temps. La modulation

LoRa va ajouter sa pierre à l'édifice en permettant plusieurs transmissions sur le même Canal et donc démultiplier les capacités de transmission (4800 devices).

notes



Deux types de modulation en LPWAN: 2 approche pour un même objectif

SigFox

– Bande Ultra étroite : Ultra Narrow Band

émettre un signal sur une bande de fréquence la plus étroite possible pour maximiser la puissance en un point et passer au dessus du bruit

LoRa

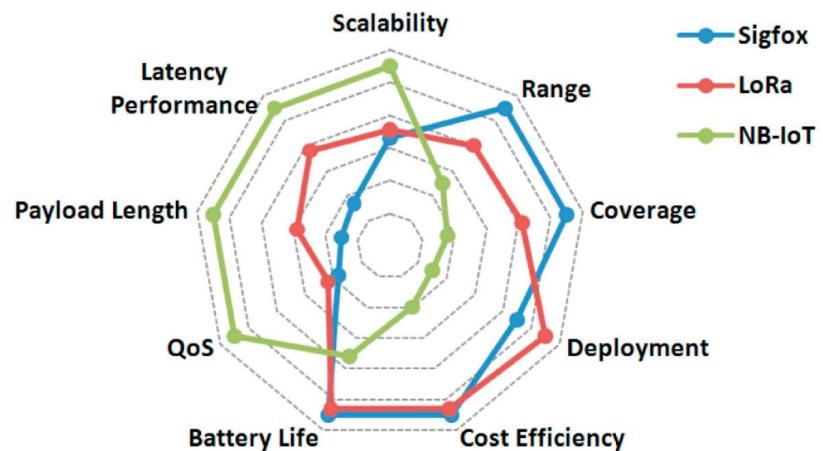
-- étalement du spectre

émettre le même signal sur plusieurs fréquences pour contourner le bruit

notes



- **LTE-M face à LoRa : débits pour l'un, consommation électrique pour l'autre**
- **Le principal intérêt de LTE-M : le roaming**
- **LTE-M : fréquences, déploiement**
- **La 5G et sa latence super basse**



Long-Term Evolution for Machines: LTE-M

Ces technologies sont dérivées de la 4G, LTE-M (développé par Orange) et NB IoT (développé par [SFR](#)).

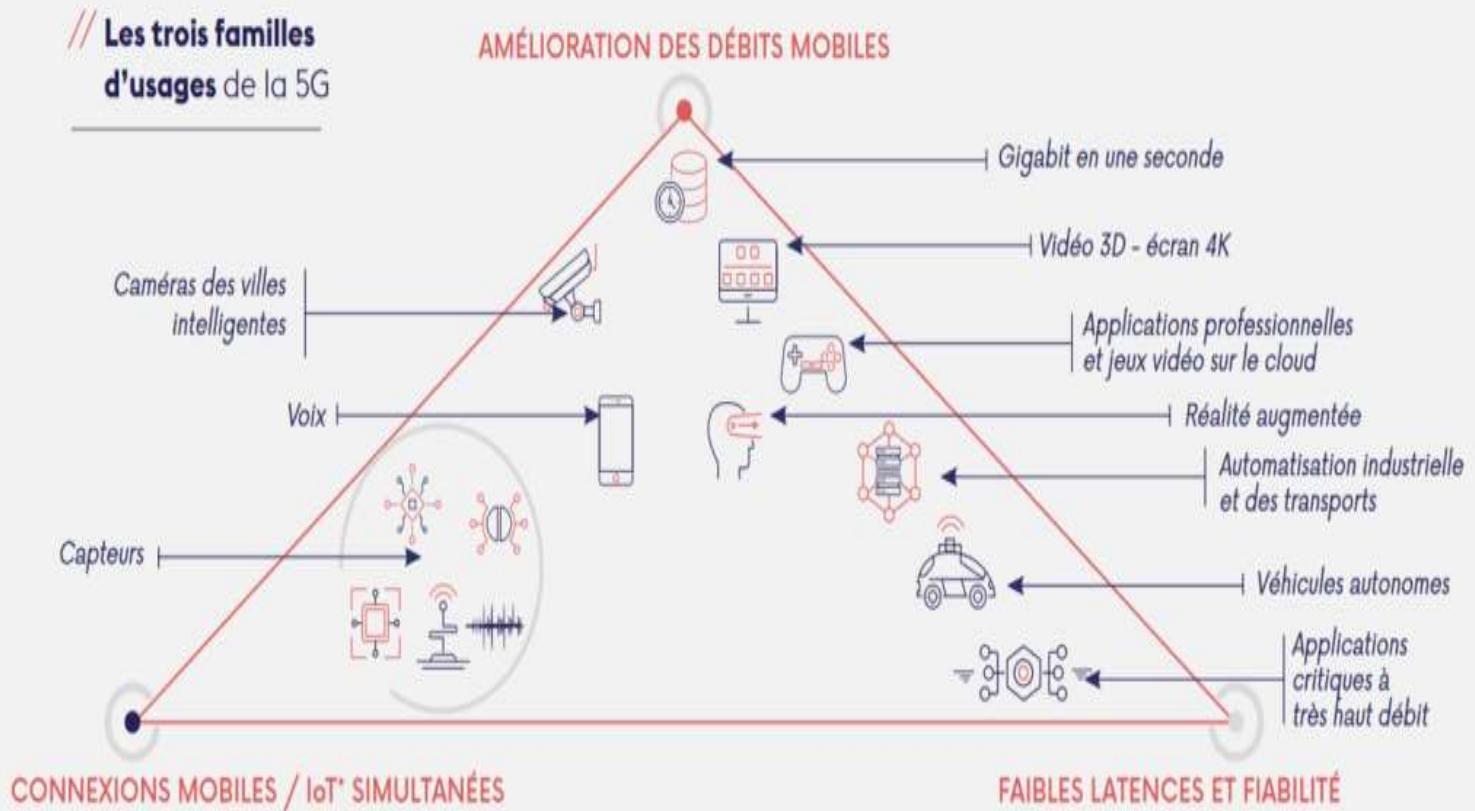
Certains usages complexes, comme le suivi d'actifs, la surveillance à distance, la télématique ou l'automobile, exigent cependant des débits supérieurs et une faible latence. La technologie LTE-M répond à ce besoin.

Le principal intérêt de LTE-M : le roaming, fréquences, déploiement

simple à mettre en œuvre : il suffit d'une mise à jour du logiciel pilotant les antennes et équipements réseaux 4G. Comme son nom l'indique, LTE-M se base sur les réseaux 4G existants, déjà déployés un peu partout dans le monde. À l'instar des smartphones, les objets connectés en LTE-M pourront profiter de l'itinérance (*roaming*), c'est-à-dire qu'ils seront capables de fonctionner dans d'autres pays, à condition que les opérateurs locaux aient également déployé cette technologie. Comme avec la 4G classique, cela dépendra des accords d'itinérance noués entre les uns et les autres.

Dans tous les cas, LoRa et LTE-M répondent à des besoins différents. Si l'objet ne quitte pas la France et/ou que l'autonomie est le critère principal, alors LoRa est tout désigné. Si vous avez besoin d'un débit plus soutenu, d'une faible latence et du roaming, alors il faudra partir sur LTE-M

notes



La 5G, le réseau multi-usage de demain

Un seul réseau pour les connecter tous. C'est la promesse IoT de la 5G dont les premiers déploiements ont eu lieu en 2020.

La 5G et sa latence super basse

Atout de la 5G : la latence. Cela désigne le délai de transit d'une donnée entre le moment où elle est envoyée et celui où elle est reçue. Celui-ci sera divisé par 10 par rapport à la 4G, avec un temps de réponse d'à peine une milliseconde. Cette réactivité est cruciale pour l'industrie, car des échanges constants et quasi-immédiats sont requis pour faire émerger des usages comme le transport autonome.

Troisième point fort de la 5G : la densité. Avec elle, la 5G supportera « *un nombre très important de connexions mobiles simultanées* ». Cela va « *multiplier par 10 le nombre d'objets connectés au réseau simultanément* », confirme l'agence nationale des fréquences. En clair, il s'agit d'éviter l'engorgement des réseaux à l'heure où tout devient connectable et que les capteurs pullulent.

Si nous partons de l'hypothèse qu'un objet connecté autonome a une durée de vie de 10 ans, l'utilisation d'un objet installé en 2020 avec une solution Lora ou Sigfox durera jusqu'en 2030. Si la 5G tient ses promesses, alors l'avenir de ces deux solutions au-delà de cette échéance est fortement compromis.

Pour conclure, **Lora et Sigfox resteront les pionniers** qui ont permis le déploiement en masse des premiers objets connectés. Mais les deux sociétés vont devoir innover et faire évoluer leurs propositions de valeur **d'ici à 2030** pour pouvoir se différencier de la 5G en Europe.

notes



P2

Introduction*Organisation - définition - demain*

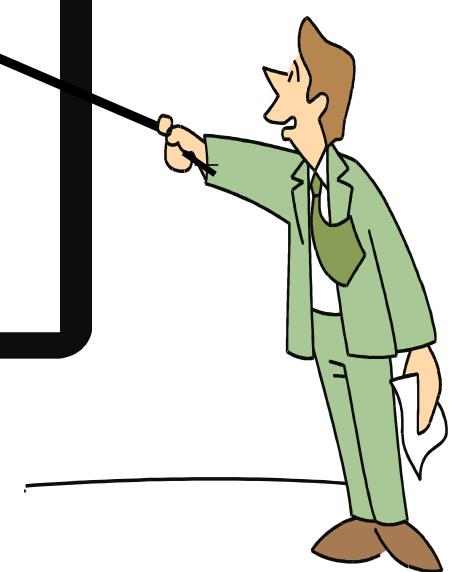
P12

LORA*Etalement de spectre – modulation LoRa*

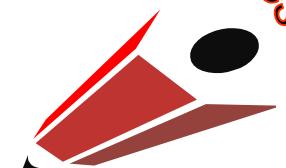
P24

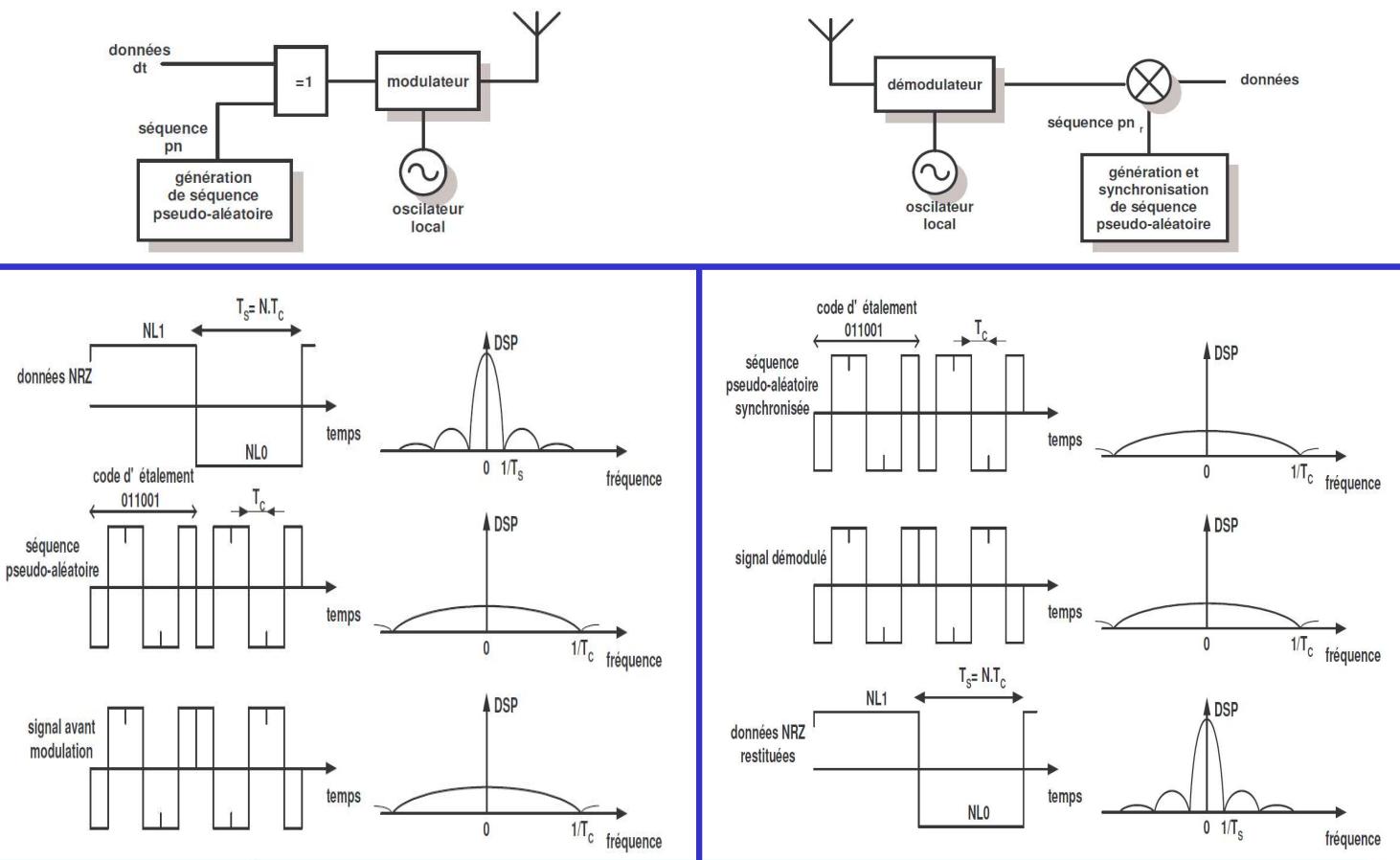
LORAWAN*LORAWAN : architecture – classes – sécurité - activation*

P44

Application*réseau – objet – passerelle – TTN - Plateforme IOT*

notes



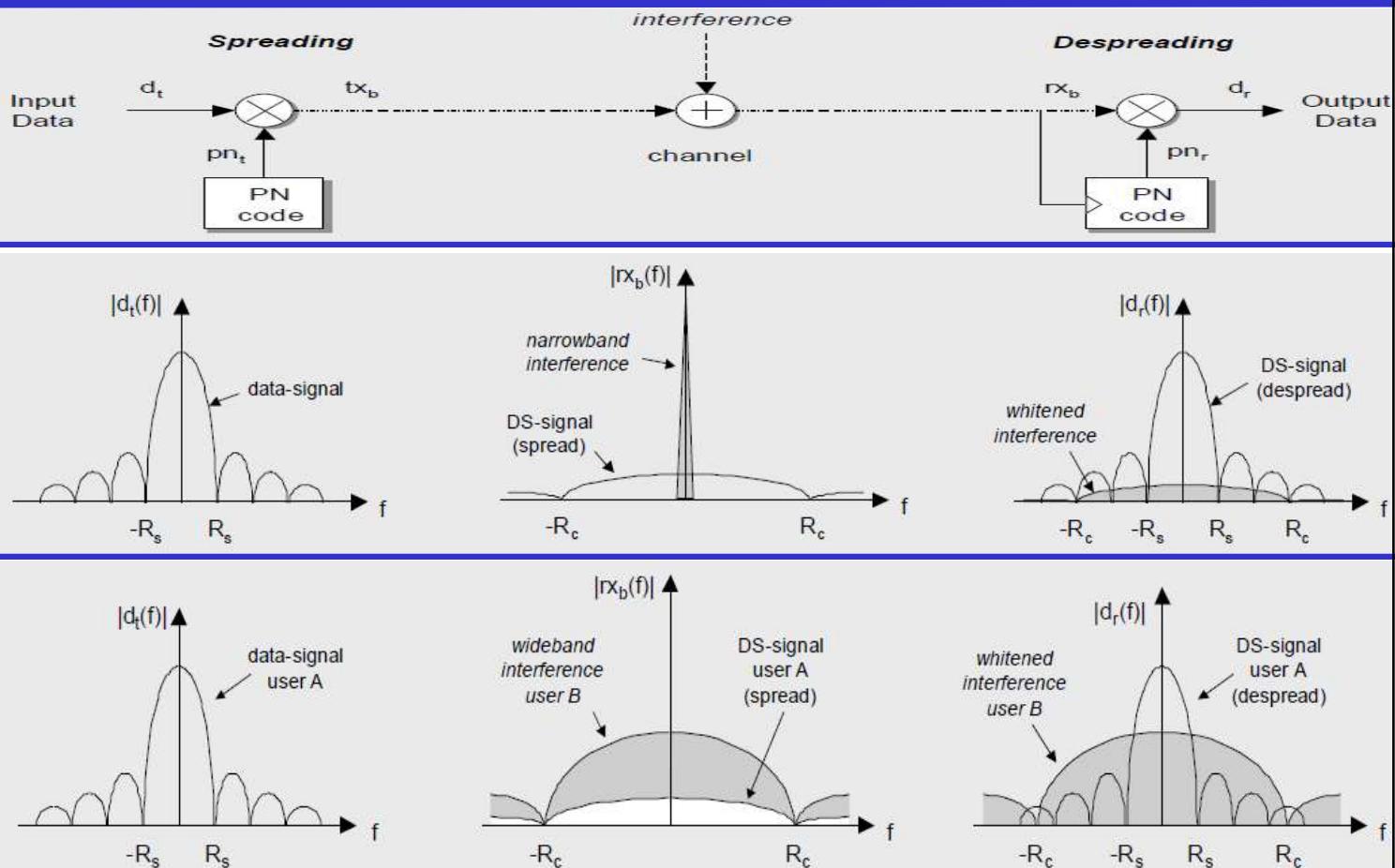


La transmission d'information par canal hertzien utilise classiquement des systèmes fonctionnant à bande de fréquence étroite, ce qui permet un multiplexage efficace du canal. Dans le système à étalement de spectre comme LoRa, on va chercher au contraire à ce que le signal radiofréquence occupe un spectre le plus large possible, ce qui à puissance d'émission égale donne une densité spectrale de puissance proportionnellement plus faible, de l'ordre de grandeur du bruit ambiant. Les premières applications étaient de type militaire, le signal ainsi émis étant presque impossible à détecter par un récepteur non autorisé et très difficile à brouiller.

Afin que la figure soit lisible, on a pris un rapport 6 entre F_c et F_s . Dans la pratique, ce rapport est beaucoup plus élevé, la densité spectrale de puissance du signal NRZ pouvant être vu comme une impulsion de Dirac par rapport à celle de la séquence d'étalement. La multiplication effectuée dans le domaine temporel se traduisant par une convolution dans le domaine fréquentiel, le Dirac étant l'élément neutre de l'opération de convolution, le spectre du signal étalé a presque la même forme que celui de la séquence d'étalement.

Pour retrouver le signal, le récepteur doit connaître la séquence pseudo-aléatoire et être capable de la synchroniser avec celle du signal reçu.

notes



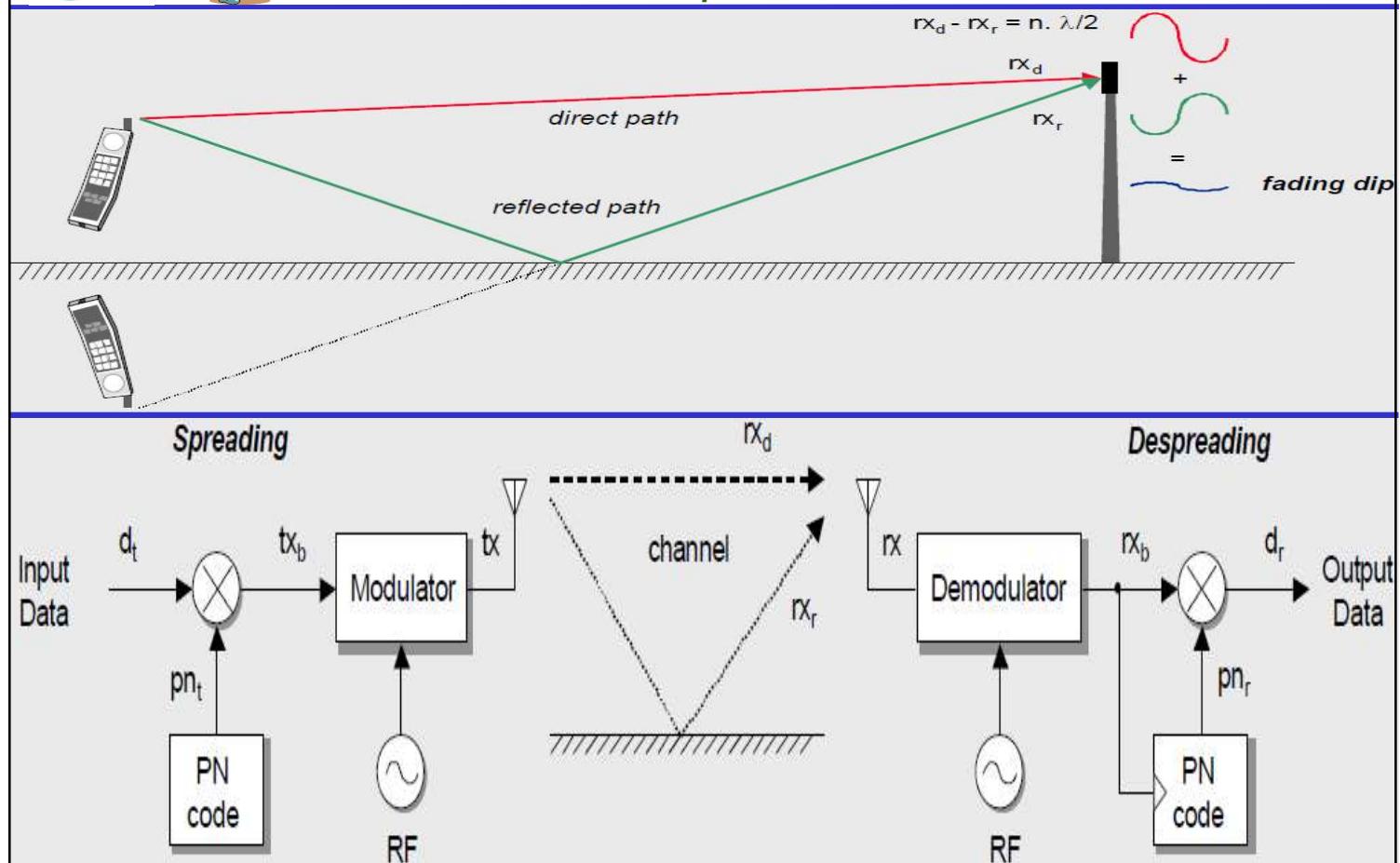
Une interférence en bande étroite, verra son spectre étalé par l'opération de désétalement ou désembrouillage (despread) et sa densité spectrale de puissance (DSP) diminuer d'autant; son influence sera alors extrêmement réduite.

Une interférence en bande large, même de DSP supérieure au signal utile verra aussi son spectre étalé et sa DSP diminuer au niveau du récepteur, alors que le signal utile subira l'opération inverse.

Sur une même bande de fréquence, plusieurs signaux peuvent être transmis, à des utilisateurs différents, les signaux étant étalés par des codes différents; il suffit que chaque récepteur possède le code nécessaire pour lire le message qui lui est destiné. On parle alors d'accès multiple avec répartition par code ou CDMA (Code Division Multiple Access).

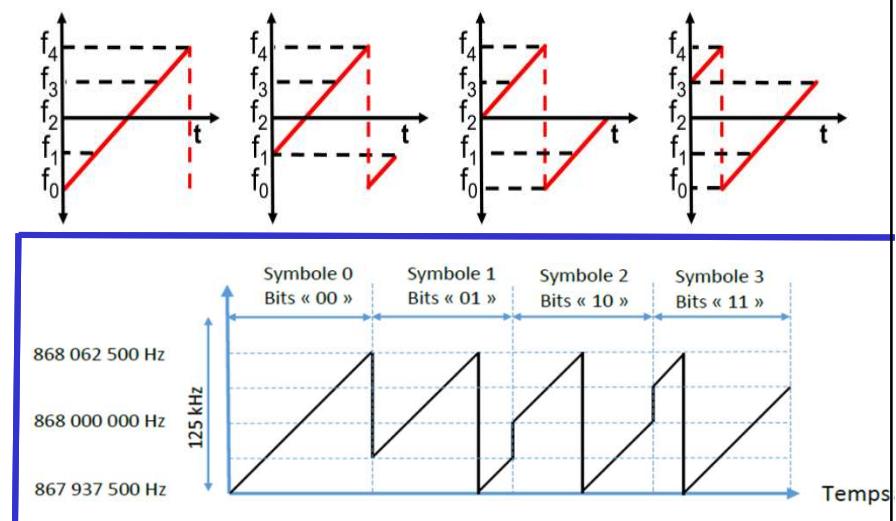
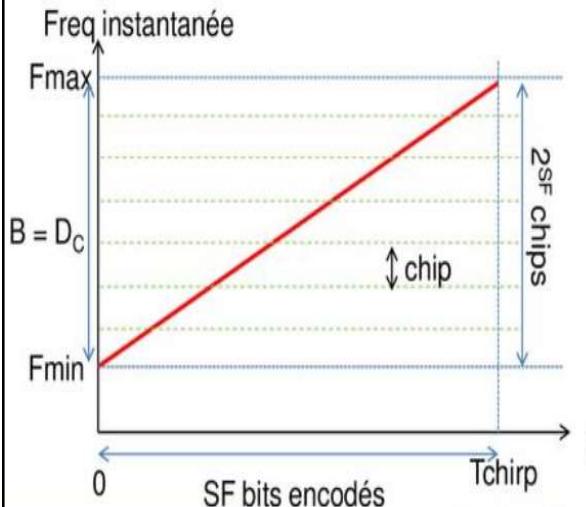
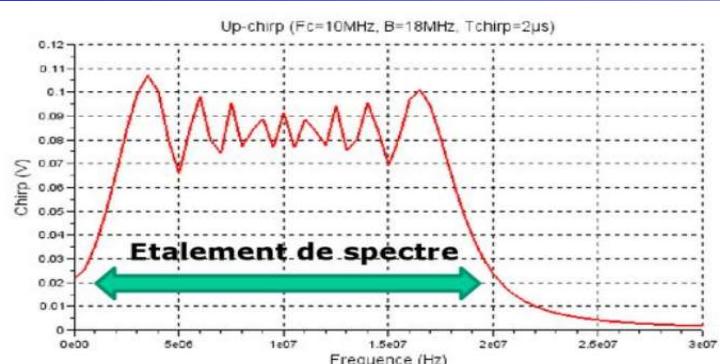
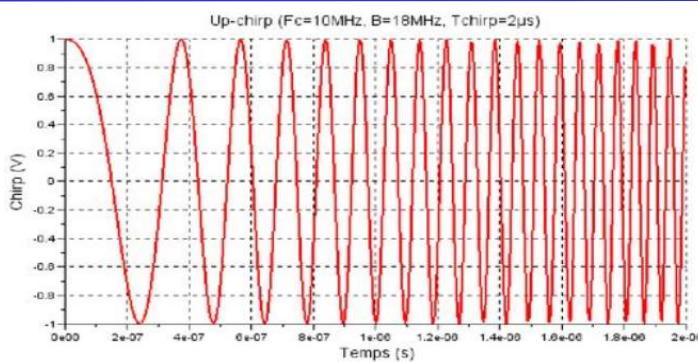
Bien que les méthodes de modulation par étalement du spectre les plus connues utilisent des "codes" pour atteindre ce résultat, Le LoRa utilise à la place, des symboles (appelés Chirp), d'où son nom : "Chirp Spread Spectrum (CSS) modulation".

notes



Lorsque le signal hertzien se propage sur des trajets multiples, les différents échos reçus au niveau du récepteur peuvent être vus comme une perturbation large bande; n' étant pas synchronisé avec la séquence du récepteur, ils subiront aussi un étalement.

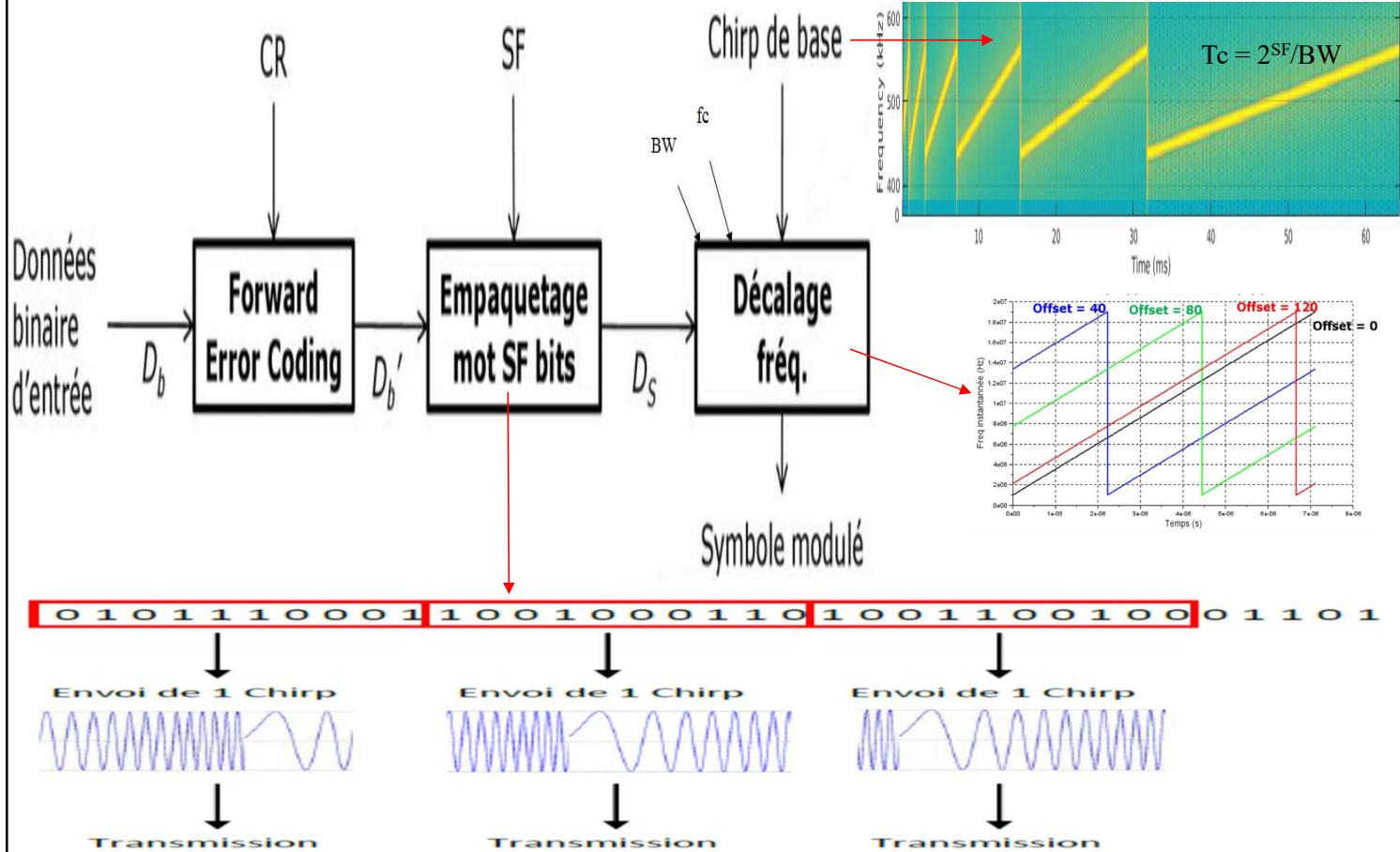
notes



Le LoRa utilise une méthode d'étalement de spectre qui n'est pas exactement celle que nous venons d'étudier. La finalité est cependant la même : Pouvoir transmettre en même temps, sur le même canal. Au lieu d'utiliser des codes d'étalement (CDMA), elle utilise une méthode appelée Chirp Spread Spectrum.

Le signal émis par la modulation LoRa est un symbole dont la forme de base est représentée ci-dessus. Son nom (Chirp) vient du fait que ce symbole est utilisé dans la technologie Radar (**Chirp** : Compressed High Intensity Radar Pulse). Pour faciliter la représentation de ce symbole, on utilise plutôt un graphique Temps/Fréquence. La fréquence centrale est appelée canal, la bande passante est la largeur de bande occupée autour du canal ($F_{max} - F_{min} = 125\text{ kHz}$). Le **CHIRP** est le symbole de base, chaque symbole représente un certain nombre de bits transmis (**Spreading Factor**). Les bits sont regroupés par paquet de SF bits, puis chaque paquet est représenté par un symbole particulier parmi 2^{SF} formes de symboles possibles. Les différents symboles correspondants aux différents paquets de bits sont différenciés par l'offset du saut de fréquence dans le chirp. Le nombre de positions possibles d'offsets fréquentiel dépend du paramètre SF et est égale à 2^{SF} . Ces positions d'offset fréquentiel sont appelées Chips. Voir exemple avec SF = 2.

notes



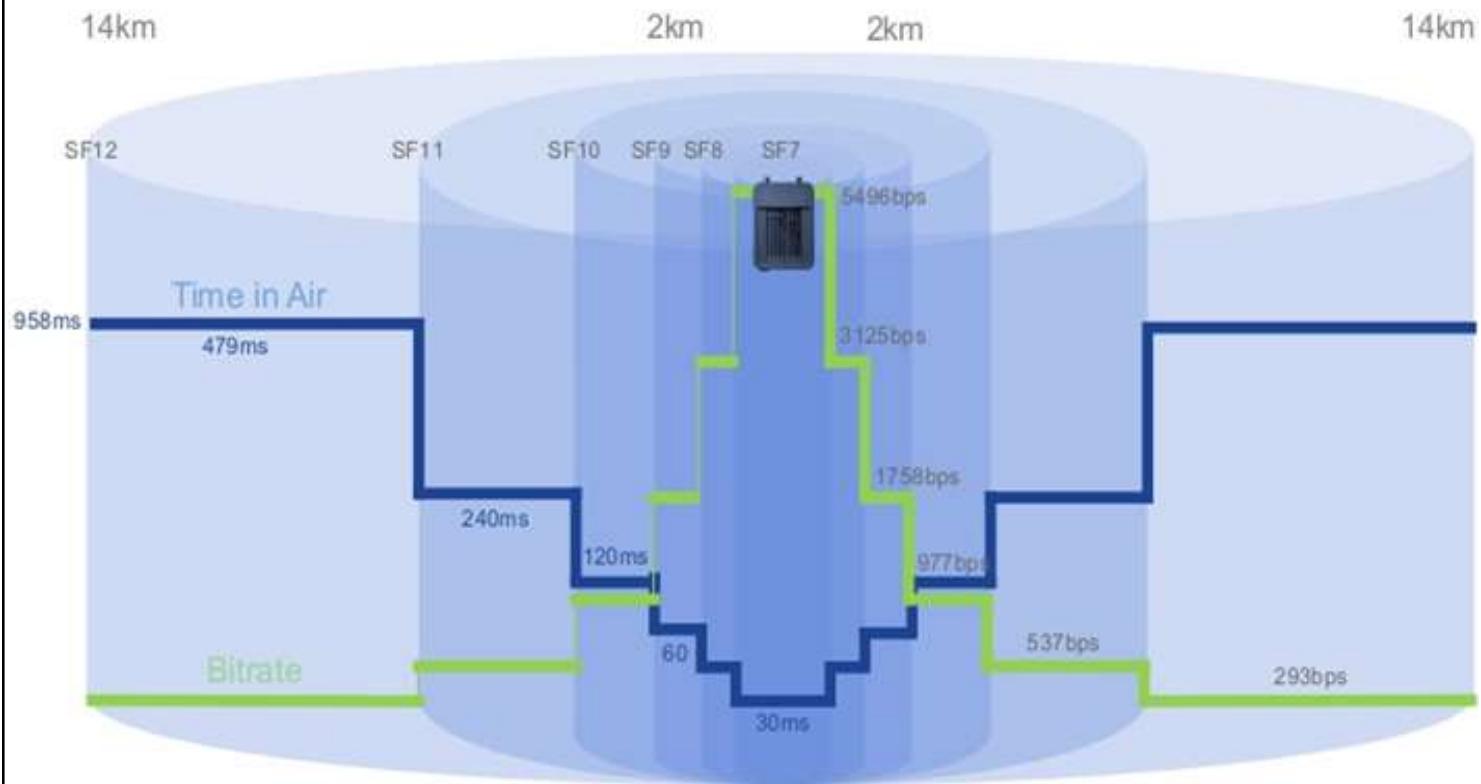
3 paramètres sont réglables:

- SF = Spreading Factor (nombre de bits par symbole) de 7 à 12
- B = bande passante du signal chirp (125, 250 et 500 khz au US)
- CR = Coding Rate nombre de bits insérés dans les données d'entrées en vu de détection/ correction d'erreur exprimé en ratio: 4/5, 4/6, 4/7, 4/8. Un coding rate de 4/8 entraîne 8 bits transmis réellement pour 4 bits de donnée effective. Avantage d'un coding rate élevés: le système peut détecter et auto-corriger un certain nombre d'erreur dans la trame sans retransmission du signal.

Classiquement le LoRa utilise 6 SF (SF7 à SF12) donc permet la transmission de 6 devices par canal distingués par leur SF, soit 48 devices en tout. Si on tient compte du temps multiplexé (duty cycle de 1%) on obtient bien une capacité de transmission de 4800 devices simultanées.

Cependant ces Spreading Factor ont également un cout il augmente la durée d'émission de chaque symbole (Chirp) et donc la consommation pour les SF élevés. Plus le SF est grand et plus le temps d'émission sera long, le débit faible et la consommation élevé.

notes



$$\text{Débit} = \text{SF} \cdot \text{BW}/2^{\text{SF}}$$

LoRaWAN supporte 6 facteurs d'étalements (de SF7 à SF12). La figure suivante présente le *débit utile*, la portée en fonction du *facteur d'étalement*.

La formule précédente ne tient pas compte du coding rate, c'est pourquoi les débits sont plus faibles en pratique:
 $\text{Débit} = \text{SF} \cdot \text{BW}/2^{\text{SF}} \cdot \text{CR}$

Les coding rate varient de 4/5 à 4/8.

notes



Spreading Factor	Data Rate (bit/s)	Time on Air (ms)	Maximum Payload Size	End-device sensitivity (dBm)
SF12	250 DR 0	1400	59 bytes	-137
SF11	440 DR 1	740	59 bytes	-135
SF10	980 DR 2	370	59 bytes	-133
SF9	1760 DR 3	200	123 bytes	-130
SF8	3125 DR 4	100	250 bytes	-127
SF7	5470 DR 5	28	250 bytes	-124

Plus le Spreading Factor est élevé, plus le débit binaire est faible.

Plus la bande passante est élevée, plus le débit binaire est élevé.

La combinaison de SF et BW est appelée DR (Data Rate). Dans la bande EU868, il est normalisé de DR0 à DR6. DR6 correspond à un SF7 et une bande passante de 250 khz.

notes



Émetteur

 P_T : Puissance transmise

Air

Récepteur


 P_R : Puissance Reçue
+
 P_B : Puissance du Bruit

Bruit

RSSI – Received Strength Signal Indicator



- Le dBm est la puissance par rapport à 1 mW : 0 dBm correspond à 1 mW
- Le RSSI (Received Signal Strength Indication) est la puissance reçue P_R
- Le SNR (Signal over Noise Ratio) est le rapport entre la puissance reçue (P_R) et la puissance du bruit (P_B).

Bande de fréquence EU



20

Aout, 2024

Le RSSI est la puissance du signal reçu en milliwatt et mesuré en dBm. Cette valeur peut être interprétée comme une mesure de la qualité de la réception, comment un récepteur entend le signal envoyé. Un RSSI de 0 correspond à une réception maximale. Sur LoRa le RSSI est compris entre -30dBm et -120dBm

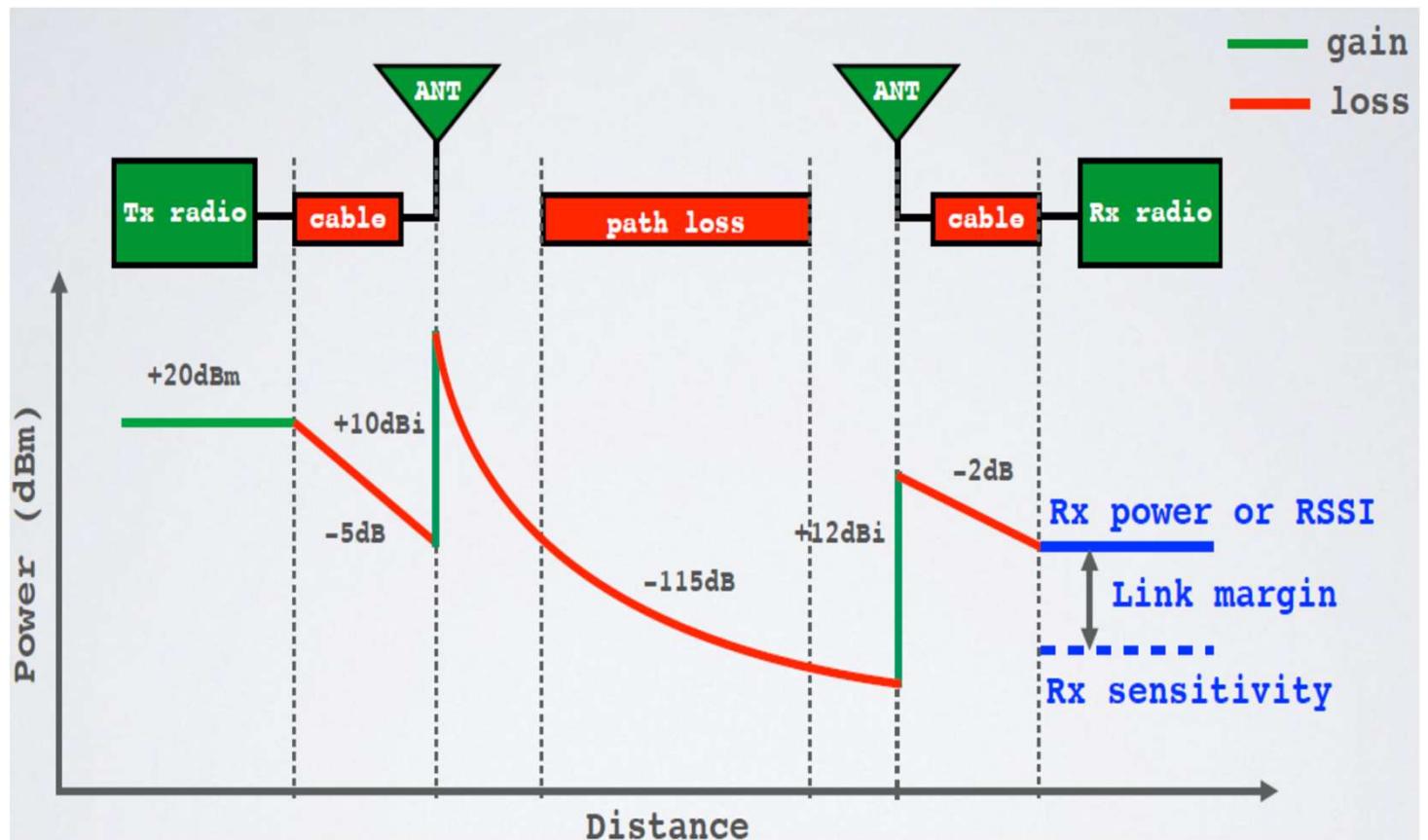
Le rapport signal sur bruit (SNR) est le rapport entre la puissance du signal reçu et la puissance du bruit de fond. Plus ce rapport est grand et plus il sera facile d'extraire le signal du bruit.

Remarque: Normalement le bruit de fond est la limite physique de la sensibilité, cependant Lora peut démoduler des signaux inférieurs jusqu'à -20 dB au bruit de fond.

$$P(\text{dBm}) = 10 \log(P(\text{mw})/1\text{mw})$$

$$13 \text{ dBm} = 20 \text{ mW}$$

notes



La sensibilité est la puissance PR 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.

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.

Au final, c'est la différence entre la puissance émise Pt et la sensibilité du récepteur qui importe. C'est ce qu'on appelle le bilan de liaison (Link budget).

notes

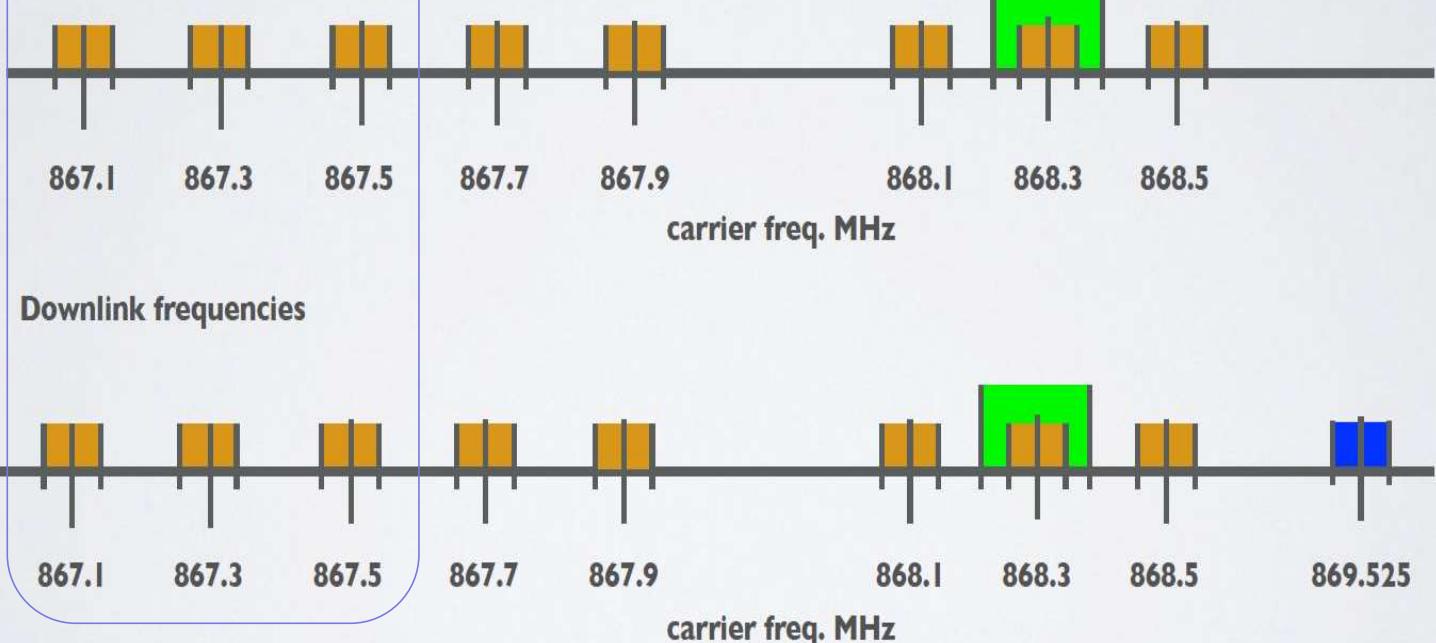


LoRaWan frequencies used in Europe

TTN freq. plan: EU863-870

Default Channel

Uplink frequencies



Downlink frequencies



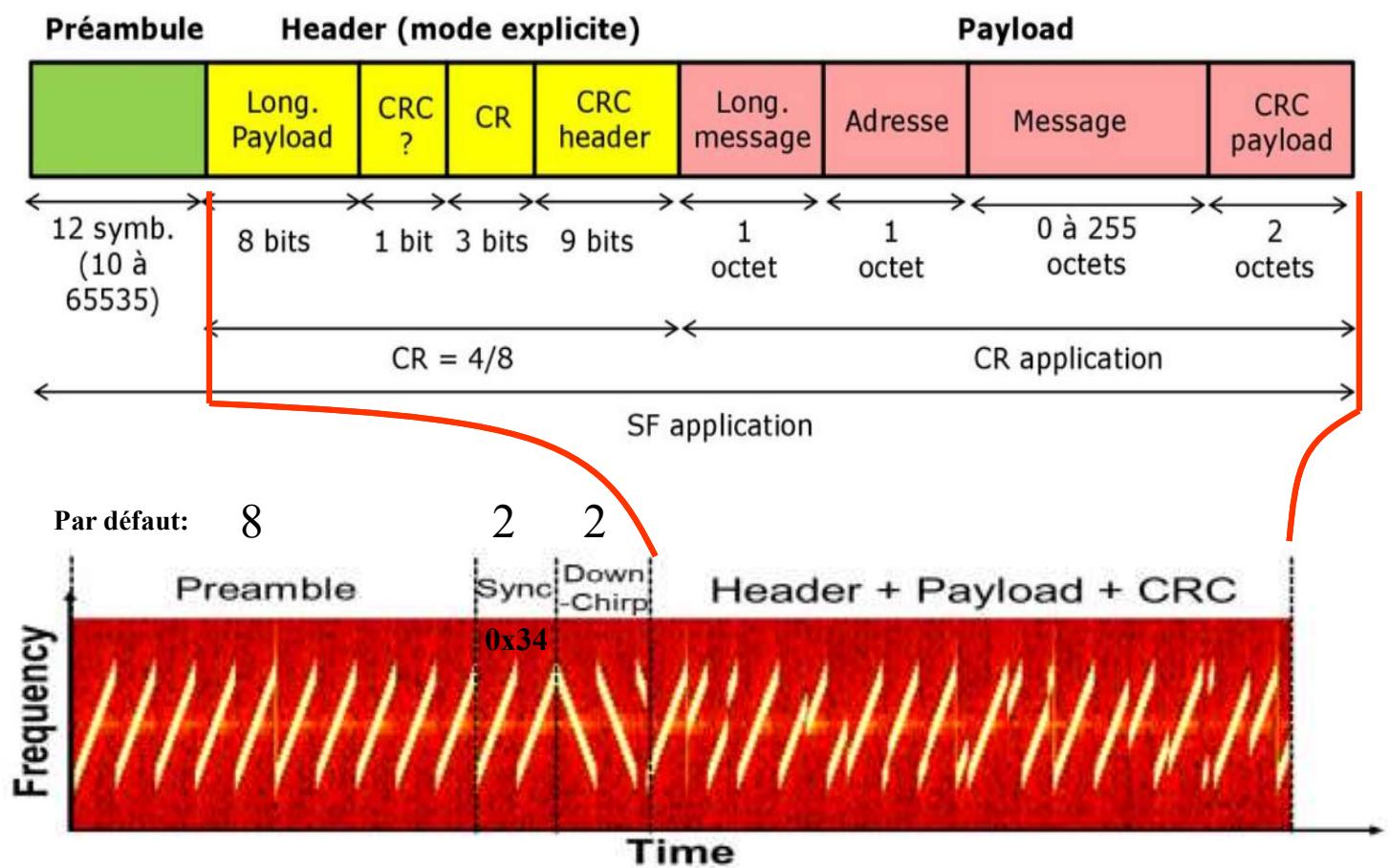
Le choix du moment d'émission des Devices LoRa se fait de façon simple : Lorsqu'un équipement doit émettre, il le fait sans contrôle et ne cherche pas à savoir si le canal est libre. Si le paquet a été perdu, il le retransmettra simplement au bout d'un temps aléatoire (ALOHA).

En uplink 8 canaux sont utilisés dont 3 imposés et 5 libres attribuées par l'opérateur du réseau. En downlink, pour le slot1 on utilise le même canal que lors du uplink donc 8 canaux possibles. Pour le slot2 seul le canal 869.525 Mhz est possible.

Remarque: en downlink, il existe 2 slots temporels pour recevoir.

Un end-node change de canal de manière pseudo-aléatoire pour chaque transmission. La modification des fréquences rend le système plus résistant aux interférences.

notes



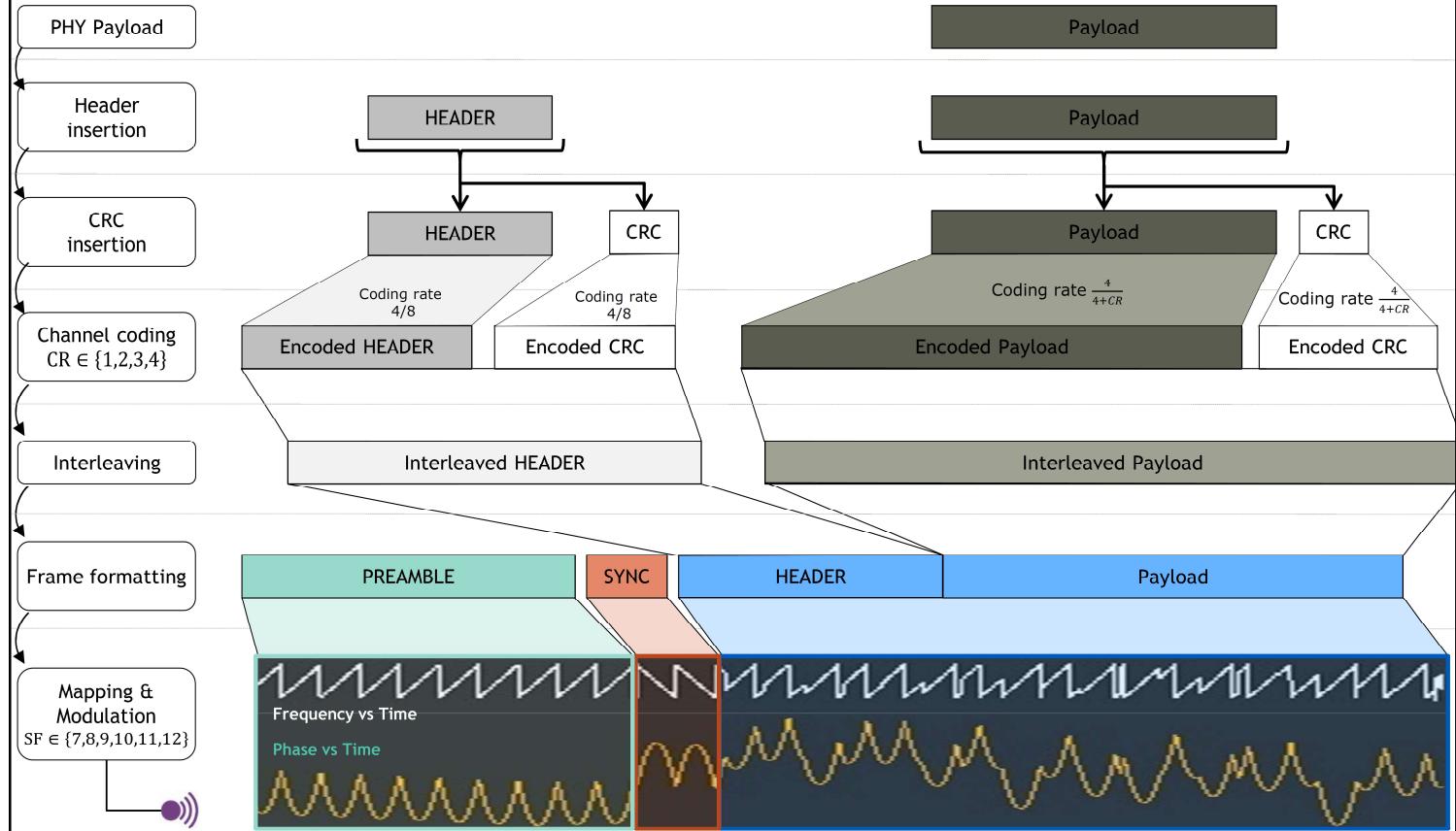
La trame LoRa est constituée de trois parties distinctes, modulées différemment : préambule (preamble), header, payload.

La trame commence par une séquence contenant un nb implicite de symboles non modulés (ou symboles de base) pour assurer la synchro du récepteur sur la fréquence et la base de temps de l'émetteur. Il est suivi de 2 symboles modulés qui ont la valeur du mot de synchronisation (par convention 0x34 avec LoRaWAN). Le préambule se termine par deux symboles down-chirp non modulés. Le nombre de symboles du préambule est paramétrable et est fixé par défaut à 12 (8 symboles de base + 2 synchronisation + 2 symboles inversés). Le mot de synchronisation constitue la signature du réseau LoRa.

A la suite du préambule, après une pause de longueur un quart de symbole, se trouve l'entête physique du message. Il est composé de 21 bits avec : la longueur du payload sur 1 octet, 1 bit pour indiquer la présence d'un CRC sur le payload, 3 bits pour indiquer la valeur du coding rate, puis 9 bits de CRC sur le header. L'entête est par convention protégée par le coding rate maximal 4/8. Deux modes de transmission du header sont prévus : explicite (toutes les infos précédentes sont transmises) ou implicites (le header n'est pas transmis car ces infos sont déjà connus à l'avance).

Le payload est composé de symboles modulés. Il est encodé avec le CR paramétré par l'application. Le payload se termine par un champ CRC de 2 octets si cela a été indiqué dans le header. Le payload contient au minimum 2 octets : un octet indiquant la longueur du message puis un octet d'adresse. Le message qui suit peut contenir de 0 à 255 octets.

notes



La trame LoRa est constituée de trois parties distinctes, modulées différemment : préambule (preamble), header, payload.

La trame commence par une séquence contenant un nb implicite de symboles non modulés (ou symboles de base) pour assurer la synchro du récepteur sur la fréquence et la base de temps de l'émetteur. Il est suivi de 2 symboles modulés qui ont la valeur du mot de synchronisation (par convention 0x34 avec LoRaWAN). Le préambule se termine par deux symboles down-chirp non modulés. Le nombre de symboles du préambule est paramétrable et est fixé par défaut à 12 (8 symboles de base + 2 synchronisation + 2 symboles inversés). Le mot de synchronisation constitue la signature du réseau LoRa.

A la suite du préambule, après une pause de longueur un quart de symbole, se trouve l'entête physique du message. Il est composé de 21 bits avec : la longueur du payload sur 1 octet, 1 bit pour indiquer la présence d'un CRC sur le payload, 3 bits pour indiquer la valeur du coding rate, puis 9 bits de CRC sur le header. L'entête est par convention protégée par le coding rate maximal 4/8. Deux modes de transmission du header sont prévus : explicite (toutes les infos précédentes sont transmises) ou implicites (le header n'est pas transmis car ces infos sont déjà connus à l'avance).

Le payload est composé de symboles modulés. Il est encodé avec le CR paramétré par l'application. Le payload se termine par un champ CRC de 2 octets si cela a été indiqué dans le header. Le payload contient au minimum 2 octets : un octet indiquant la longueur du message puis un octet d'adresse. Le message qui suit peut contenir de 0 à 255 octets.

notes



- Un émetteur fournit une puissance de 13 dBm en utilisant une antenne avec un gain de 2 dBm. La perte dans l'air est de 60 dBm. L'antenne de réception a un gain de 2 dBm et est connectée à un récepteur avec une sensibilité de -80 dBm. Le signal sera-t-il reçu ?
- Quel est le bilan de liaison de LoRa ?
- Comparer avec le bilan de liaison LTE (4G) : 130 dBm
- On considère les 2 cas suivants [SF7, 125 kHz, CR 4/5] et [SF12, 125 kHz, CR 4/5]. Donner les débits binaires brut correspondant .
- Vérifiez les calculs "Equivalent Bitrate" avec le LoRa Modem Calculator.
- Calculer le débit net avec un payload de 1 octet
- Quel est la période minimale d'utilisation dans ce cas ?

Vous trouverez le LoRa Modem Calculator dans votre répertoire de travail.

En pratique le débit net est beaucoup plus faible car le protocole LoRaWan rajoute un entête d'au moins 13 octets. On peut donc constater que le LoRa n'est pas adapté pour le temps réel.

notes



P2

Introduction*Organisation - définition - demain*

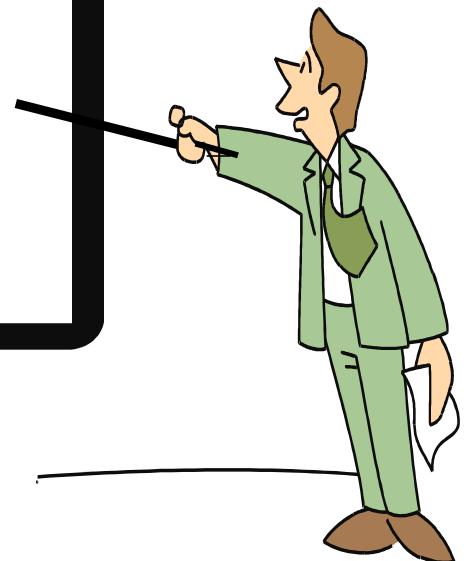
P12

LORA*Etalement de spectre – modulation LoRa*

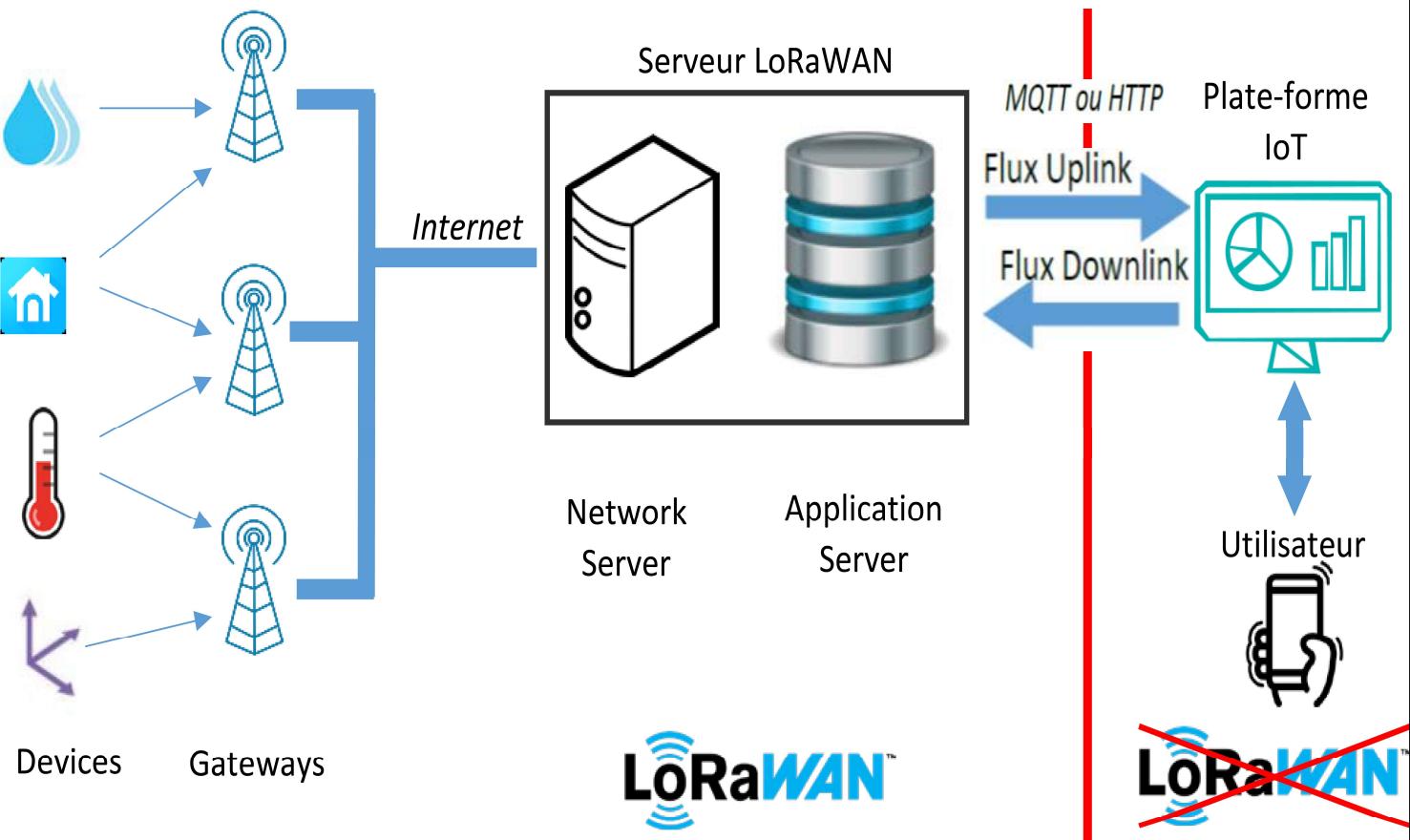
P24

LORAWAN*LORAWAN : architecture – classes – sécurité - activation*

P44

Application*réseau – objet – passerelle – TTN - Plateforme IOT*

notes



LoRaWAN™

Les end-devices ne communiquent pas exclusivement avec une passerelle définie mais avec l'ensemble des concentrateurs qui les couvrent.

Les Gateways écoutent sur tous les canaux, et sur tous les Spreading Factor, soit 48 combinaison possibles (8 canaux, 6 SF). Lorsqu'une trame LoRa est reçue, elle transmet son contenu sur internet à destination du Network Server qui a été configuré dans la

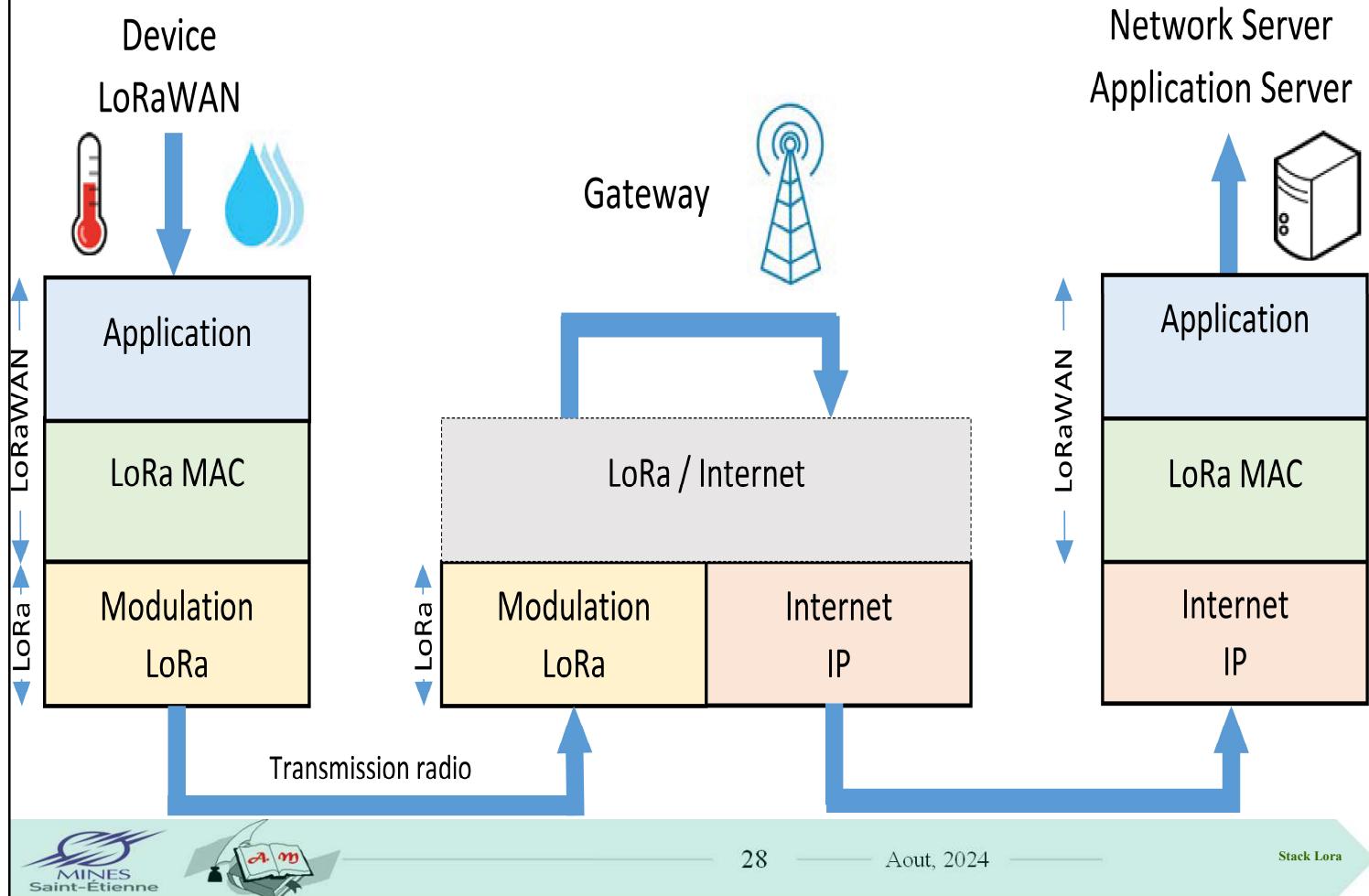
Gateway au préalable. Elle joue donc le rôle de passerelle entre une modulation LoRa, et une communication IP. Le Network Server reçoit les messages transmis par les Gateways et supprime les doublons (plusieurs Gateway peuvent avoir reçu le même message). Le serveur réseau assure la gestion de la sécurité, du débit adaptatif, de la redondance et communique avec des serveurs applicatifs qui exploitent les données en provenance des end-devices.

L'application server est souvent sur le même support physique que le Network Server. Il permet de dissocier les applications les unes des autres. Chaque application enregistre des Devices LoRa qui auront le droit de stocker leurs données (Frame Payload)

Cette application doit d'une part récupérer les données de l'Application Serveur, avec le protocole HTTP ou le protocole MQTT et d'autre part mettre à disposition les données aux utilisateurs sous forme par exemple d'un serveur web, d'une base de données et d'une visualisation graphique.

Le flux habituel dans l'IoT est le flux Uplink (flux montant), c'est-à-dire l'émission de données des objets vers le serveur. Comme nous l'expliquerons plus loin, en LoRaWAN il est aussi possible de transférer des données aux Device LoRa par un flux Downlink (flux descendant).

notes



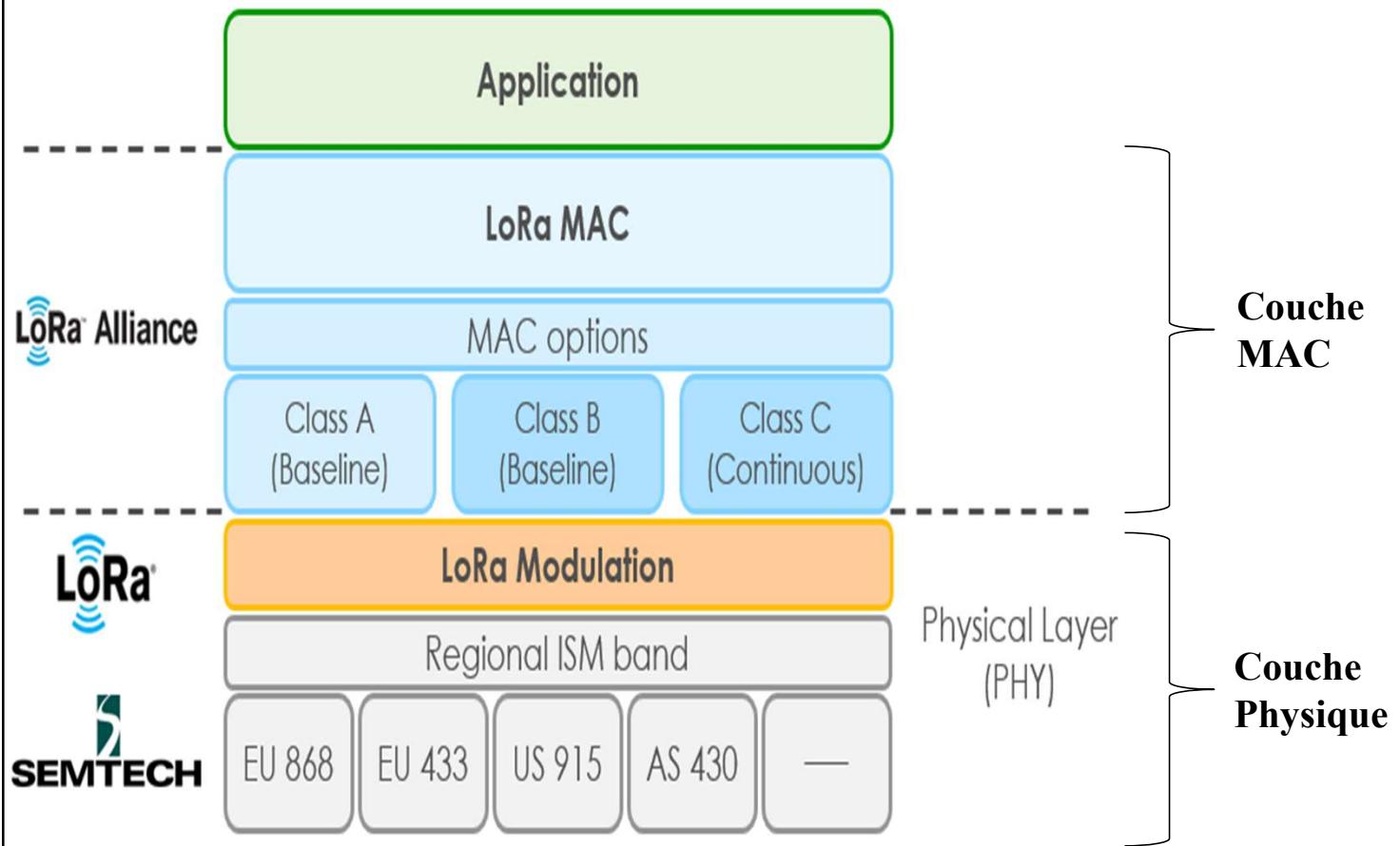
La Gateway reçoit d'un côté un message radio modulé en LoRa et transmet de l'autre côté une trame IP à destination du Network Server.

Coté interface Radio : La Gateway réceptionne une trame LoRaWAN et extrait le PHY Payload. La Gateway extrait aussi toutes les informations utiles sur les caractéristiques de la réception qui a eu lieu : SF, Bandwidth, RSSI, Time On Air...etc...

Coté interface réseau : La Gateway transmet l'ensemble de ces informations dans un paquet IP (UDP) au Network Server. Les données transmises sont du texte en format JSON. La Gateway a donc bien un rôle de passerelle entre le protocole LoRa d'un côté et un réseau IP de l'autre.

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

notes



La figure suivante présente l'architecture en couche de LoRaWAN en reprenant la terminologie du modèle OSI.

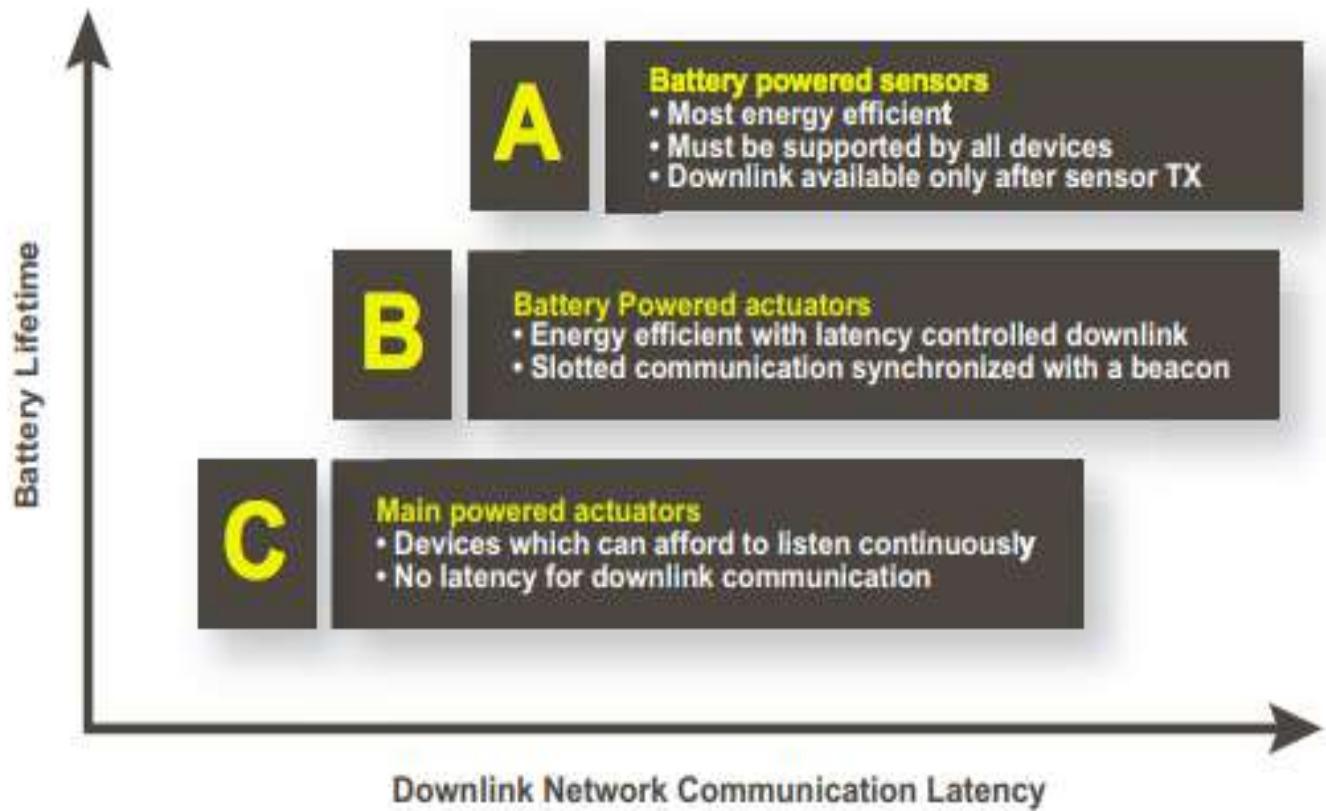
LoRa définit la couche physique (PHY), LoRaWAN définit la couche MAC (protocole de communication et architecture du réseau).

LoRaWAN est proposé par LoRa Alliance.

La couche MAC, donc le protocole LoRawan est open source. Les composants physiques sont sous License SEMTECH.

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

notes



Les Devices LoRa sont classés en 3 catégories (A, B, C) en fonction de leur consommation et de leur accessibilité en Downlink, c'est-à-dire la facilité qu'un utilisateur aura à transmettre une trame au Device LoRa.

Classe A (« All »):

Description: Le module écoute après chaque émission

Usage: Modules sans contraintes de latence de réception de messages

Conso: la plus économique énergétiquement. Supportée par l'ensemble des modules. Adapté aux modules sur batterie.

Classe B (« Beacon »):

Description: Le module écoute à une fréquence régulière paramétrable

Usage: Modules ayant une contrainte de latence de réception de message de quelques secondes

Conso: optimisée par rapport à l'application visée. Adaptée aux modules sur batterie

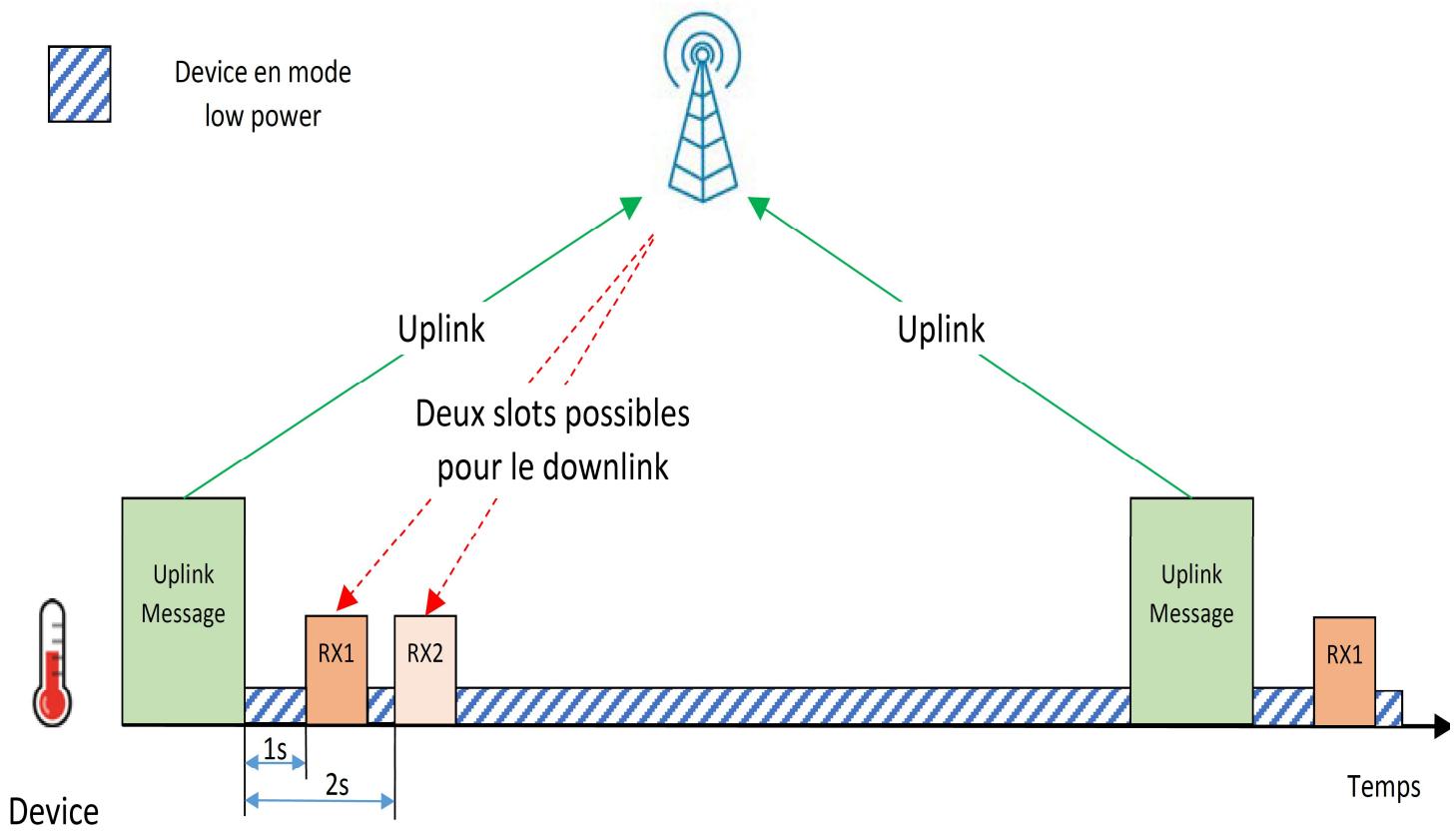
Classe C (« Continuous »):

Description: Le module écoute en permanence

Usage: Modules ayant une contrainte de latence de réception forte (moins d'une seconde).

Conso: adaptée aux modules sur secteur ou n'ayant pas de contrainte d'autonomie.

notes



Tous les Devices LoRaWAN sont de classe A. Chaque Device peut transmettre (Uplink) à la Gateway sans vérification de la disponibilité du récepteur. Si la transmission échoue, elle sera réémise après un certain temps. Cette transmission est suivie de 2 fenêtres de réception très courtes. La Gateway peut alors transmettre pendant le « RX Slot 1 » ou le « RX Slot 2 », mais pas les deux.

La durée des fenêtres doit être au minimum la durée de réception d'un préambule. Un préambule dure 12.25 Tsymbole et dépend donc du Data Rate. Lorsque qu'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 bien à destination du Device LoRa, alors la deuxième fenêtre n'est pas ouverte.

Première fenêtre de réception :

- Le Slot RX1 est programmé par défaut à 1 seconde +/- 20 µs après la fin de l'émission Uplink.
- La fréquence (canal) et le Data Rate (DR) sont les même que ceux choisis lors de la phase d'émission (Uplink).

Seconde fenêtre de réception :

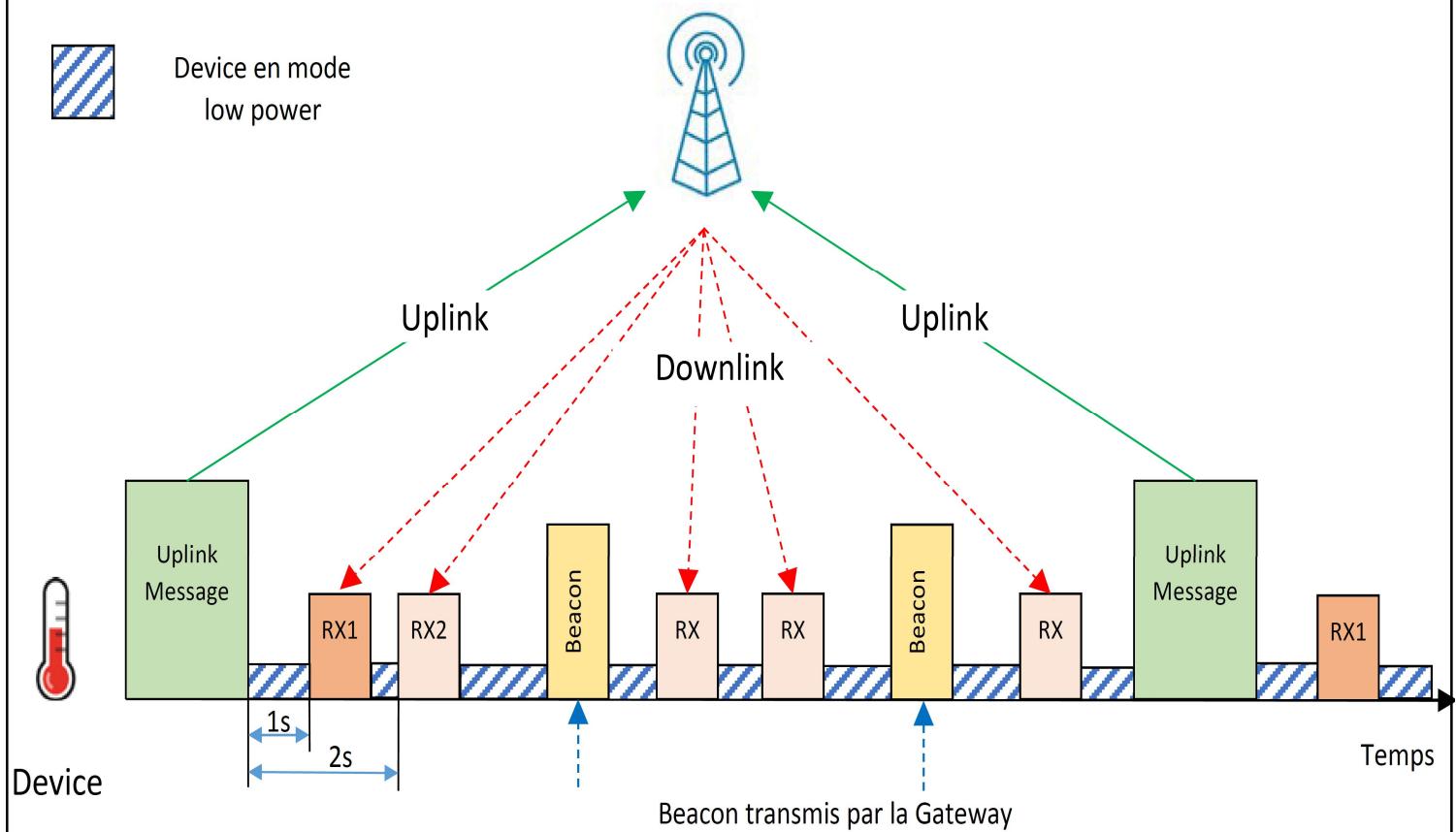
- Le Slot RX2 est programmé par défaut à 2 secondes +/- 20 µs après la fin de l'émission Uplink.
- La fréquence et le Data Rate (DR) sont configurables mais fixes.

Entre les slots le device est en sommeil (basse consommation)

Un Device LoRa qui est uniquement de classe A ne peut pas recevoir s'il n'a pas émis.

Il n'est donc pas joignable facilement.

notes

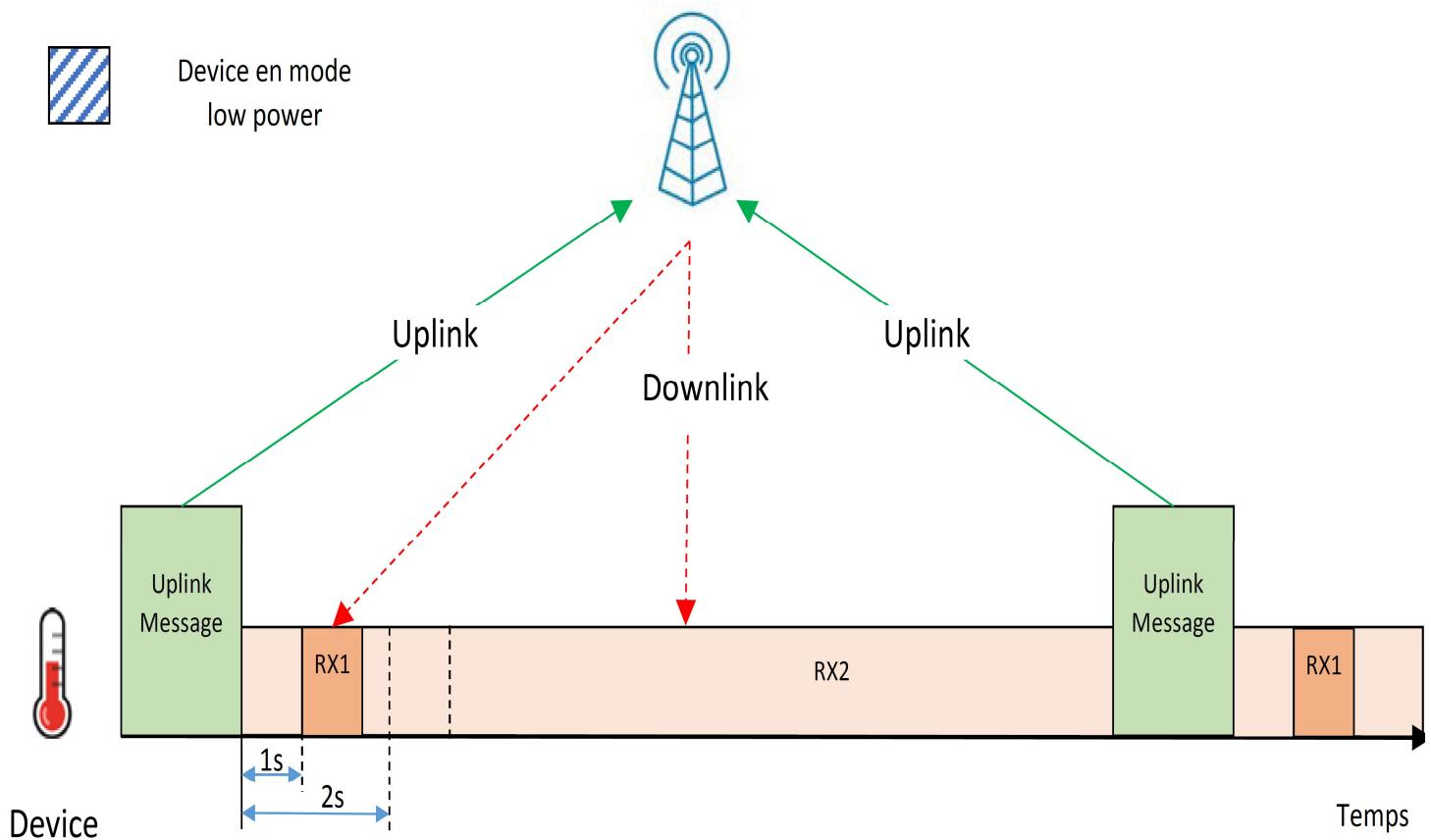


Tous les Devices démarrent et rejoignent le réseau en tant que Devices de classe A.

Les Devices de classe B ont le même comportement que les Devices de classe A, mais d'autres fenêtres de réceptions sont programmées à des périodes précises. Afin de synchroniser les fenêtres de réception du Device LoRa, la Gateway doit transmettre des balises (Beacons) de façon régulière, toutes les 2 minutes.

Un Device LoRa de classe B est joignable régulièrement sans qu'il soit nécessairement obligé d'émettre. En revanche, il consomme plus qu'un Device de classe A.

notes

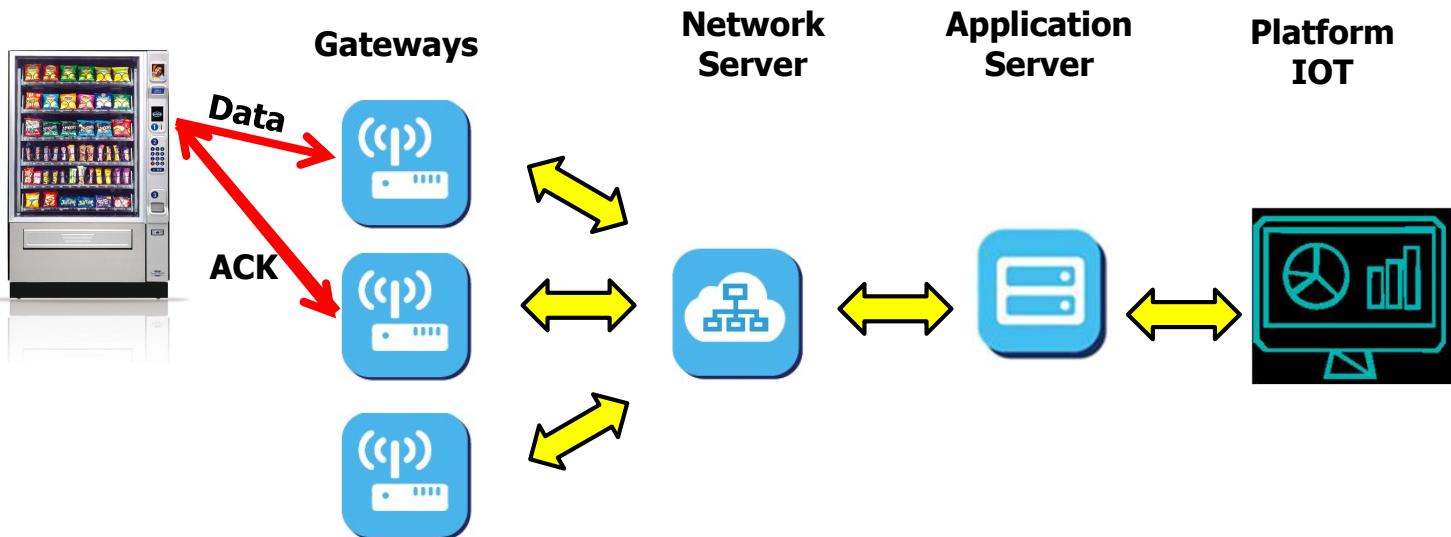


Les Devices de classe C ont des fenêtres de réception constamment ouvertes entre 2 Uplinks. Ces Devices consomment donc beaucoup plus.

Un Device LoRa de classe C est joignable en permanence. En revanche, c'est la classe de Device qui est la plus énergivore.

La Gateway utilisée pour le Downlink est celle qui a reçu le dernier message du Device LoRa. Un message ne pourra jamais arriver à destination d'un Device LoRa si celui-ci n'a jamais transmis, quel que soit sa classe A, B ou C.

notes



1. Le distributeur transmet la donnée
2. Les passerelles passent la donnée au Network serveur
3. Le Network serveur passe la donnée au serveur d'application distributeur
4. Le serveur de réseau sélectionne une passerelle
5. Finalement la passerelle transmet l'acquittement au end-node

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.

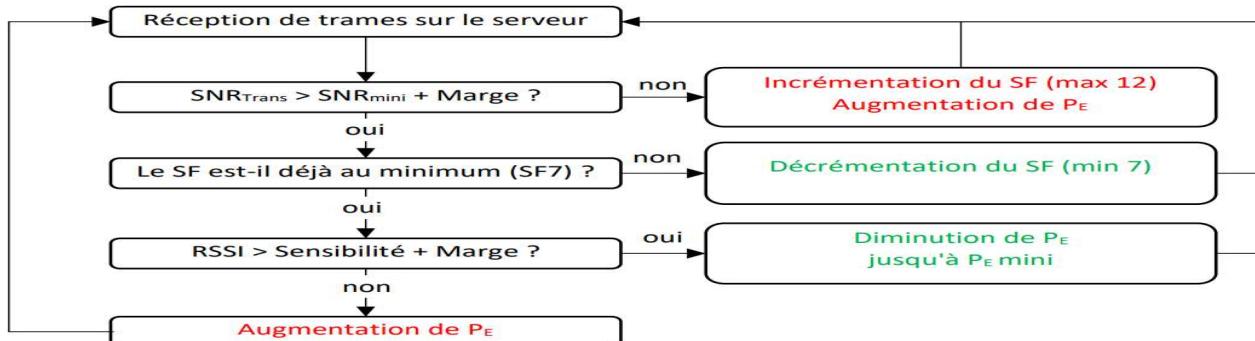
Le serveur réseau sélectionne le meilleur chemin, la meilleure passerelle pour l'acquittement au device.

Le serveur réseau a une visibilité totale sur la qualité de la liaison montante, y compris le RSSI et le SNR du paquet reçu, ainsi que sur le duty-cycle utilisé par toutes ses passerelles. Par conséquent, il peut prendre une décision éclairée sur la passerelle à utiliser pour la transmission de paquets en liaison descendante.

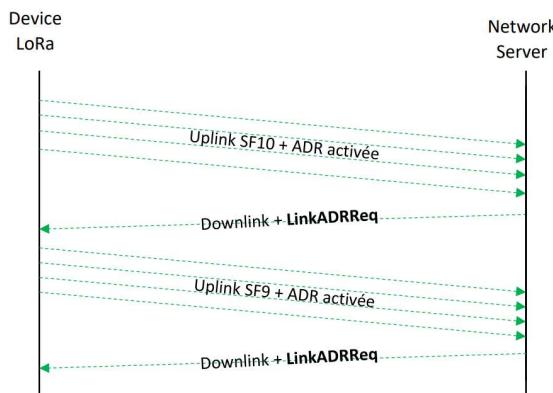
notes



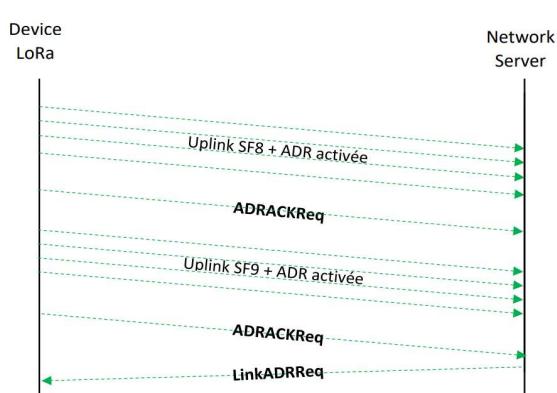
□ ALGO ADR



□ Scénario avec downlink



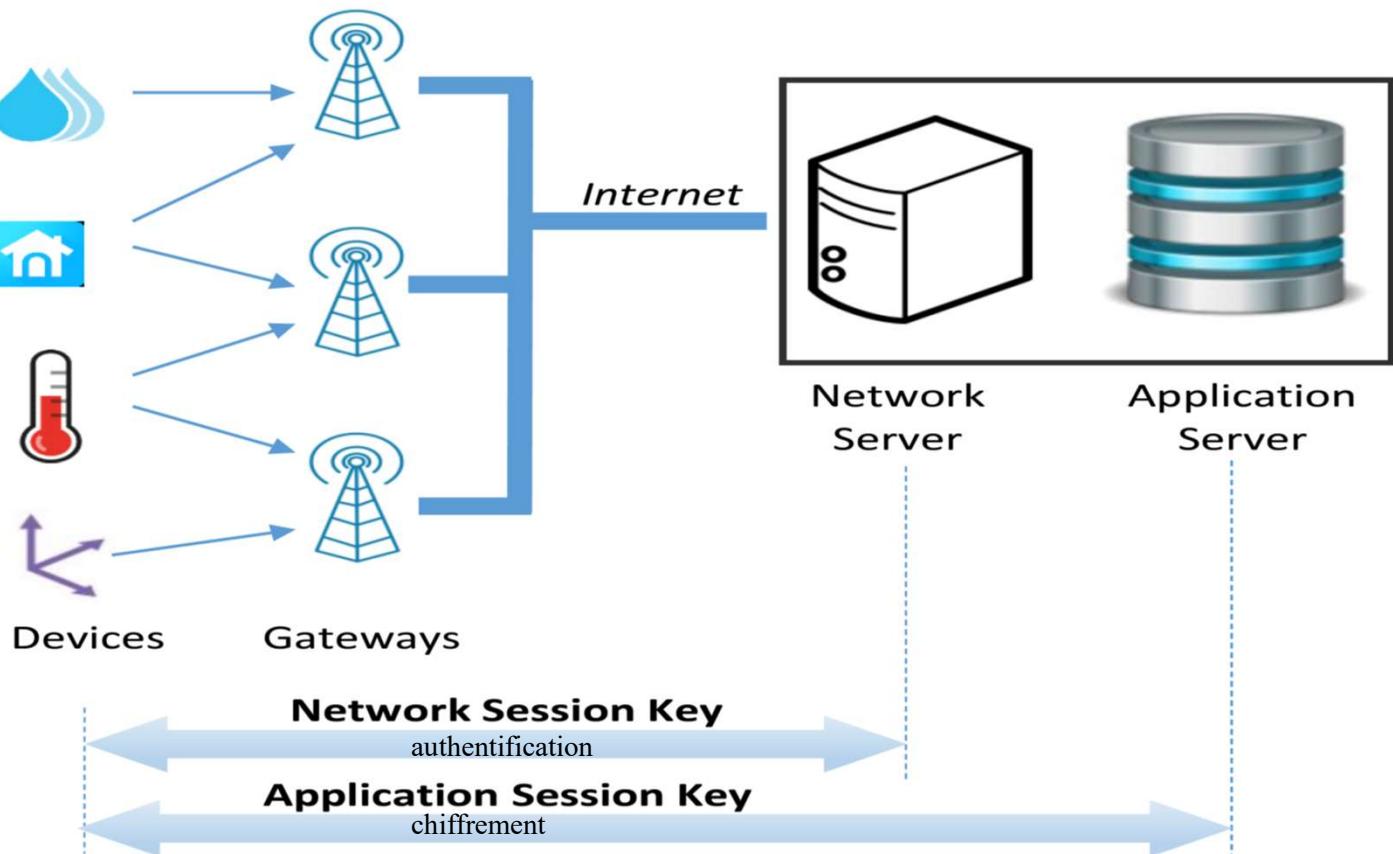
□ Scénario sans downlink



Le réglage du SF et du PE (puissance émise) 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 / PE.

- L'Adaptive Data Rate n'est actif que si le Device le demande explicitement.
- On peut choisir le SF et la PE par défaut pour les premiers envois de trames.
- La trame linkADRReq n'est pas envoyée seule. Elle est mutualisée avec une trame de données en Downlink. S'il n'y a pas de trame de données Downlink disponible alors le Network Serveur attendra avant d'envoyer cette commande.
- En utilisant des trames de données Uplink "Confirmed" (c'est-à-dire en demandant un acquittement au Network Server), on génère un flux Downlink qui facilite l'envoi de la commande linkADRReq. Le Device LoRa converge donc plus vite vers la valeur optimale.
- Lorsque le Device n'a pas reçu de trame Downlink, il sait que le Network Server n'a pas eu d'opportunité de lui transmettre sa commande linkADRReq. Si 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**.
 - . S'il n'obtient pas de réponse dans un temps imparti, alors la communication avec le serveur est considérée comme perdue et le Device LoRa va incrémenter SF / PE, jusqu'à rétablir la communication avec le serveur.

notes



Les informations transmises au Network Server depuis les Devices LoRa sont authentifiés grâce à une clé AES 128 bits appelée **Network Session Key** : **NwkSKey**. Nous parlons bien ici d'authentification, et non pas de chiffrement comme nous le verrons plus tard.

- 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

Les messages transmis à l'application server sont chiffrés grâce à une clé AES 128 bits appelée Application Session Key : **AppSKey**.

L'Application Server est souvent sur le même support physique que le Network Server. Il permet de dissocier les applications les unes des autres. Chaque application enregistre des Devices LoRa qui auront le droit de stocker leurs données (Frame Payload).

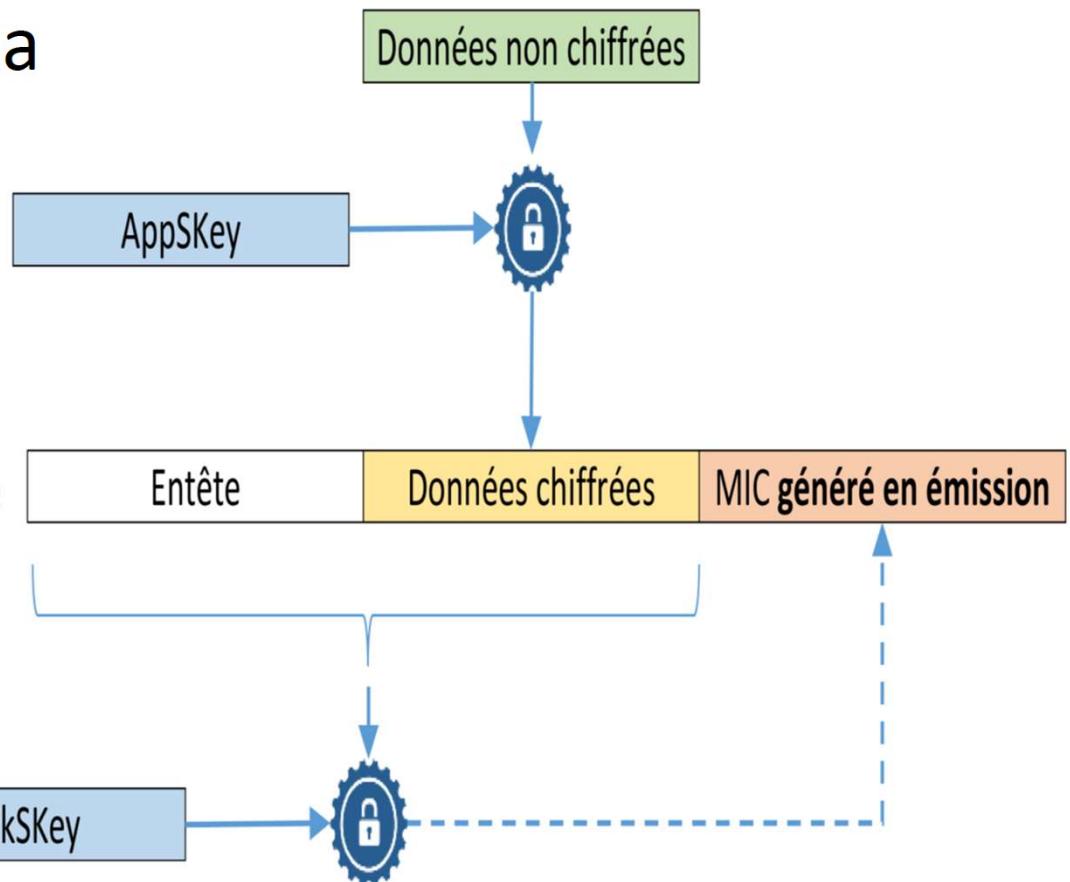
notes



Device LoRa



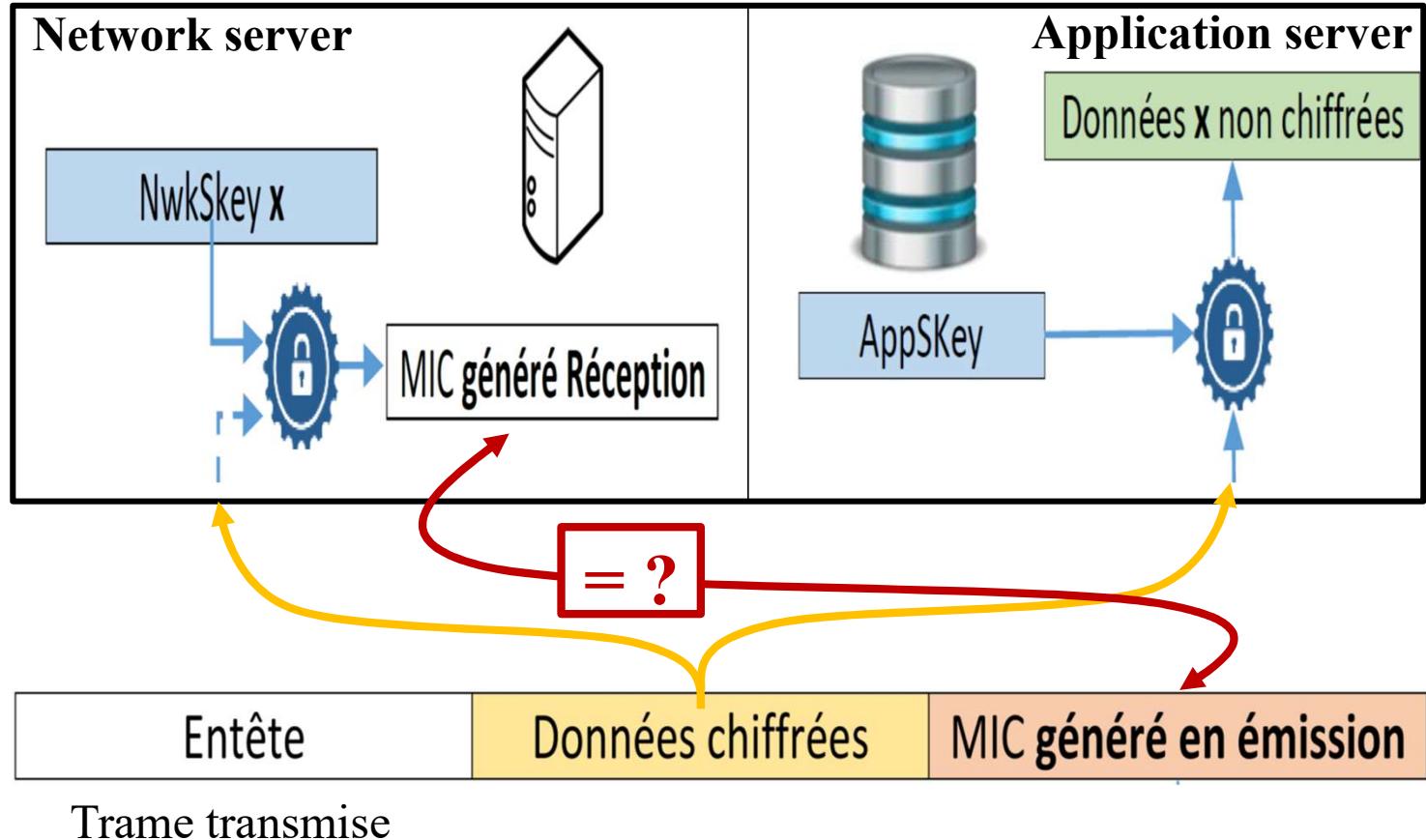
Trame transmise =



La première étape est le chiffrement de la donnée issu du capteur via une première clé (Application Session Key) . Ce chiffrement sera décrypté uniquement par le serveur de l'application.

Ensuite à la trame est ajouté une sorte de checksum, appellé Message Integrity Control généré via une autre clé (Network Session Key) . Ce checksum, cette signature, sera vérifié par le serveur réseau pour authentifier le message.

notes



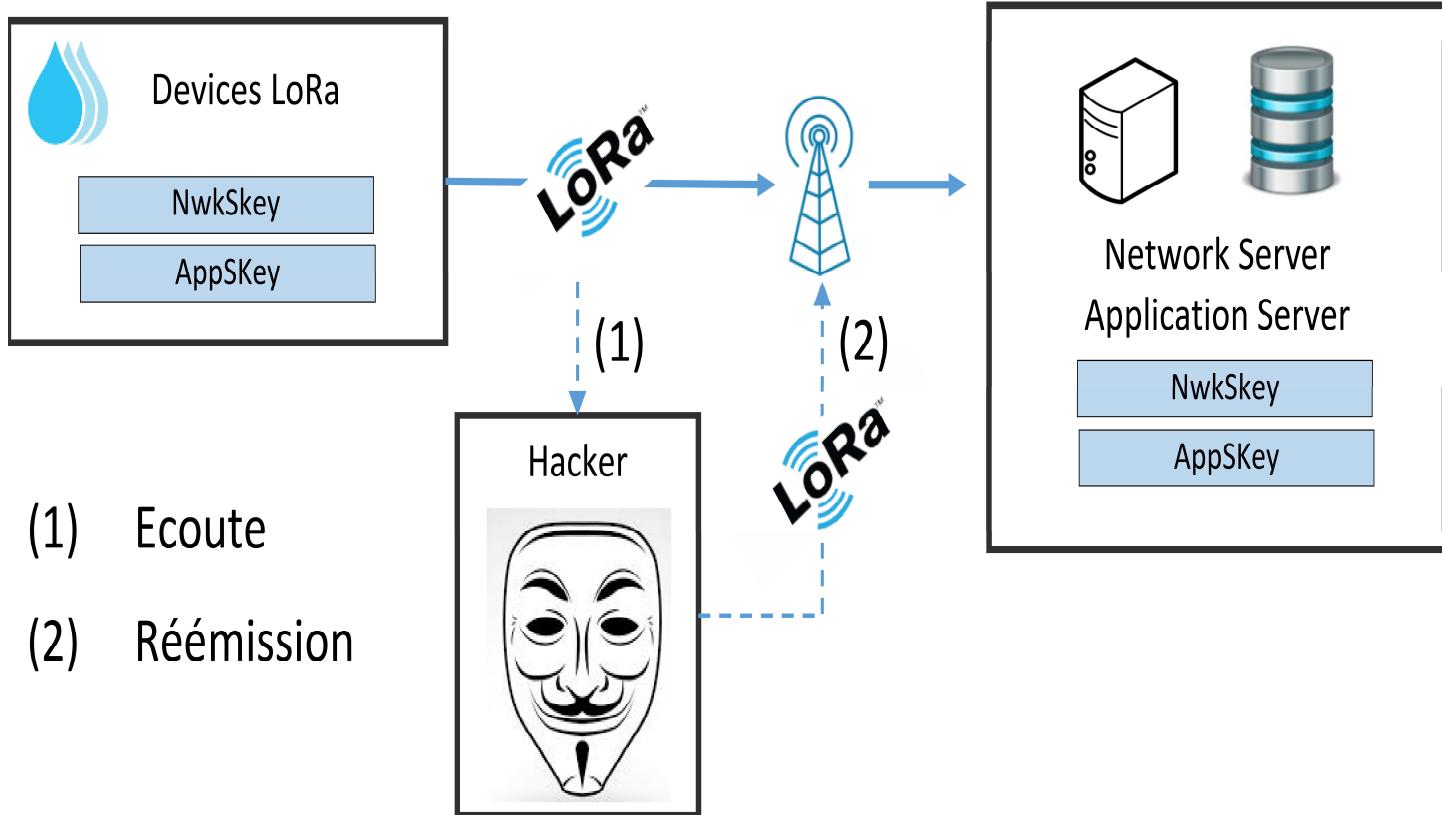
L'Application Session Key (AppSKey) sert pour le chiffrement entre le Device LoRa et l'Application Server. Les données chiffrées seront alors décodées à la réception sur l'Application Server s'il possède la même clé.

La Network Session Key (NwkSKey) sert à l'authentification entre le Device LoRa et le Network Server. Afin de réaliser cette authentification entre le Device et le Network Server, un champ MIC (Message Integrity Control) est rajouté à la trame. Il est calculé en fonction des données transmises et du NwkSkey. A la réception, le même calcul est effectué. Si les clés sont équivalentes dans le Device et dans le Network Server alors les deux MIC calculés doivent correspondre.

Conclusion: le device et les serveurs doivent partager 2 clés: NwSKey et AppSKey. Deux méthodes sont possibles pour fournir ces informations à la fois au Device LoRa et au serveur.

- Activation By Personalization : APB
- Over The Air Activation

notes



SOLUTION: *FRAME COUNTER*

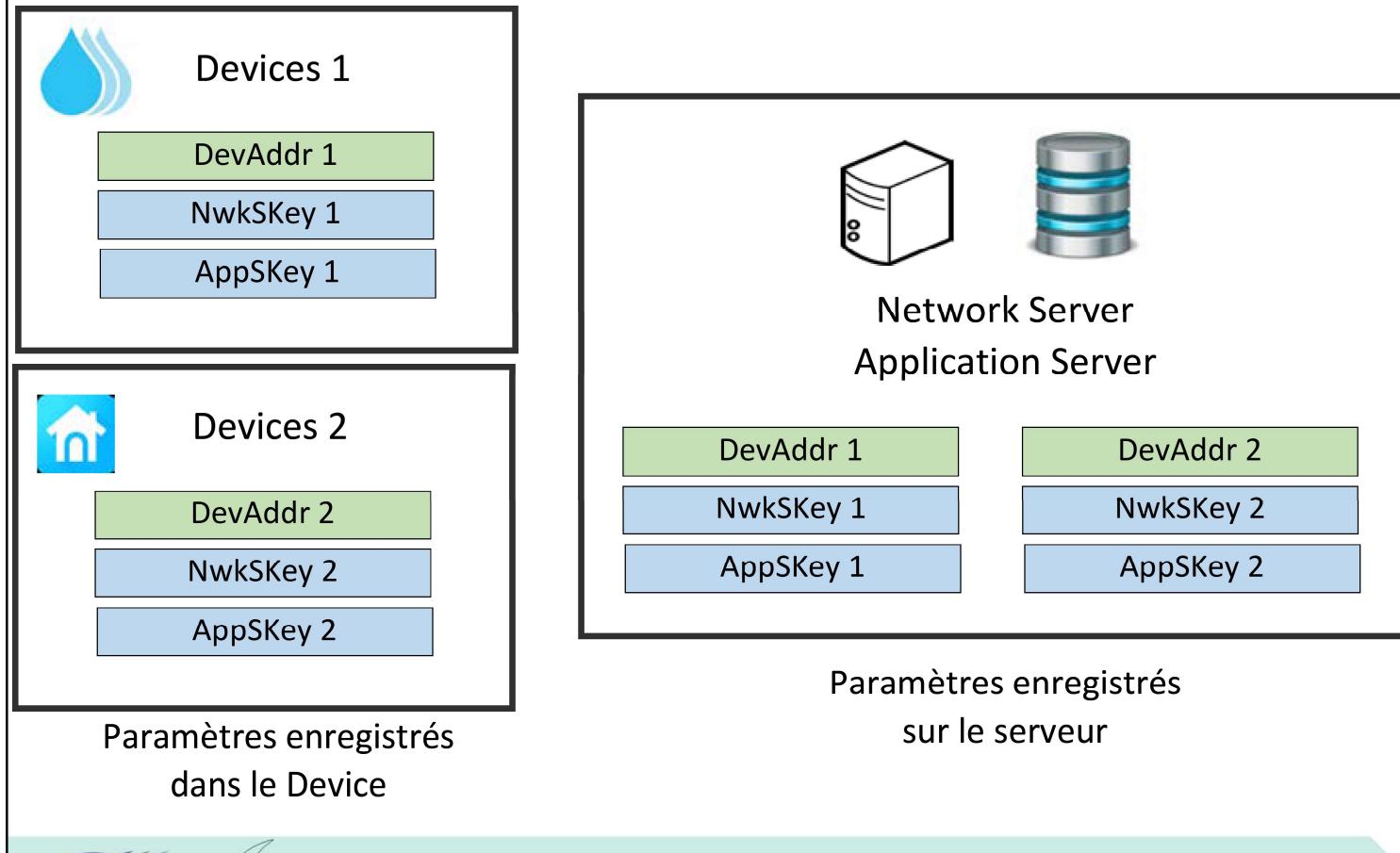
Les clés AES 128 bits permettent le chiffrement des informations transmises en LoRaWAN. Malgré ce chiffrement, une attaque connue en Wireless est celle du REPLAY. C'est-à-dire que le Hacker enregistre des trames chiffrées circulant sur le réseau LoRa pour les réémettre plus tard. Même si le Hacker ne comprend pas le contenu (les trames sont chiffrées), les données qui sont transportées sont, elles, bien comprises par l'Application Server. Des actions peuvent donc être réalisées simplement par mimétisme.

Pour éviter cela, la trame LoRaWAN intègre un champ variable appelé « Frame Counter ». Il s'agit d'un nombre numérotant la trame sur 16 ou 32 bits. L'Application Server acceptera une trame uniquement si le « Frame Counter » reçu est supérieur au « Frame Counter » précédemment. Donc si le Hacker, retransmet la trame telle qu'elle, le « Frame Counter » sera équivalent, donc la trame sera refusée. Et comme la trame est chiffrée le Hacker n'a pas moyen de savoir comment modifier un champ de cette trame.

Cette sécurité implémentée par les « Frame Counter » est intéressante pour s'affranchir d'une attaque par REPLAY, mais dans nos tests cela peut poser problème. En effet, à chaque fois que nous redémarrons le microcontrôleur, notre « Frame Counter » revient à zéro, alors que celui de l'Application Server continue de s'incrémenter. Cela peut être résolue par différentes manières :

1. Désactivation du « Frame Counter Check » : Dans certain Application Server, cette option peut être proposée. Bien sûr, il faudra garder en tête la faille de sécurité que cela engendre.
2. Utiliser l'authentification par OTAA au lieu de ABP. En effet, à chaque ouverture de session en OTAA, le « Frame Counter » est réinitialisé coté serveur.
3. Conserver la valeur du « Frame Counter » dans une mémoire non volatile et récupérer sa valeur au démarrage du Device LoRa.

notes



En LoRaWAN, les trois éléments indispensables pour la communication sont le **DevAddr** (32 bit) pour l'indentification du Device, ainsi que deux clés : le **NwkSKey** pour l'authentification et l'**AppSKey** pour le chiffrement. Deux méthodes sont possibles pour fournir ces informations à la fois au Device LoRa et au serveur.

ABP : Activation By Personalization

C'est la plus simple des méthodes. C'est donc peut être celle que nous avons tendance à utiliser lors d'un test de prototype et d'une mise en place de communication LoRaWAN.

En ABP, toutes les informations nécessaires à la communication sont déjà connues par le Device LoRa et par le Serveur.

Avantage : Raccordement au réseau simplifié ; l'objet est rapidement opérationnel.

Inconvénient : Les clés de chiffrement permettant la communication avec le réseau, sont préconfigurées dans l'objet : sécurité affaiblie.

ATTENTION : Cette stratégie de simplicité dans la procédure de jonction se fait au détriment de la sécurité . S'il y a intrusion physique dans l'objet, le vol des clés est possible et peut conduire à l'usurpation de l'identité de l'objet, entraînant une corruption des données collectées.

notes



DevEUI

AppEUI

AppKey

Join Request



DevAddr

NwkSKey

AppSKey

Device/Server
Avant activation

Over the Air «OTAA»

Device/Server
Après activation

DevEUI:	O.U.I 24bits	Serial number 40bits
AppEUI:	O.U.I 24bits	Serial number 40bits

OTAA : Over The Air Activation :

C'est la méthode qu'il faut privilégier car c'est celle qui est la plus sécurisée. Le Device LoRa doit connaître : le **DevEUI**, l'**AppEUI**, et l'**Appkey**. Le Network Server doit lui connaître la même **Appkey**.

Tous les éléments notés EUI (Extended Unique Identifier) sont toujours sur une taille de 64 bits.

- DevEUI : Identificateur unique de périphérique 64 bits. Équivalent à l'adresse MAC d'une carte réseau. Cette adresse est inscrite dans l'appareil par le fabricant en même temps que le firmware. Il se compose d'un identificateur unique de 24 bits (*) et d'un numéro de série de 40 bits. Le code OUI (Organizationally Unique Identifier) est unique pour n'importe quel fabricant et fourni par l'IEEE et le code série est librement défini par le fabricant.

- AppEUI : Ce numéro identifie de façon univoque un Join Server

Il se compose d'un identificateur unique de 24 bits (*) et d'un numéro de série de 40 bits. Le code OUI est le code du fabricant de l'entité exploitant le serveur de jointure. Le numéro de série peut être utilisé si plusieurs Join Server sont exploités par la même entité.

- AppKey : AES 128 clé utilisée pour générer le MIC (Message Code Integrity) lors de la Join Request. Il est partagé avec le Network server.

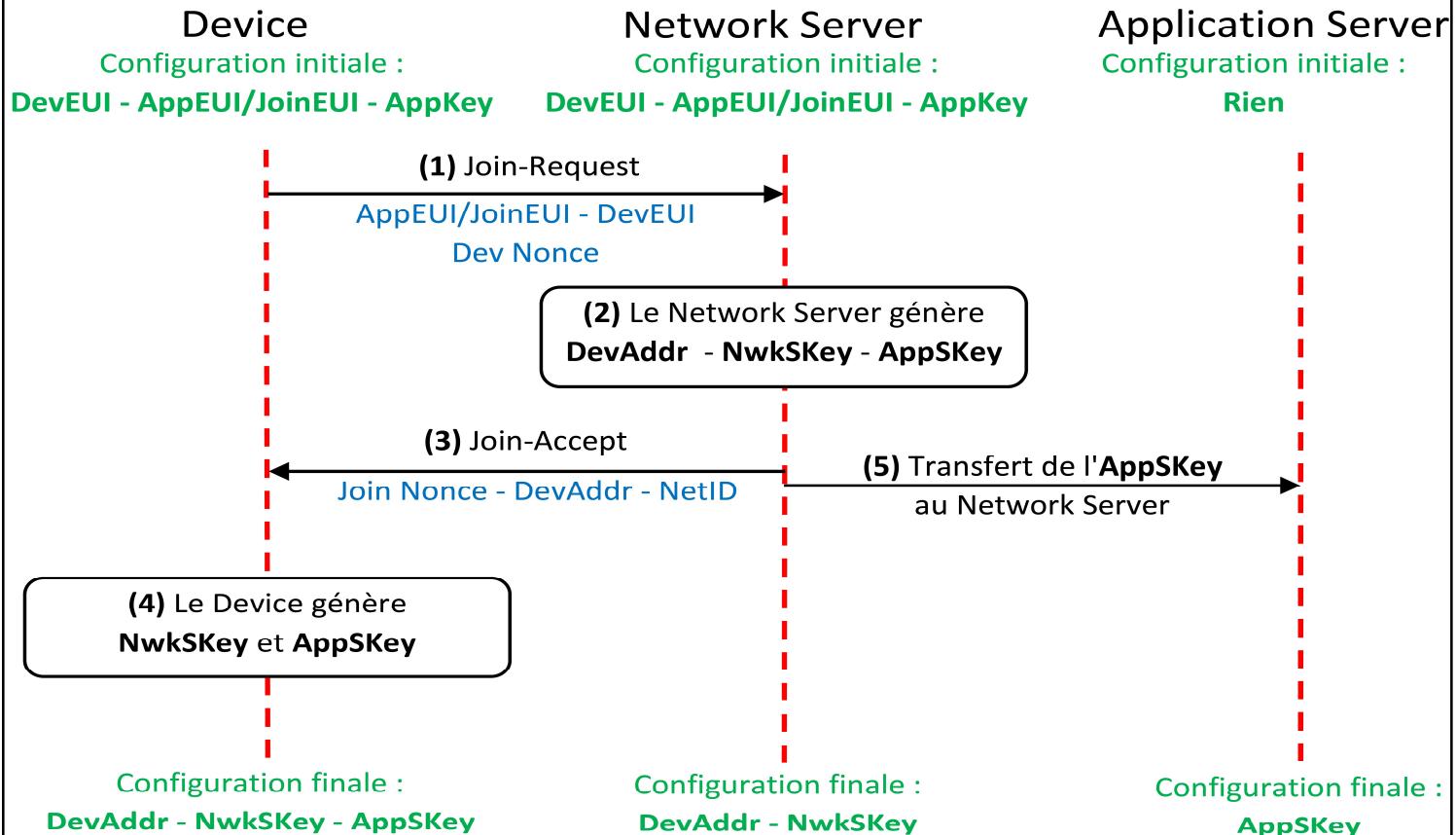
- NwkSKey : Utilisé pour l'authentification avec le Network Server

- AppSKey : Utilisé pour le chiffrement des données

Grâce à ces informations de départ et une négociation avec le Network Server (Join Request), le Device LoRa et le serveur vont générer les informations essentielles :

DevAddr, NwkSKey et AppSKey.

notes



Pour activer un équipement sur le réseau par la méthode OTAA, l'équipement doit transmettre au réseau une demande d'accès : *join request*. Pour ce faire, celui-ci doit être en possession de trois paramètres :

Le DevEUI, identifiant unique de l'équipement (fourni par l'équipementier).

AppEUI, identifiant du fournisseur de l'application.

AppKey, clé AES 128 déterminée par le fournisseur de l'application.

L'équipement envoie, à travers le réseau, la requête *join request*, contenant DevEUI, AppEUI ainsi qu'un MIC calculé via la clé AppKey. Cette requête est transmise au serveur d'enregistrement qui vérifie le MIC via la clé AppKey (qui lui a été communiquée au préalable). Si l'équipement est autorisé par le serveur d'enregistrement, la requête *join accept* est transmise en réponse à l'équipement.

Cette réponse contient des données à partir desquelles l'équipement va pouvoir calculer les clés de session (réseau et applicative). Parmi les données contenues dans cette réponse, se trouve également l'adresse – Device Adress (*DevAddr*) sur 32 bits – qu'utilisera l'équipement pour communiquer sur le réseau.

A chaque nouvelle session, les clés de session sont renouvelées.

Avantage : Le réseau génère et envoie les clés de chiffrement ; la sécurité est renforcée.

C'est la méthode la plus utilisée dans le monde de l'IoT/LoRaWAN, car la plus sécurisée.

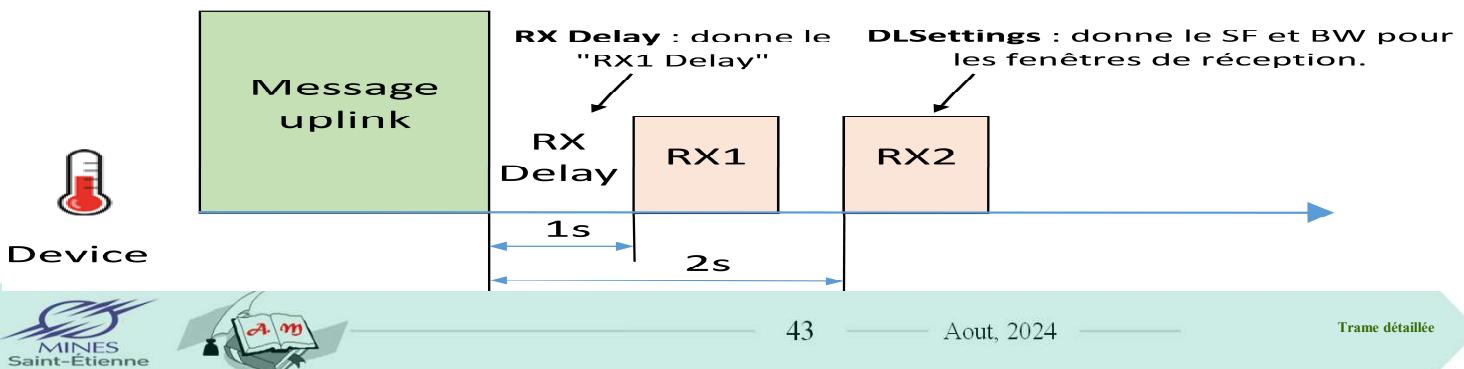
Inconvénient : L'objet doit implémenter ce mécanisme de jonction ce qui introduit une complexité supplémentaire.

notes



	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	Compliqué 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

CFList : information sur les canaux disponibles pour les messages uplink



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

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

Elles doivent donc être stockées dans des mémoires hautement sécurisées

Cependant, 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. 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.

le Join-Accept comprend :

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.

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.

Le champ 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.

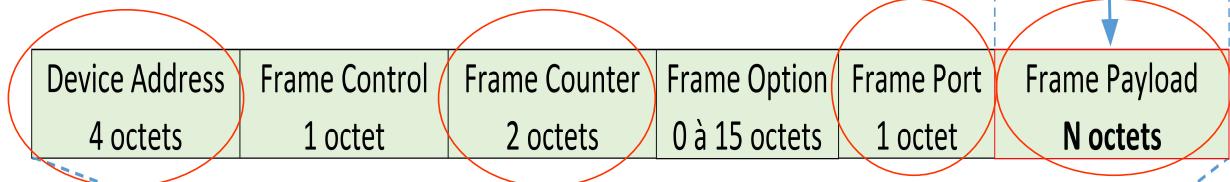
Notes



Chiffrement des données utilisateur par l'AppSkey



Couche Application



Couche Mac



Couche Physique



Couche application

La couche Application accueille les données de l'utilisateur. Avant de les encapsuler, elles sont chiffrées avec l'AppSKey afin de sécuriser la transaction.

Un ensemble de champs nommé **Frame Header** permet de spécifier le **DevAddr**, le **Frame Control**, le **Frame Counter**, et le **Frame Option**.

Le **Frame Port** dépend du type d'application et sera choisi par l'utilisateur.

Le **Frame Payload** sont les données chiffrées à transmettre. Le nombre d'octets maximum N pouvant être transmis varie selon le Data Rate, de 51 octets (DR0) à 222 octets (DR6).

Couche MAC

Cette trame est destinée au Network Server. Elle est authentifiée grâce au champ MIC (Message Integrity Protocol) selon la méthode expliquée précédemment

Le protocole LoRa MAC est composé de :

1. MAC Header : Version de protocole et type de message
2. MAC Payload : Contient tout le protocole applicatif.
3. MIC : Message Integrity Code, pour l'authentification de la trame.

Couche Physique

Déjà vu précédemment

notes



- Quel est l'avantage principal de la modulation LoRa (chirp spread spectrum) utilisée dans LoRaWAN ?
- Quel rôle joue une passerelle (Gateway) dans un réseau LoRaWAN ?
- Quel spectre de fréquence est principalement utilisé par LoRaWAN en Europe ?
- Quel paramètre peut être ajusté pour équilibrer la portée et la consommation d'énergie dans un réseau LoRaWAN ?
- Quelle méthode d'activation est recommandée pour les dispositifs LoRaWAN nécessitant une haute sécurité et une flexibilité de déploiement ?
- Dans une trame de données uplink, que signifie le bit ADR dans le champ FCtrl ?
- Quelle classe de dispositifs LoRaWAN utilise des créneaux de réception planifiés et des balises pour synchroniser les transmissions ?

notes





P2

Introduction*Organisation - définition - demain*

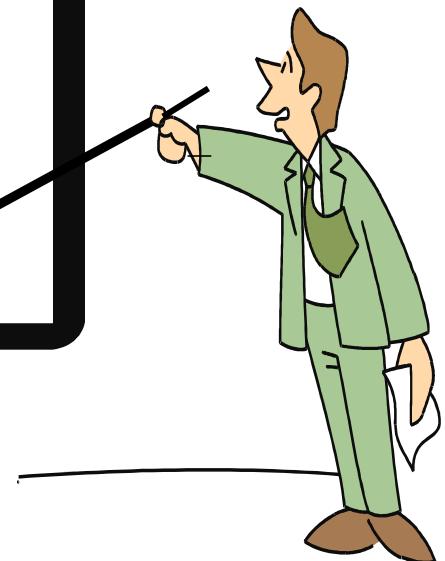
P12

LORA*Etalement de spectre – modulation LoRa*

P24

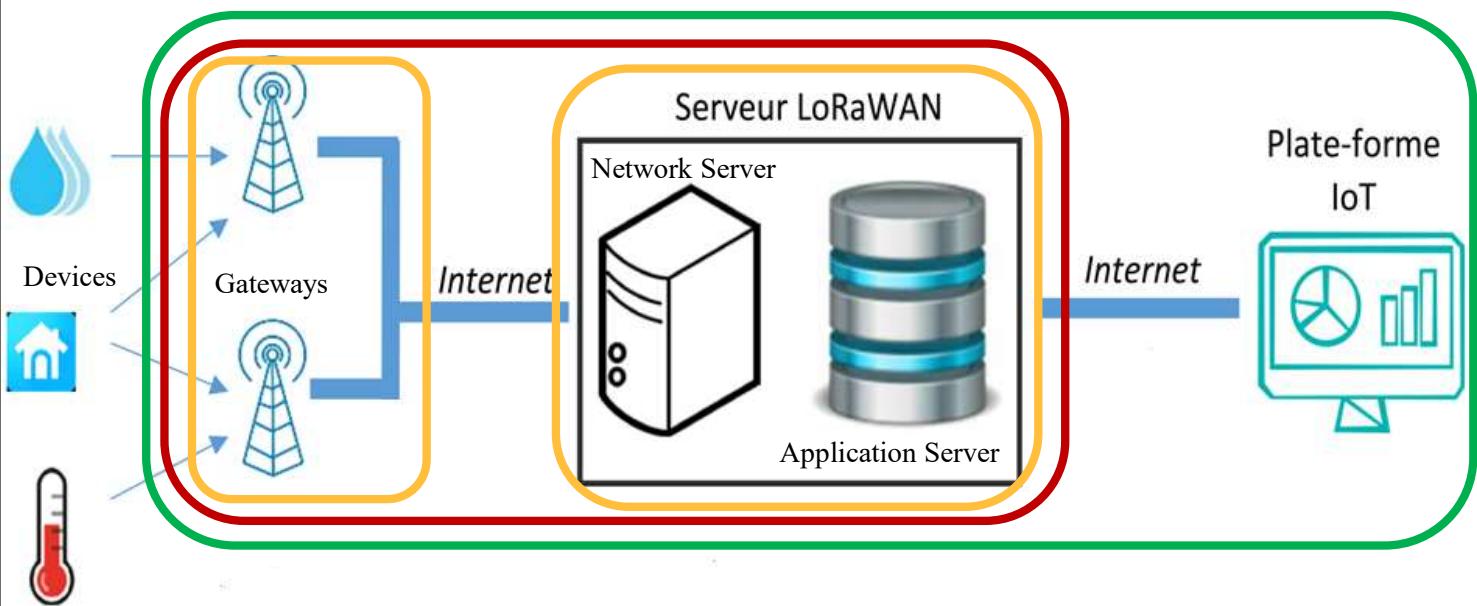
LORAWAN*LORAWAN : architecture – classes – sécurité - activation*

P44

Application*réseau – objet – passerelle – TTN - Plateforme IOT*

notes





- **LoRaWAN opérés**
- **LoRaWAN privés**
- **LoRaWAN Hybrides**

Les réseaux LoRaWAN opérés

Ce sont des réseaux LoRaWAN proposés par les opérateurs. Par exemple Objenious (filiale de Bouygues) ou encore Orange, KPN, Proximus. L'utilisateur a juste besoin de s'occuper des Devices LoRa et de l'application utilisateur (plateforme IOT). Les Gateways, le Network Server et l'Application Server sont gérés par l'opérateur.

Exemple: 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).

Les réseaux LoRaWAN privés

Chacun est libre de réaliser son propre réseau privé en implémentant sa propre Gateway pour communiquer avec ses Devices LoRa. L'implémentation du Network Server et de l'Application Server peuvent alors être envisagés de différentes façons.

Dans certaines Gateways, une implémentation de ces deux serveurs est déjà proposée (Kerlink: Wanesy SPN). Sinon il est possible de les mettre en place en achetant le logiciel (Actility, ResIOT). Il existe également des serveurs (Network et Application) open source (The Things Stack, Chirpstack).

notes



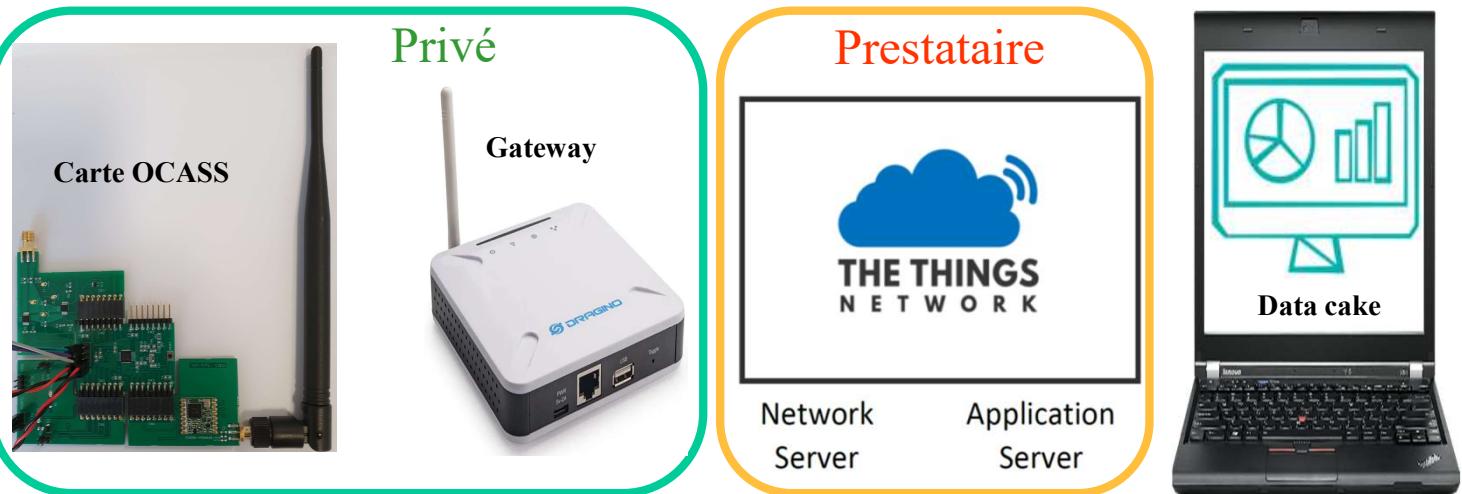
	Réseau privé	Réseau exploité
Coût de l'abonnement	Pas d'abonnement	Environ 1,5 € / mois par Device LoRaWAN
Coûts d'infrastructure	Investissement important au début (Gateways et Serveurs)	Inclus dans l'abonnement
Compétences requises	Requiert des compétences pour l'installation, l'administration et la 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

Les réseaux LoRaWAN Hybrides

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. Vous devrez acheter un hébergement cloud pour le serveur LoRaWAN (Actility, TTN, Loriot, ResIOT). Sur chaque serveur LoRaWAN hébergé sur un cloud, il y a très souvent une inscription gratuite mais limitée à quelques Gateways et quelques Devices.

notes

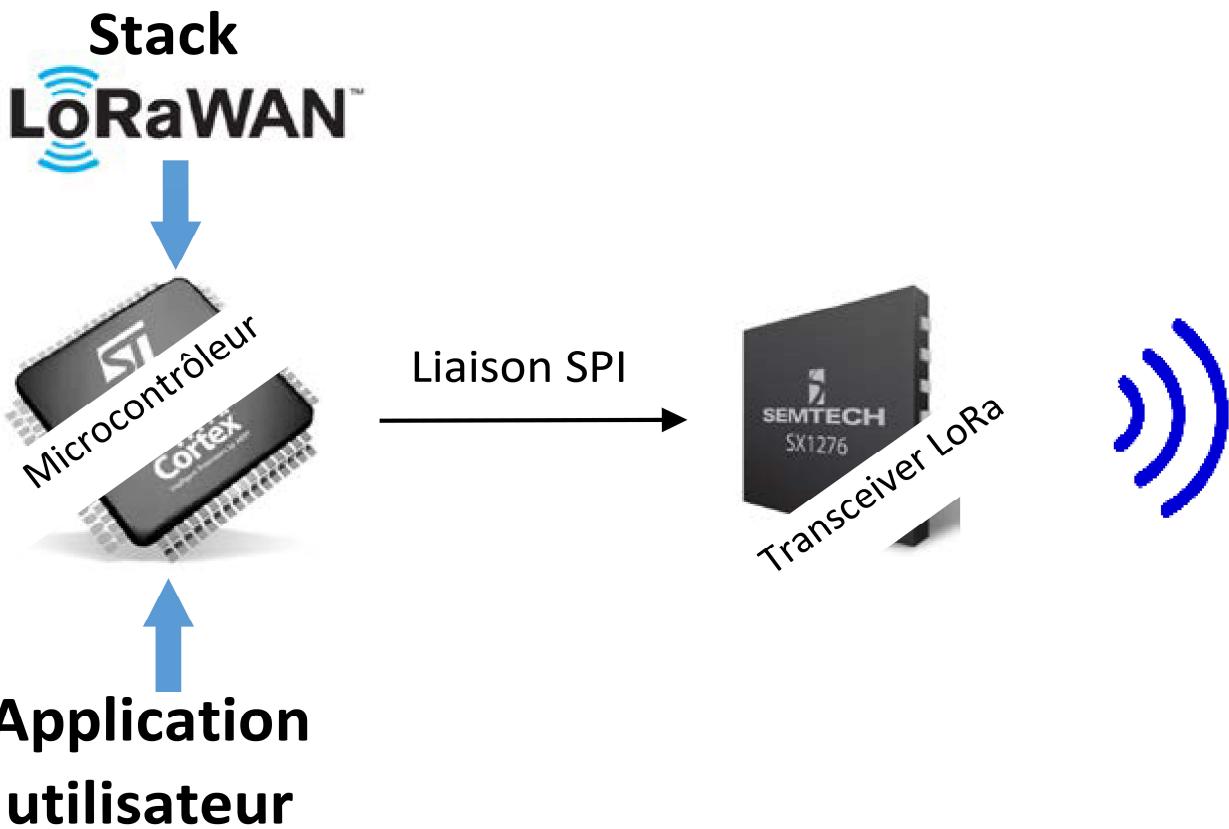


C'est une solution hybride avec nos propres Gateways et le serveur LoRaWAN TTN hébergé sur le cloud (version gratuite) que l'on utilisera dans notre application.

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.

notes



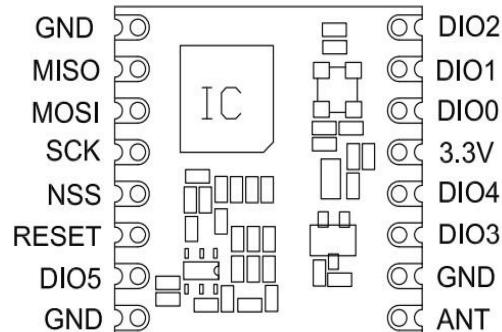
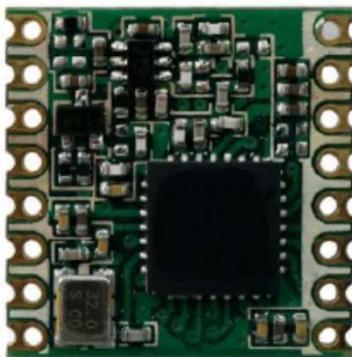
L'architecture choisi pour le device est du type "Microcontrôleur + Transceiver"

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 si possible open source. Le RFM95W est le Transceiver LoRa choisi . 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 SX1276) avec un bus SPI.

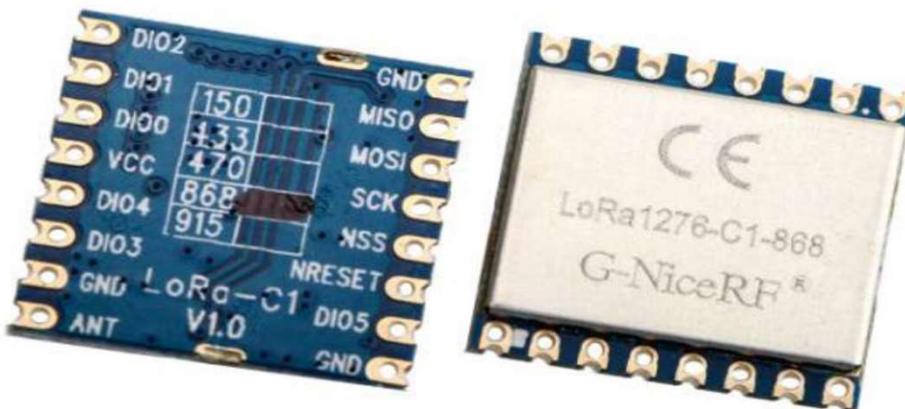
Ce choix est plutôt intéressant et optimisé en termes de consommation électrique, car il n'y a qu'un seul microcontrôleur qui gère tout. En revanche, c'est une solution plus complexe car il est nécessaire de se plonger dans la Stack LoRaWAN. En effet, même si l'application utilisateur et la Stack sont 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.

notes



RFM95W



NiceRF

Une solution intéressante est d'utiliser des "breakout boards", qui contiennent déjà le Transceiver et quelques composants utiles comme le RFM95W ou le NiceRF.

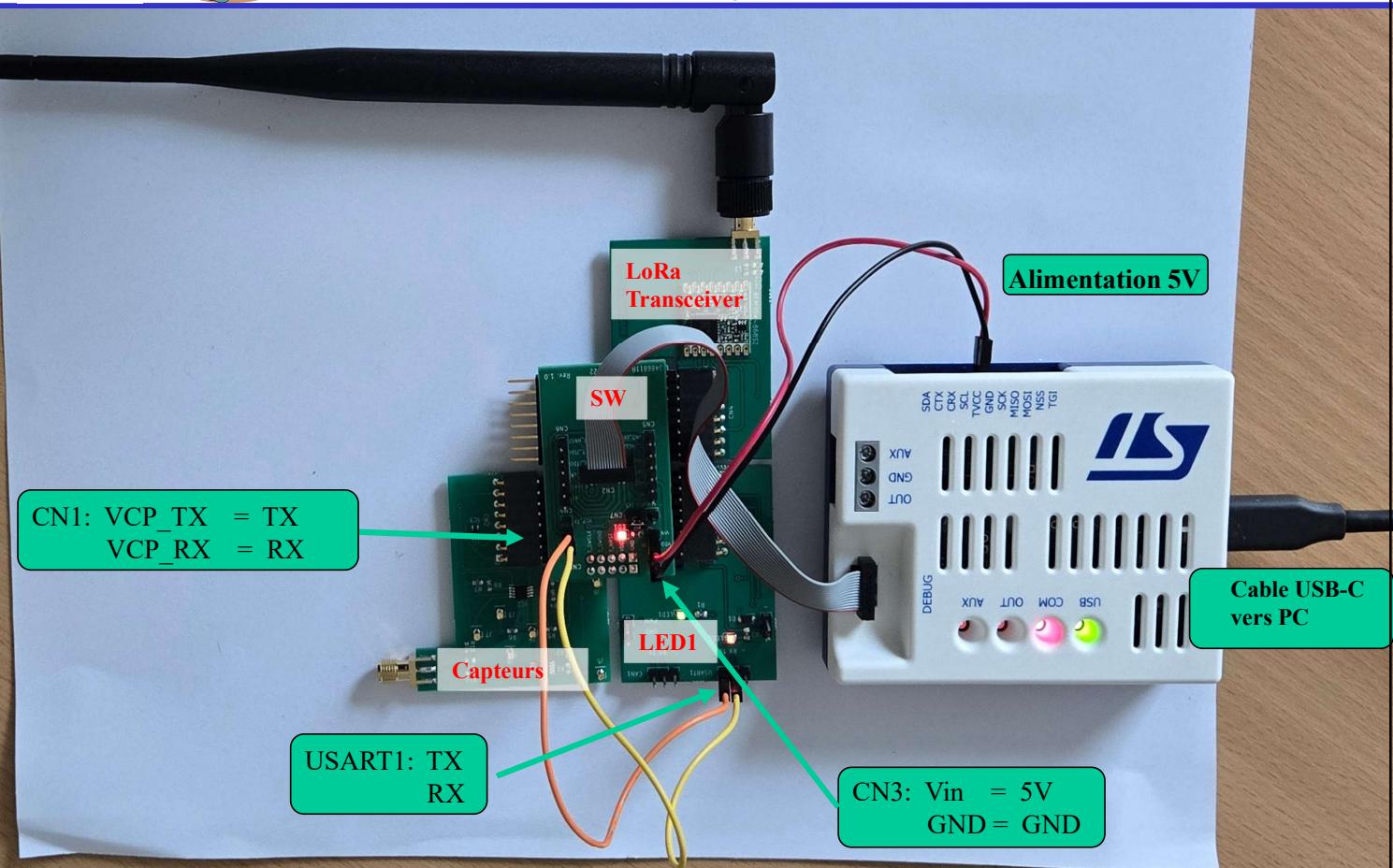
Ces modules sont assez répandus dans le milieu du prototypage d'objets connectés en LORA.

Ils sont de taille assez réduite 16mm x 16mm x 4mm et très économique avec un prix avoisinant les 5€.

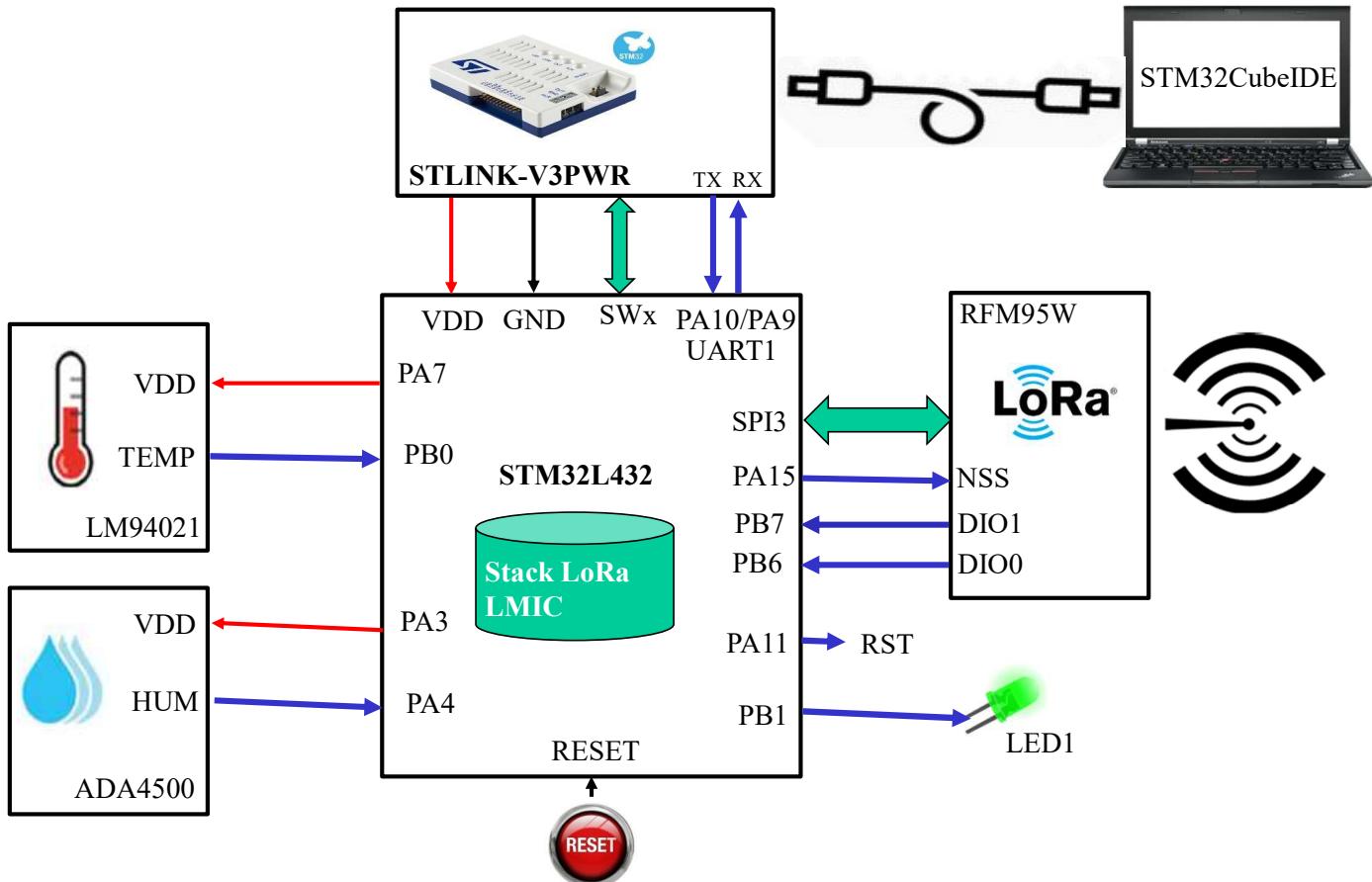
Parmi les caractéristiques principales de ce module nous pouvons noter :

- Embarque un module radio SX1276 LoRa® avec interface SPI
- Plage de fréquences RF : bande ISM 433 / 868 / 915 MHz
- Bande passante du canal : 125 kHz, 250 kHz ou 500 kHz
- Puissance de sortie RF : jusqu'à +20 dBm
- Sensibilité du récepteur : -119 dBm @ 1,2 kb/s (FSK), -148 dBm @ 18 b/s (LoRa, SF=12, CR=4/6)
- Portée jusqu'à 15 km (sans obstacles)
- RSSI à plage dynamique de 127 dB
- Capteur de température et indicateur de charge de batterie faible
- Alimentation : +1,8 V à +3,6 V c.c.
- Indice de température de fonctionnement : -20 à +70 °C

notes



notes



Comme notre device n'intègre pas déjà une Stack LoraWan, nous devons l'intégrer nous-mêmes. Voici une brève description des Stacks LoRaWAN disponibles :

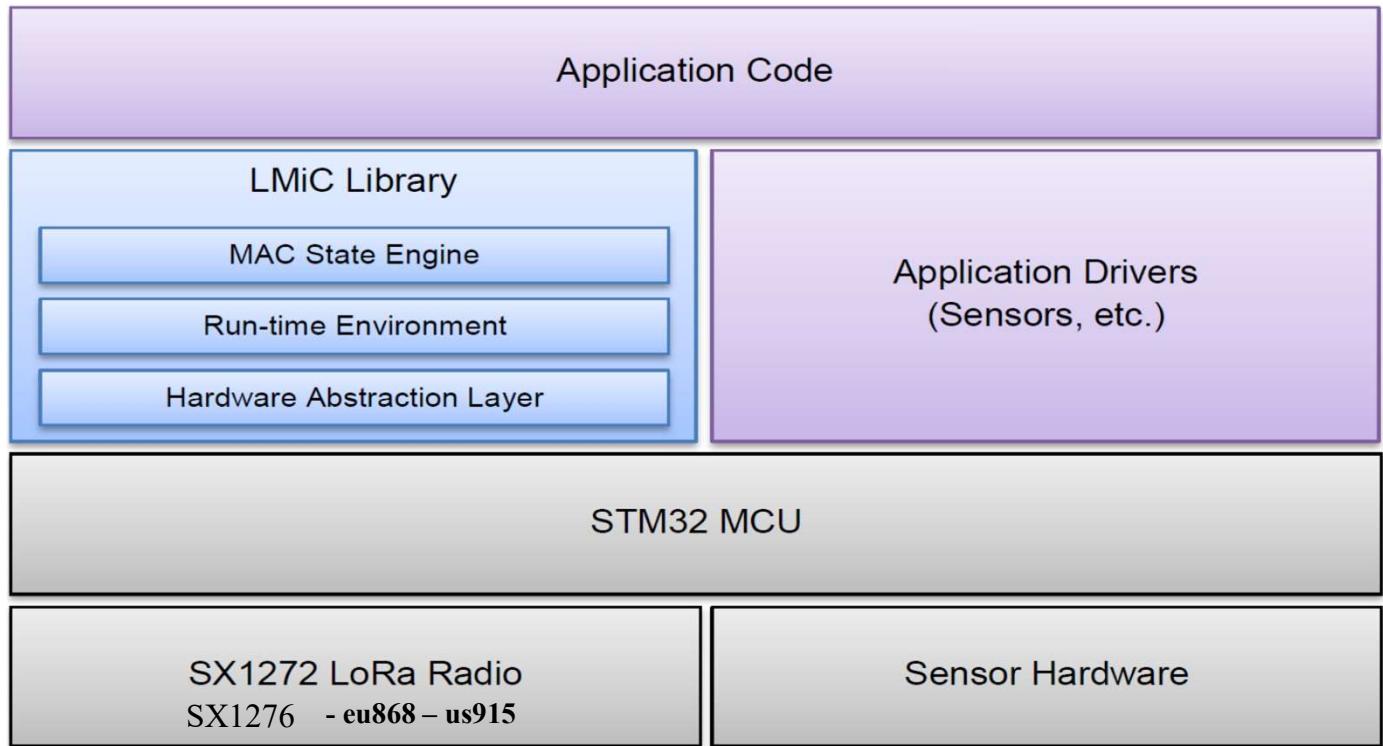
1. La Stack la plus connue est développée par SEMTECH. Cette Stack s'appelle LoRa MAC Node et est disponible pour tous les microcontrôleurs. Elle est très bien maintenue et suit la spécification LoRaWAN.
2. L'autre Stack bien connue est LMIC (LoRa Mac In C). C'est la Stack préférée lorsqu'on utilise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

notes



- LMIC = Real Time Kernel + LoRaWan MAC



The IBM LoRaWAN C-library (LMiC) is a portable implementation of the LoRa™ MAC specification for the C programming language. It supports both the EU-868 and the US-915 variants of the specification and it can handle class A and class B devices. The library takes care of all logical MAC states and timing constraints and drives the SEMTECH SX1272 radio. This way, applications are free to perform other tasks and the protocol compliance is guaranteed by the library. In order to ensure compliance with the specification and associated regulations, the state engine has been tested and verified using a logic simulation environment. The library has been carefully engineered to precisely satisfy the timing constraints of the MAC protocol and to even consider possible clock drifts in the timing computations. Applications can access and configure all functionality via a simple event-based programming model and do not have to deal with platform-specific details like interrupt handlers. By using a thin hardware abstraction layer (HAL), the library can be easily ported to new hardware platforms. For the STM32/Cortex-M3 platform, a reference implementation of the HAL is supplied and the overall code footprint of all components on this platform is less than 20K.

In addition to the provided LMIC library a real-world application also needs drivers for the sensors or other hardware it desires to control. These application drivers are outside the scope of this document and their code will not be provided by IBM.

notes



- Sélectionner le microcontrôleur STM32L432KCx en la préconfigurant par défaut

- Horloge noyau à 32 KHz via le timer 7 (valider IT)

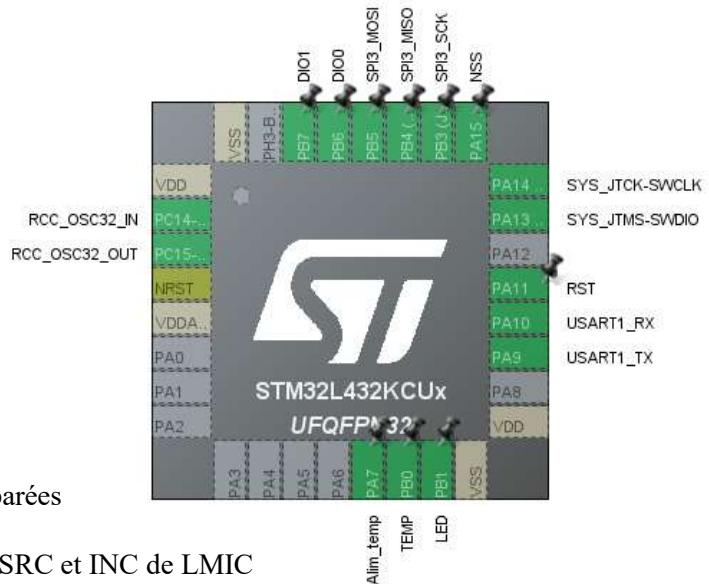
- Liaison radio SPI3

CPOL = 0 / CPHA = 0

Motorola format MSB first

1,25Mbauds

+ DIO0, DIO1 en IT externe + NSS



- Liaison Debug UART1 (115 200 baud)

- Generate avec les initialisations des périphériques séparées

- Copier/coller dans le projet les fichiers contenus dans SRC et INC de LMIC

- Insérer dans les propriétés du projet les symboles: CFG_eu868 et CFG_sx1276_radio
Propriété/ C++ General / Paths and Symbols/# Symbols

- Compléter votre main.c avec le contenu du fichier Append_main.c (campus)

- Corriger les erreurs de compilation , Attention mettre les define entre Begin et End



Le noyau temps réel LMIC nécessite, bien entendu un timer.

Le noyau a besoin d'un timer cadencé entre 15 us et 100 us, par défaut il est configuré à 32 khz (31 us) c'est pourquoi on recommande d'utiliser un basic timer comme le timer 7 avec une fréquence d'horloge de 32 khz. Ce timer sert de base de temps au noyau, le temps est donc toujours modulo 31 us. Pour tenir compte des débordements et éviter ainsi que le temps revienne à zéro toutes les 2 secondes, ces overflows sont compté dans l'interruption timer associée. On forme ainsi un compteur de temps avec en poids fort le nombre d'overflow et en poids faible le contenu du timer soit 37 heures de comptage. Les tâches (ou job) créés sont toutes périodiques, avec une période pouvant varier de 31 us à 37 heures.

Le LMIC utilise des noms de pins prédéfinis donc il faut respecter la notation de celle-ci dans Cube sous peine d'erreurs de compilation.

Les répertoires SRC et INC de LMIC sont sur campus

1. [SMIN](#)

2. [2A Semestre 7](#)

3. [GP - Conception de Systèmes Electroniques 2](#)

4. [Projet IOT](#)

Librairies LMIC

Append_main

notes

Remarque: pour l'instant les EUI restent vides pour pouvoir compiler. Dans un deuxième temps, le site TTN permettra de générer ces numéros, il faudra alors les reporter.



- Valider le noyau temps réel sans la radio avec la tâche hellofunc à la place de la tâche « initfunc »

```
// counter
static int cnt = 0;
static osjob_t hellojob;
static void hellofunc (osjob_t* j) {
    // say hello
    debug_str("Hello World!\r\n");
    // log counter
    debug_val("cnt = ", cnt);
    // toggle LED
    debug_led(++cnt & 1);
    // reschedule job every second
    os_setTimedCallback(j, os_getTime() + sec2osticks(1), hellofunc);
}
```

```
// setup initial job dans main.c
os_setCallback(&hellojob, hellofunc);
// execute scheduled jobs and events
os_runloop();
// (not reached)
return 0;
```

Les fonctions debug écrivent dans le terminal série à 115200 baud et la led1 verte

Résultat attendu

led clignotante 1s et terminal suivant:

===== DEBUG STARTED =====

Hello World!

cnt = 00000000

Hello World!

cnt = 00000001

Hello World!

cnt = 00000002

.....

Attention il faut valider l'interruption du Timer7 sinon le comptage ne défile pas.

notes



□ Test du join request avec TTN

```

static osjob_t blinkjob;
static u1_t ledstate = 0;
static void blinkfunc (osjob_t* j) {
    // toggle LED
    ledstate = !ledstate;
    debug_led(ledstate);
    // reschedule blink job
    os_setTimedCallback(j, os_getTime() + ms2osticks(100), blinkfunc);
}

///////////////////////////////
// LMIC EVENT CALLBACK
////////////////////////////

void onEvent (ev_t ev) {
    debug_event(ev);

    switch(ev) {

        // network joined, session established
        case EV_JOINING:
            debug_str("try joining\r\n");
            blinkfunc(&blinkjob);
            break;
        case EV_JOINED:
            os_clearCallback(&blinkjob);
            debug_led(1);
            //reportfunc(&reportjob);
            break;
    }
}

```

Il faut utiliser la tâche initfunc originale (elle initialise la radio). **Os_setCallback(&initjob, initfunc);**

On peut rajouter une tâche de clignotement de led pour faire plus beau. En cas de succès du join la led s'arrête de clignoter.

Pour l'instant on peut mettre en commentaire reportfunc qui comme son nom l'indique envoie la donnée après la phase joined.

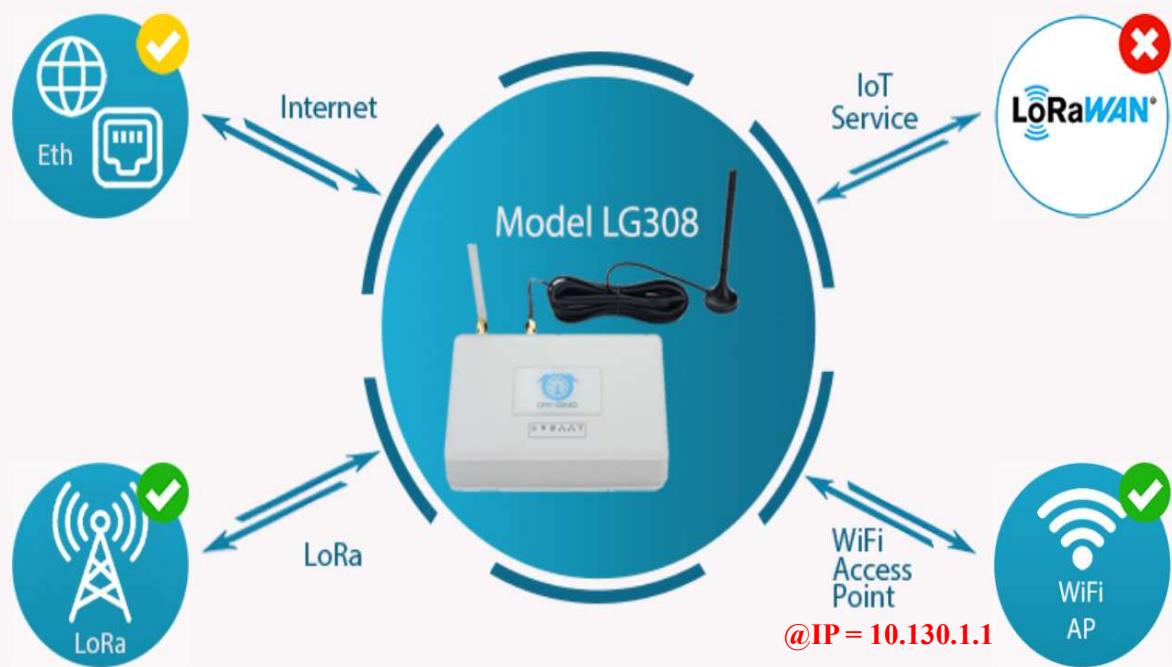
Sortie du terminal:

```
===== DEBUG STARTED =====
JOINING
Try joining
EV_TXSTART          // début du uplink join request
Unknown event        // TXSTART n'est pas pris en compte dans la machine d'état
JOINED              // Bien sur, avant il faut créer un compte sur TTN et le configurer
```

notes



System Overview



LoRaWAN Service			
Process:	LoRaWAN process pit_frd	Running	
Status:	online		
Server:	192.168.104.158		



Passerelle peut être utilisé: Dragino Gateway LG308 (dans mon bureau).

Avantage: la configuration se fait via internet **@IP:8000**

User Name: root

Password: petit dragon

notes



LoRaWAN Configuration

General Settings

Email

Gateway ID

Primary LoRaWAN Server

Service Provider Server Address

Uplink Port Downlink Port

Packet Filter

Fport Filter ? DevAddr Filter ?

Current Mode: LoRaWAN Semtech UDP



Par défaut la passerelle est configuré pour transmettre vers le serveur public TTN mais bien sûr on peut avoir son propre network server et donc pointer dessus en renseignant le paramètre **Server Address**.

notes



- <https://www.thethingsnetwork.org>
- Create an account



CREATE AN ACCOUNT

Create an account for The Things Network and start exploring the world of Internet of Things with us.

USERNAME

This will be your username — pick a good one because you will not be able to change it.



EMAIL ADDRESS

You will occasionally receive account related emails. This email address is not public.



PASSWORD

Use at least 6 characters.

Il faut avoir une adresse mail valide. Une fois le compte créer il faut lancer la console

notes





gateway-ismin-semtech
ID: gateway-ismin-semtech

Last seen 1 second ago ↑ 0 ↓ 0 1 Collaborator 0 API keys Created 3 days ago

General information		Live data
Gateway ID	gateway-ismin-semtech	See all activity →
Gateway EUI	AA 55 5A 00 00 00 09 67	• Live data
Gateway description	None	11:07:31 Receive gateway status Metrics: { ackr: 0, rxfw: 0, rxin: 0 }
Created at	Jul 16, 2021 14:10:57	11:07:01 Connect gateway
Last updated at	Jul 16, 2021 14:13:24	
Gateway Server address	eu1.cloud.thethings.network	Location Change location settings →
LoRaWAN information		No location information available
Frequency plan	EU_863_870_TTN	
Global configuration	Download global_conf.json	

La console permet d'enregistrer vos propres gateways.

La gateway a déjà été enregistrée, c'est juste pour information ou si dans la cadre d'un projet vous avez à configurer votre propre gateway.

Chaque gateway est référencée par un numéro unique EUI à renseigner sur TTN.

notes



Owner *

acaciomarques



Application ID *

my-new-application

Application name

My new application



Description

Description for my new application

Optional application description; can also be used to save notes about the application

Create application

Donner un nom et une description à votre application, c'est purement informatif. Par la suite il faudra ajouter le device (end-node).

notes



Ocass applio

Overview

End devices

Live data

Payload formatters

Integrations

Collaborators

API keys

General settings

Register end device

From The LoRaWAN Device Repository Manually

Frequency plan *

Europe 863-870 MHz (SF9 for RX2 - recommended)

LoRaWAN version *

MAC V1.0.2

Regional Parameters version *

PHY V1.0.2 REV A

Show advanced activation, LoRaWAN class and cluster settings ^

Activation mode *

Over the air activation (OTAA)

Activation by personalization (ABP)

Define multicast group (ABP & Multicast)

Additional LoRaWAN class capabilities

None (class A only)

Network defaults

Use network's default MAC settings

Cluster settings

Use external LoRaWAN backend servers

DevEUI *

70 B3 D5 7E D0 04 BB C2

Generate 1/50 used

AppEUI *

00 00 00 00 00 00 00 00

Fill with zeros

AppKey *

B3 4C BD E6 25 E6 E2 74 DF 6C 89 84 F1 54 85 1B

Generate

End device ID *

eui-70b3d57ed004bbc2

This value is automatically prefilled using the DevEUI

After registration

View registered end device

Register another end device of this type

Register end device

< Hide sidebar

MINES Saint-Etienne

63 Aout, 2024 Copier EUI

Choisir la configuration manuelle car notre carte n'est pas référencée.

La version du protocole Lorawan installé sur votre end-node doit être indiqué, voir doc LMIC.

Il faut générer un DevEUI , AppEUI et AppKey qu'il faudra reporter sur le main du end-node. Le copier/coller se fera sur la vue suivante qui récapitule les informations de votre end-node.

notes



Register end device

APPLICATION : réseau – objet – passerelle – TTN - Plateforme IOT

Live data

Overview

End devices

Live data

Payload formatters

Integrations

MQTT

Webhooks

Storage Integration

AWS IoT

Azure IoT Hub

LoRa Cloud

Collaborators

API keys

General settings

Hide sidebar

eui-70b3d57ed004bbc2

ID: eui-70b3d57ed004bbc2

Last activity 4 minutes ago

Overview Live data Messaging Location Payload formatters Claiming General settings

General information

End device ID: eui-70b3d57ed004bbc2

Description: This end device has no description

Created at: Jan 25, 2022 11:27:37

Activation information

AppEUI: 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 (lsb)

DevEUI: 0xC2, 0xBB, 0x04, 0xD0, 0x7E, 0x00 (lsb)

Root key ID: n/a

AppKey: 0xB3, 0x4C, 0xBD, 0xE6, 0x2 (msb)

NwkKey: n/a

Session information

Device address: 26 0B 64 90

NwkSKey: (redacted)

SNwkSIntKey: (redacted)

NwkSEncKey: (redacted)

AppSKey: (redacted)

Live data

See all activity

↑ 12:08:57 Forward uplink data message Payload: { accelerometer_2: {} }
↑ 12:08:57 Successfully processed data message DevAddr: 26 0B 64 90 F
↑ 12:08:40 Forward uplink data message Payload: { accelerometer_2: {} }
↑ 12:08:40 Successfully processed data message DevAddr: 26 0B 64 90 F
↑ 12:08:27 Forward uplink data message Payload: { accelerometer_2: {} }
↑ 12:08:27 Successfully processed data message DevAddr: 26 0B 64 90 F

Copier

Location

Change location settings

No location information available

64 Aout, 2024 Test Uplink

AppKey: est une clé d'encryption spécifique à un end-device, utilisée durant OTAA pour dériver la AppSKey et la NwkSkey.

Copier les 3 EUI dans le main de votre device. Attention les 2 premiers sont LSB first et le dernier MSB first.

Recompiler et tester sur le terminal le join request:

Sortie du terminal:

```
===== DEBUG STARTED =====
```

JOINING

Try joining

EV_TXSTART

Unknown event

JOINED

Si le join ne se fait pas augmenter le Data Rate durant le join request fixer SetDrJoin à DR_SF11 dans le fichier Lmic.c

notes



- Tester la transmission Uplink avec la valeur fixe par défaut du capteur 0xDF
Pour cela décommenter la tâche périodique de 15 s **reportfunc** après le JOINED dans la machine d'état

```

static osjob_t reportjob;
// report sensor value every minute
static void reportfunc (osjob_t* j) {
// read sensor
u2_t val = readsensor();
debug_val("val = ", val);
// prepare and schedule data for transmission
LMIC.frame[0] = val << 8;
LMIC.frame[1] = val;                                // La fonction LMIC_setTxData2 envoie
// (port 1, 2 bytes, unconfirmed)                  // la trame Lora : LMIC.frame
// reschedule job in 15 seconds
os_setTimedCallback(j, os_getTime()+sec2osticks(15), reportfunc);
}

```

Vous devriez observer tous les 15 s la valeur de payload 0xDF dans TTN via l'onglet du devices **Live data et dans le terminal série:**

===== DEBUG STARTED =====

JOINING

try joining

EV_TXSTART

Unknown event

JOINED

val = DF

EV_TXSTART

Unknown event

TXCOMPLETE

EV_TXCOMPLETE (includes waiting for RX windows)

val = DF

EV_TXSTART

Unknown event

TXCOMPLETE

EV_TXCOMPLETE (includes waiting for RX windows)

val = DF

notes



Tester le downlink

- [Overview](#)
- [End devices](#)
- [Live data](#)
- [Payload formatters](#)
- [Integrations](#)
- [MQTT](#)
- [Webhooks](#)
- [Storage Integration](#)
- [AWS IoT](#)
- [Azure IoT Hub](#)
- [LoRa Cloud](#)
- [Collaborators](#)
- [API keys](#)
- [General settings](#)

eui-70b3d57ed004bbc2
ID: eui-70b3d57ed004bbc2

↑ 20
↓ 2
Last activity 2 hours ago

Overview
Live data
Messaging
Location
Payload formatters
Claiming
General settings

Uplink
Downlink

Schedule downlink

Insert Mode

Replace downlink queue

Push to downlink queue (append)

FPort*

Payload type

Bytes JSON

Payload

The desired payload bytes of the downlink message

Confirmed downlink

Schedule downlink



66

Aout, 2024

Test downlink

Le message suivant s'affiche sur le terminal en cas de downlink:

TXCOMPLETE

EV_TXCOMPLETE (includes waiting for RX windows)

Received bytes of payload

val = 000000DF

EV_TXSTART

Unknown event

notes



- Configurer le downlink : General settings / Network layer / expand

The screenshot shows the TTN LoRaWAN network configuration interface. On the left, a sidebar lists various settings: Overview, End devices (selected), Live data, Payload formatters, Integrations (MQTT, Webhooks, Storage Integration, AWS IoT, Azure IoT Hub, LoRa Cloud), Collaborators, API keys, and General settings. The main area is titled "Advanced MAC settings". It includes fields for "Frame counter width" (32 bit selected), "Desired Rx1 delay" (set to 1), "Desired Rx1 data rate offset" (0), "Desired Rx2 data rate index" (0), "Desired Rx2 frequency" (869525000), "Desired maximum duty cycle" (100%), "Factory preset frequencies" (+ Add Frequency), "Status count periodicity" (200), "Status time periodicity" (86400), "Use ADR" (checked), and "ADR margin" (20). A note "Rx1 1s après le uplink" is placed next to the Rx1 delay field, "même DR que le uplink" next to the Rx1 data rate offset, and "RX2 en SF12" next to the Rx2 data rate index. A note "SNR marge pour l'ADR" is placed next to the ADR margin field. At the bottom is a "Save changes" button.



Si le downlink ne fonctionne pas , vous pouvez expérimenter les paramètres ci-dessus.

Par défaut RX1 delay est à 5 s sur TTN , ce qui ne correspond pas à la norme LoRaWan.

Enlever Status count periodicity et Status time periodicity

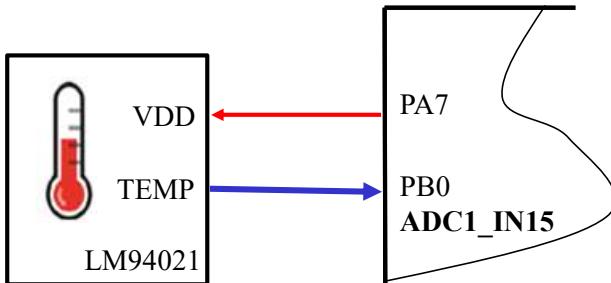
notes



- ❑ Commander la LED1 sur le end device via TTN
- ❑ Il faut compléter le case EV_TXCOMPLETE pour traiter la donnée de downlink que l'on récupère dans la trame

```
case EV_TXCOMPLETE:  
    debug_str("EV_TXCOMPLETE (includes waiting for RX  
windows)\r\n");  
    if (LMIC.txrxFlags & TXRX_ACK)  
        debug_str("Received ack\r\n");  
    if (LMIC.dataLen) {  
        debug_valdec("Received bytes of payload\r\n:",LMIC.dataLen);  
        debug_val("Data = :",LMIC.frame[LMIC.dataBeg]);  
        debug_led(LMIC.frame[LMIC.dataBeg]);  
    }  
break;  
  
// LMIC.databeg représente le début de la zone data dans la trame.  
//debug_led actionne la led.  
  
// Downlink 00 pour éteindre, 01 pour allumer
```

notes



Linéarisation entre 0 et 50 °C $T \text{ } ^\circ\text{C} = (1034 - V_{\text{TEMP}}) / 5,48$

Conversion sur 12 bits: $V_{\text{TEMP}} (\text{ mv }) = 3300.N / 4095$

$$T_{1000 \text{ } ^\circ\text{C}} = 188686 - 147. N_{(12 \text{ bits})}$$

- Lancement d'une conversion toutes les secondes via le timer 6
- Remplacer la donnée en uplink (0xDF) dans readsensor () par la température
- Vérifier qu'elle remonte bien via le terminal ou TTN

EV_TXSTART
 Unknown event
 TXCOMPLETE
 EV_TXCOMPLETE (includes waiting for RX windows)
 val = 005DEE **24 °C**

La fonction de transfert du capteur est donnée dans la data sheet que je vous recommande fortement de lire. Cette fonction de transfert n'est pas parfaitement linéaire mais on peut faire une approximation linéaire sur une plage de température donnée. Par exemple de 0°C à 50 °C. Pour des questions de précision mieux vaut exprimer la température en millième de degrés. Attention il est fortement conseillé de faire une calibration de l'ADC en début de main pour éliminer l'offset souvent non négligeable.

La CallBack PeriodElapsed des timers est déjà utilisée par TIM7 dans Hal.c pour incrémenter le temps du noyau. On ne peut pas déclarer 2 fois la fonction CallBack. Je vous conseille de transférer cette fonction dans le main pour y traiter TIM7 et TIM6. Attention aux déclarations de variable utilisé dans plusieurs fichiers (extern).

Remarque: vous pouvez utiliser debug_valdec au lieu de debug_val dans la fonction reportfunc pour afficher la température en décimal dans le terminal, elle sera plus facilement interprétable.

Pour obtenir un affichage automatique sur les sites d'affichage de donnée il faut utiliser le format Low Power Payload

notes



1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

TYPE	LPP	Size	Data Resolution per bit
Digital Input	0x00	1	1
Digital Output	0x01	1	1
Analog Input	0x02	2	0.01 Signed
Analog Output	0x03	2	0.01 Signed
Illuminance Sensor	0x65	2	1 Lux Unsigned MSB
Presence Sensor	0x66	1	1
Temperature Sensor	0x67	2	0.1 °C Signed MSB
Humidity Sensor	0x68	1	0.5 % Unsigned
Accelerometer	0x71	6	0.001 G Signed MSB per axis
Barometer	0x73	2	0.1 hPa Unsigned MSB
Gyrometer	0x86	6	0.01 °/s Signed MSB per axis
GPS Location	0x88	9	Latitude : 0.0001 ° Signed MSB
			Longitude : 0.0001 ° Signed MSB
			Altitude : 0.01 meter Signed MSB

```
static void reportfunc (osjob_t* j) {
    // read sensor
    float val = readsensor_temp();
    debug_valdec("val temp = ", val);
    LMIC.frame[0] = 0;
    LMIC.frame[1] = 0x67; //temp
    val = val/100; // en 0.1 °C
    LMIC.frame[2] = val >> 8;
    LMIC.frame[3] = val;
    . . .
    . . .
}
```

Le Cayenne Low Power Payload (LPP) est un moyen pratique et facile d'envoyer des données sur des réseaux LPWAN tels que LoRaWAN. Le Cayenne LPP est conforme à la restriction de taille de la charge utile, qui peut être abaissée à 11 octets, et permet à l'appareil d'envoyer plusieurs données de capteurs en même temps. Cayenne LPP permet à l'appareil d'envoyer différentes données de capteurs dans la même trame. Pour ce faire, les données de chaque capteur doivent être préfixées de deux octets :

Canal de données : Identification unique de chaque capteur, par exemple "capteur intérieur".

Type de données : Identifie le type de données dans la trame, par exemple "température".

Si vous prenez la peine de formater vos données comme indiqué par le format LPP alors vos données seront automatiquement reconnu par les sites de visualisation des données.

La mise au format LPP peut se faire dans la tache « reportfunc ».



- ❑ Faites remonter les données toutes les 15 s.
- ❑ Vérifier sur le terminal et sur TTN la présence des données
- ❑ En choisissant le payload format LPP_cayenne , les données pourront être vu en clair dans Live Data

The screenshot shows the Cayenne IoT Platform interface. On the left, a sidebar menu includes Overview, End devices (selected), Live data, Payload formatters (expanded), Integrations, MQTT, Webhooks, Storage Integration, AWS IoT, and Azure IoT Hub. The main area displays a device card for 'eui-70b3d57ed004bbc2' with an ID of 'eui-70b3d57ed004bbc2'. It shows 20 uplink and 2 downlink messages, with the last activity being 4 hours ago. Below the card are tabs for Overview, Live data, Messaging, Location, and Payload formatters (selected). Under the 'Payload formatters' tab, there are tabs for Uplink and Downlink, with 'Uplink' selected. A 'Setup' section contains a 'Formatter type*' dropdown set to 'CayenneLPP' and a 'Save changes' button. The bottom of the screen features a green footer bar with the MINES Saint-Étienne logo, the date '71 — Aout, 2024 —', and an 'Affichage' link.

notes



Edit webhook for Datacake
Send data to Datacake via TTI adapter
[About Datacake](#) | [Documentation](#)

General settings

Webhook ID *
my-data-io

Webhook format *
JSON



TTN propose un certains nombres d'outils facilement intégrable notamment des outils d'affichage.
Choisissez en un Datacake par exemple.

Faire: **Webhooks +Add Webhook et Site d'affichage**

Suivre les instructions dans la documentation :

Sign Up

First Steps

Adding LoRaWAN Devices

The Things Stack (TTN/TTI) Manual Setup

Create Integration on TTI

Downlinks

notes



projet iot

Serial Number

70B3D57ED0069189

Last update

Tue Aug 6, 2024 09:58:19 GMT+02:00

[Tableau de bord](#)[Historique](#)[Downlinks](#)[Configuration](#)[Débogage](#)[Règles](#)[Autorisations](#)[ON](#)[OFF](#)[Downlink](#)

35 seconds ago

Temperature Sensor 0

35 seconds ago

23,8 °C

température

35 seconds ago

Default

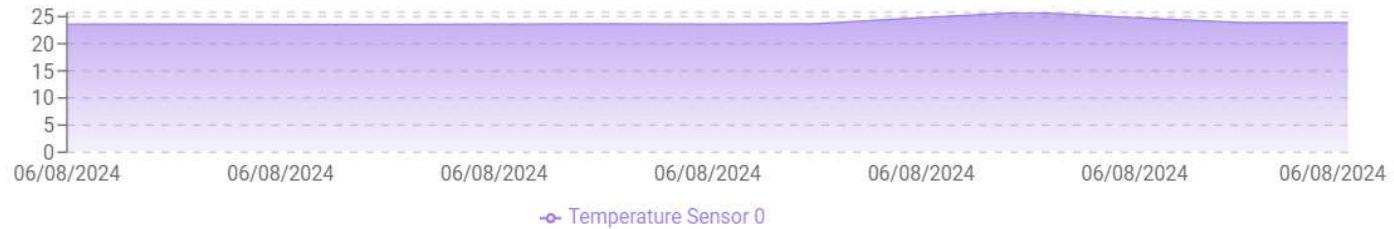
1H

1D

1W

1M

Personnalisé



73

Aout, 2024

Mettre en place quelques règles

Dans votre tableau de bord vous pouvez ajouter des widgets pour afficher la température. Vous pouvez également créer des règles: *si la température dépasse 26 degrés allumer la LED1 sinon l'éteindre*. Pour activer la LED1 il faut passer par le menu downlink.

notes



Configuration du Downlink

Nom

ON

Description

led1 ON

UUID du downlink

75e6feba-a6d9-4f40-a13d-1e4328060846

Copier

Champs utilisés

Si votre fonction d'encodage prend en compte les champs de l'appareil, il convient de les spécifier ici. Ils seront utilisés pour créer le modèle pour le générateur du downlink.

 Déclenchement en cas de mesures

Si cette option est activée, chaque fois que l'appareil enregistre une mesure dans l'un des champs utilisés, le downlink sera envoyé automatiquement.

Port

1

Encodeur Payload

```
1 * function Encoder(measurements, port) {
2     return [0x01, 0x00, 0x01];
3 }
```



Flotte > projet iot > Règles > Modifier la règle

Modifier la règle

Nom

surchauffe

Conditions

Temperature Sensor 0 est plus grand ou égal à 26 °C Avec une hystérésis de

Si

1 °C

+

Then Envoyer le downlink

Appareil

projet iot

Downlink

ON



74

Aout, 2024

A vous de jouer

Ajouter un downlink ON et OFF, voir exemple. Il faut utiliser l'encoder payload pour envoyer les données sous forme de tableau de caractères au format LPP: Data, Type, Channel.

Une fois le downlink créé vous pouvez l'utiliser dans une règle.

notes



Créer avec DALL-E de ChatGPT

notes