Architecture de Description de **LoRa**WAN avec **T**he **T**hings **N**etwork

Agenda

- Définition et structure d'un réseau LoRaWAN
  - Spécifications (SEMTECH)
  - Structure d'un réseau
  - Chiffrement et authentification
  - Classes
- Activer un premier objet connecté LoRaWAN avec TTN
- Comment configurer : type d'activation, ADR
- Analyser et optimiser les trames LoRaWAN
  - Packet Forwarder Gateways
  - Format payload : décoder, convertir, valider
- Mise en Oeuvre d'une passerelle TTIG

Architecture de Description de **LoRa**WAN avec **T**he **T**hings **N**etwork

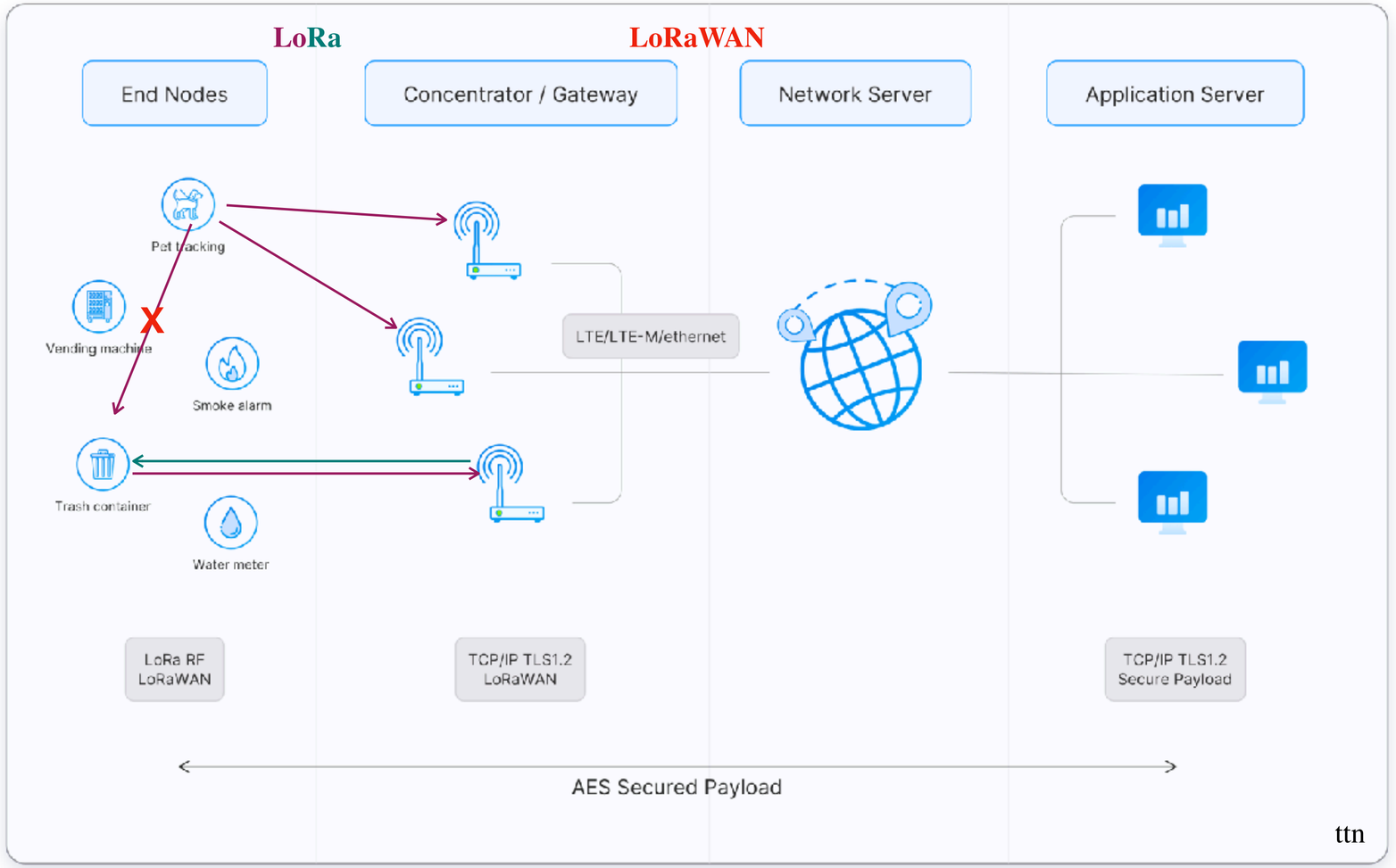
Partie

1

Définition et structure d'un réseau LoRaWAN

Spécifications

Différence LoRa et LoRaWAN ?



LoRa (RF) type de modulation CSS radio (longue portée) assurant le transit des informations bidirectionnelles entre les objets connectés et/ou les passerelles

LoRa peut se passer du LoRaWAN mais il vous faudra créer votre propre application pour l'utiliser (genre de Peer To Peer)

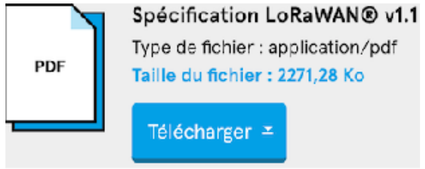
LoRaWAN protocole de communication depuis un objet jusqu'aux serveurs de réseaux et d'application (**Network server** et **Application server**)

LoRaWAN utilise du LoRa

Le protocole LoRaWAN : spécifications (LoRa SEMTECH , LoRaWAN LoRa Alliance®) et regional parameters

LoRaWAN : protocole de couche MAC (Media Access Control) construit sur la modulation LoRa
couche logicielle qui définit comment les appareils utilisent le matériel LoRa et le format des messages à transmettre
développé et maintenu par la LoRa Alliance
1e spécification publiée en janvier 2015
Historique des versions des spécifications LoRaWAN, les dernières spécifications sont 1.0.4 et 1.1

Version	Date de sortie
1,0	Janvier 2015
1.0.1	Février 2016
1.0.2	juillet 2016
1.1	Octobre 2017
1.0.3	juillet 2018
1.0.4	Octobre 2020



prévue en 2021 ?

amélioration de la sécurité (jeu de clés supl.)
classe B
fonctionnalité améliorée comme l'horodatage
changement de réseau / localisation
…

formation traite essentiellement de la version 1.0.X du protocole LoRaWAN

- **Ultra faible consommation** - jusqu'à 10 ans avec une pile bouton
- **Longue portée** - gateway plus de 10 Km en rurales et jusqu'à 3 Km en urbaines
- **Couverture intérieure profonde** - couvre facilement les bâtiments à plusieurs étages
- **Spectre sans licence** - gratuité d'utilisation
- **Géolocalisation** - détermination de l'emplacement des terminaux sans GPS par triangulation de passerelles
- **Haute capacité** - Les serveurs gèrent des millions de messages provenant de milliers de passerelles
- **Déploiements publics et privés** - déploiement facile sur les réseaux publics et privés avec le même matériel (passerelles, terminaux, antennes)
- et les mêmes logiciels (transmetteurs de paquets UDP, logiciel Basic Station, piles LoRaWAN pour terminaux)
- **Sécurité de bout en bout** - communication sécurisée garantie entre device et serveur d'applications (cryptage AES-128)
- **Mises à jour du micrologiciel sans fil** - mise à jour du micrologiciel à distance (applications et pile LoRaWAN) pour un ou plusieurs devices
- **Itinérance** - transferts transparents d'un réseau à un autre
- **Faible coût** - Infrastructure minimale, nœuds d'extrémité à faible coût et logiciels open source.
- **Programme de certification** - LoRa Alliance certifie les devices et garantit aux utilisateurs finaux leurs fiabilité et conformité à la spécification LoRaWAN
- **Écosystème** - nombreux fabricants d'appareils, de passerelles, d'antennes, de fournisseurs de services réseau et de développeurs d'applications

Architecture de Description de **LoRa**WAN avec **T**he **T**hings **N**etwork

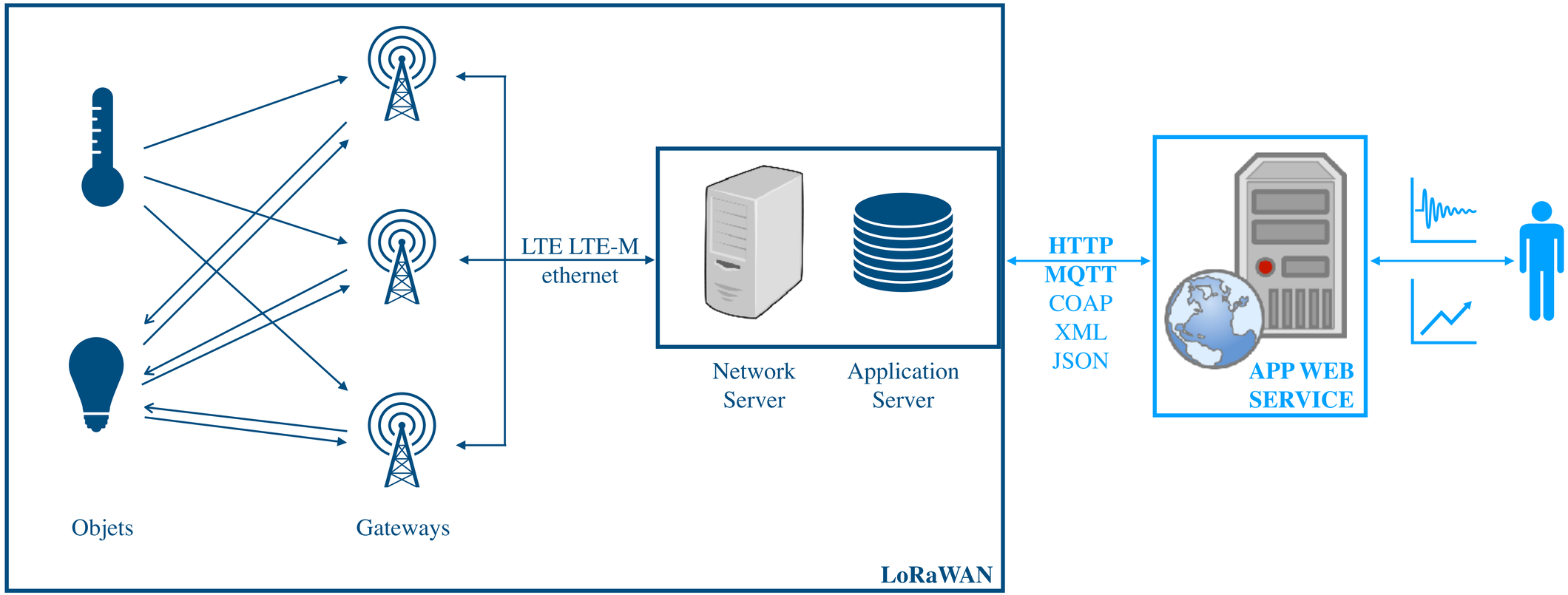
Partie

1

Définition et structure d'un réseau LoRaWAN

Structure d'un réseau

Le protocole LoRaWAN : structure du réseau



Un objet LoRa transmet à toutes les Gateways qui se trouvent dans sa zone de couverture

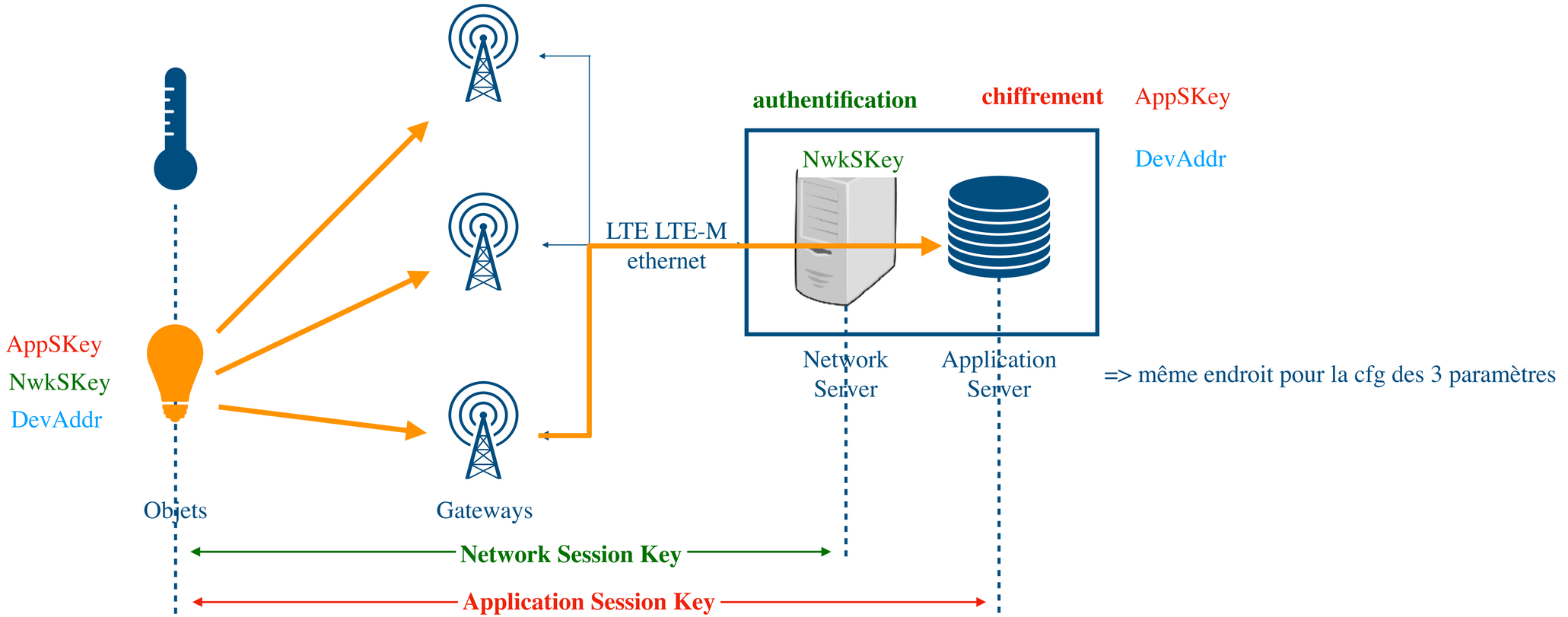
Une Gateway se charge de passer (passerelle) de la transmission Radio à IP (LTE/LTE-M/ethernet) (vis et versa) au format LoRaWAN

Le format LoRaWAN est compris par le Network server et l'Application server

Le protocole LoRaWAN s'arrête à l'application server (données reçues)

Ce qui se trouve à droite de l'Application Server ne concerne plus LoRaWAN (services web, etc… à implémenter via protocoles HTTP, …pour exploiter les données

1 Le protocole LoRaWAN : Network Server et Application Server

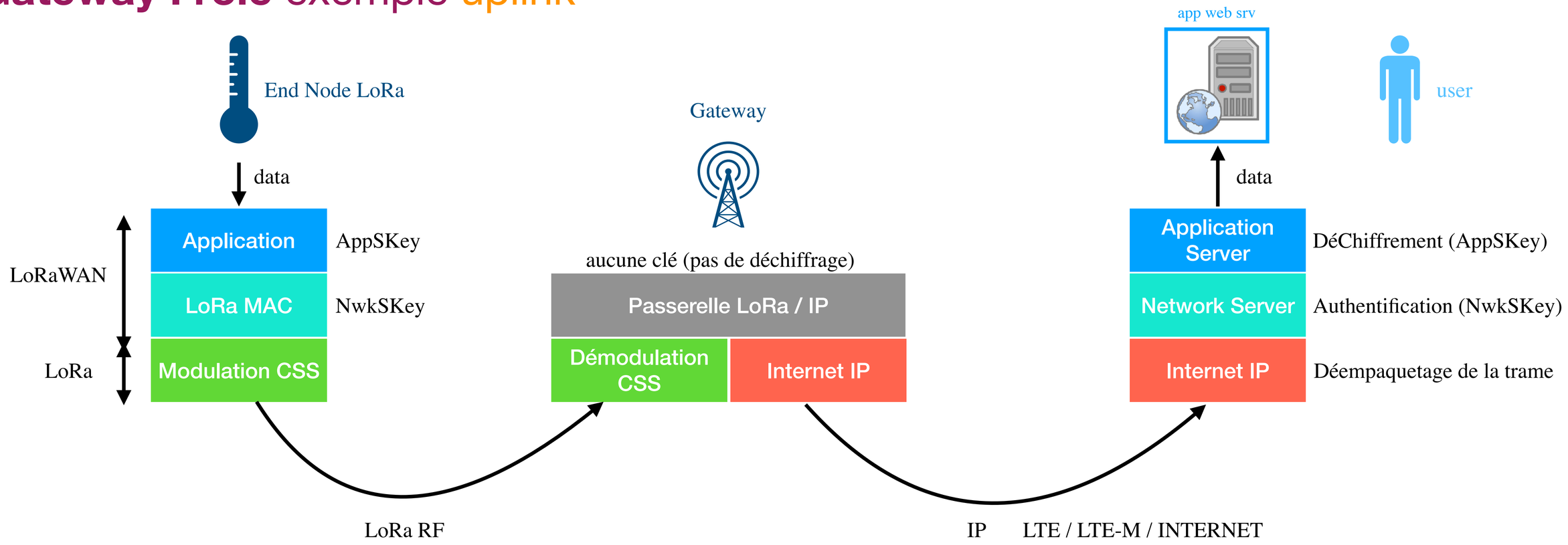


Données chiffrées via l'application server pour un jeu de clés : **Application Session Key** (si on espionne on ne doit pas comprendre l'information même si on sait la lire)

Droits d'authentifications via le network server : **Network Session Key** (on doit savoir de qui la donnée provient)

Adresser les objets via l'application server : **Device Address**

Gateway : rôle exemple uplink



les GW sont intercalées entre les objets et les serveurs

ici une représentation sous forme de couche applicative :

ex data d'un capteur de température : la temp va être chiffré par l'APPSKEY, passe par une authentification via NWKSKEY, ensuite modulation de freq

La gateway récupère via le démodulateur, et ce qu'elle a récupérer elle ne sait pas ce que c'est.

Elle fait juste transiter la signal modulé vers une trame au format ip (protocoles différents (filaire, wifi, etc…)

et envoyé au network server

le networkserver va désempaqueter le message, va l'authentifier (via Nwkskey) et l'application serveur déchiffre le message

la GW ne possède pas de clé, elle fait transiter des infos de nos objets mais aussi d'autres objet qui ne nous appartiennent pas

Architecture de Description de **LoRa**WAN avec **T**he **T**hings **N**etwork

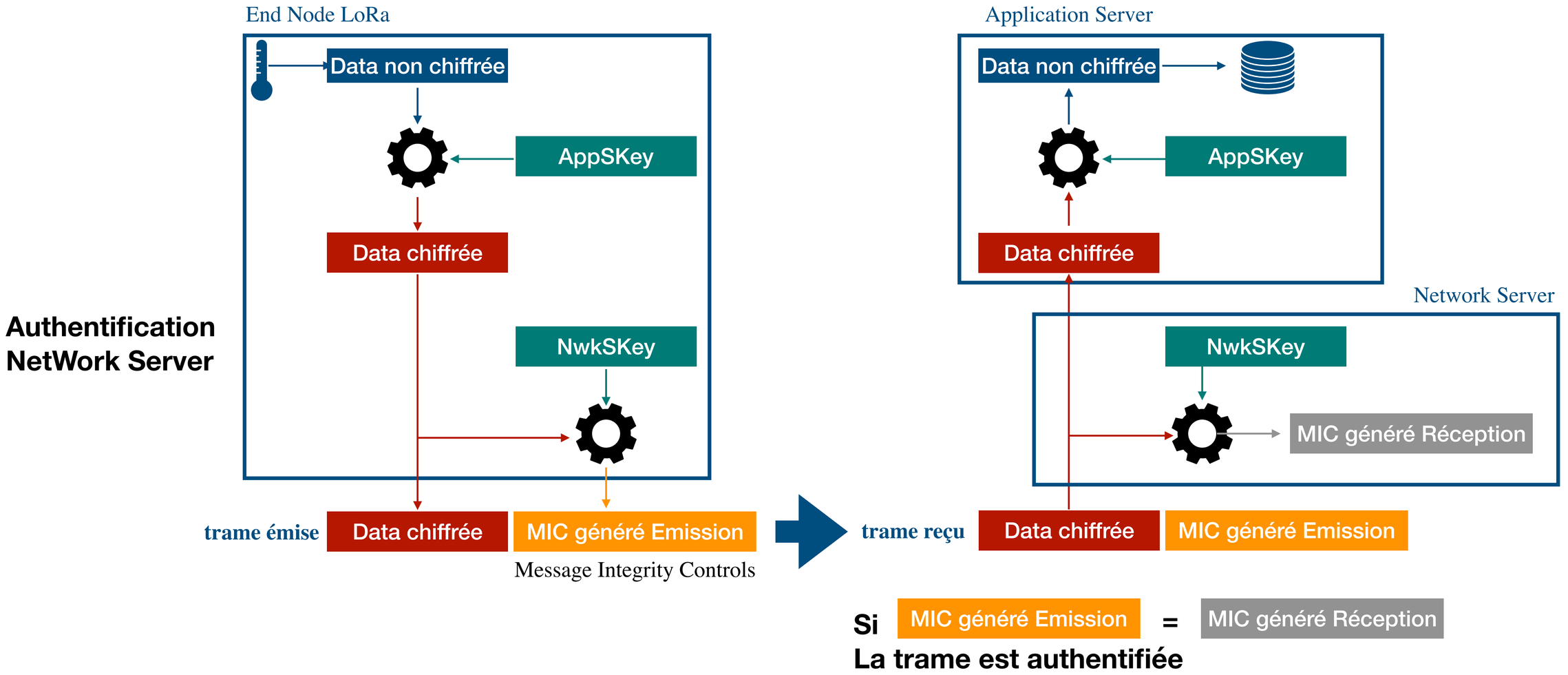
Partie

1

Définition et structure d'un réseau LoRaWAN

Chiffrement et Authentification

Chiffrement des données vers l'Application Server



Si MIC généré Emission = MIC généré Réception

La trame est authentifiée

données chiffrées sont récup par l'authentification et on va s'en servir pour générer un nombre (mic est rajouté à la trame = données + mic qui est transmis via la modulation

données chiffrées sont authentifiées avec le NwkSKey et on va régénérer le mic et on test si le mic est le mic que celui qui est dans la trame. si oui, ça veut dire que le message est authentifié

Architecture de Description de **LoRa**WAN avec **T**he **T**hings **N**etwork

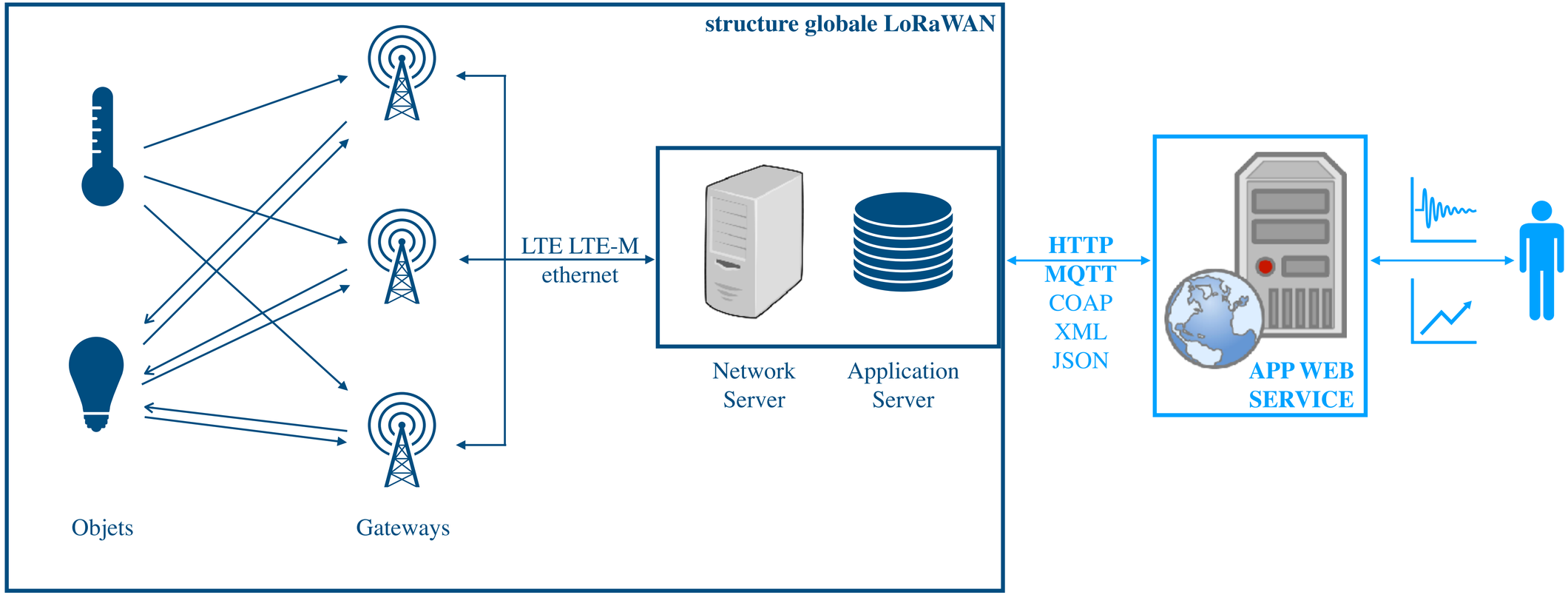
Partie

1

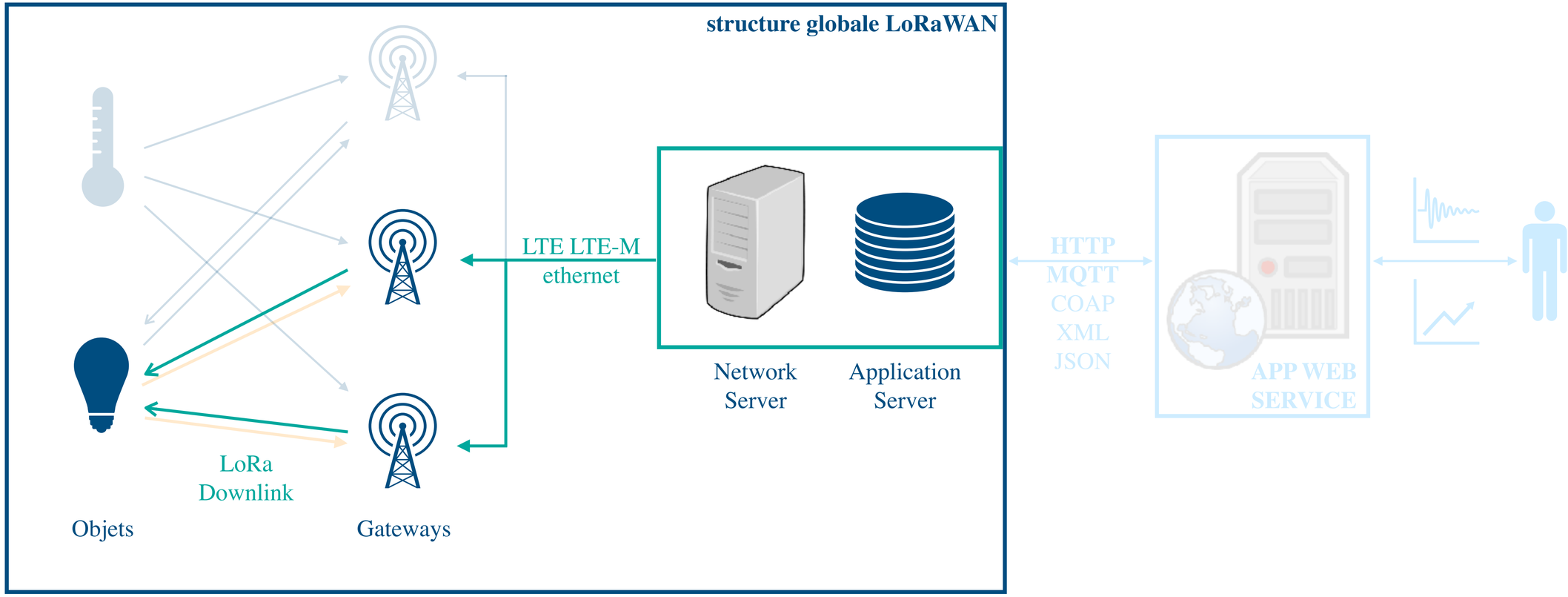
Définition et structure d'un réseau LoRaWAN

Classes

Les classes A B C des Objets



Les classes A B C des Objets



Concernant les objets LoRa on a vu l'envoie d'un flux uplink (exemple ici avec 2 GW visibles dans la zone couverte par la lampe connectée)

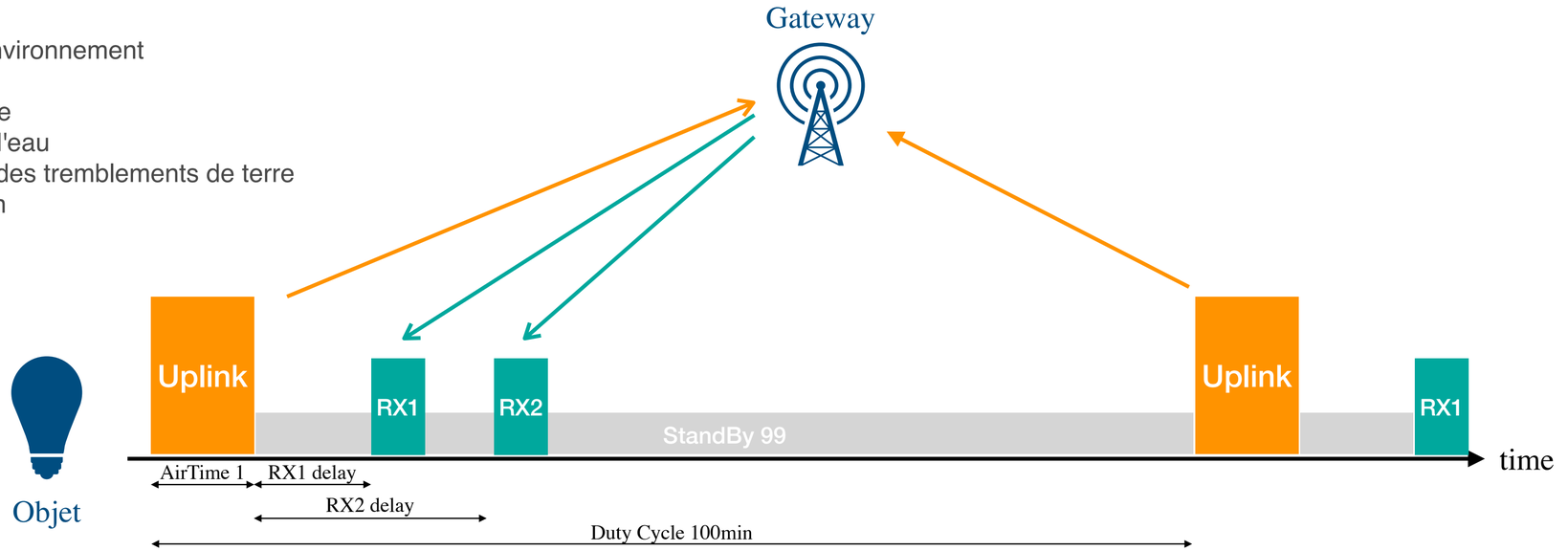
mais il est possible d'avoir des **flux downlink**

Pour comprendre comment un objet passe d'un **flux uplink** à un **flux downlink** on va s'intéresser aux classes A, B et C des objets

La classe A des Objets

explications centrées la modulation LoRa (RF)

- Surveillance de l'environnement
- Suivi des animaux
- Détection d'incendie
- Détection de fuite d'eau
- Détection précoce des tremblements de terre
- Suivi de localisation



2 flux downlink sont tjs réalisés juste après les flux uplink dans des intervalles de temps RX1 et RX2 bien définis : RX1 delay et RX2 delay

délais programmables par défaut : RX1 delay = 1s et RX2 delay = 2s après le flux uplink

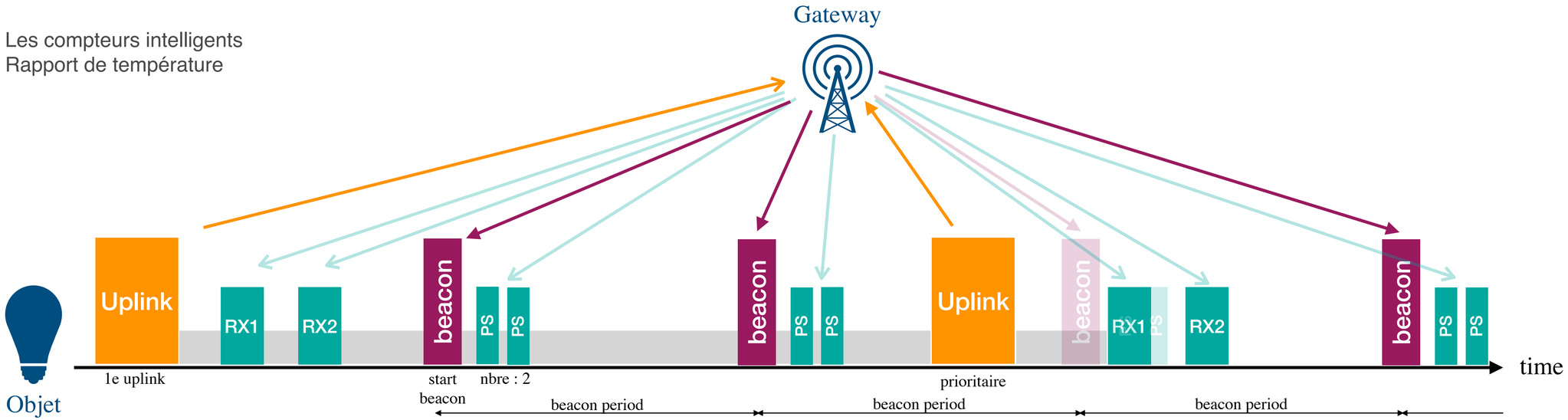
En classe A un objet LoRa ne pourra recevoir des données que s'il a au préalable envoyé un message uplink

Pour joindre un objet LoRa (classe A) il faudra donc que l'objet ait envoyé d'abord une donnée sinon impossible à lui envoyer des données

La classe B des Objets

explications centrées la modulation LoRa (RF)

- Les compteurs intelligents
- Rapport de température



La classe B d'un objet propose le même de mode de fonctionnement de la classe A
+ extension ajoutant des fenêtres de temps préconfigurées pour la réception de message downlink (nombre configurable) de manière périodique :

Des **beacon** sont mises par la passerelle permettant de synchroniser l'objet LoRa et la gateway et ouvrent périodiquement des fenêtres de réception
- le temps entre les **beacon** est connu (**beacon period**)
- le temps pendant lequel l'appareil est disponible pour recevoir des liaisons descendantes est un **slot de ping** (**PS**) le nombre est configurable

Un objet classe B est joignable périodiquement via ce processus même si l'objet n'émet pas de message uplink

le processus beacon dont la période est configurable se met en place tjs après un 1er uplink

la priorité des downlink est donnée sur les RX liés aux process Uplink

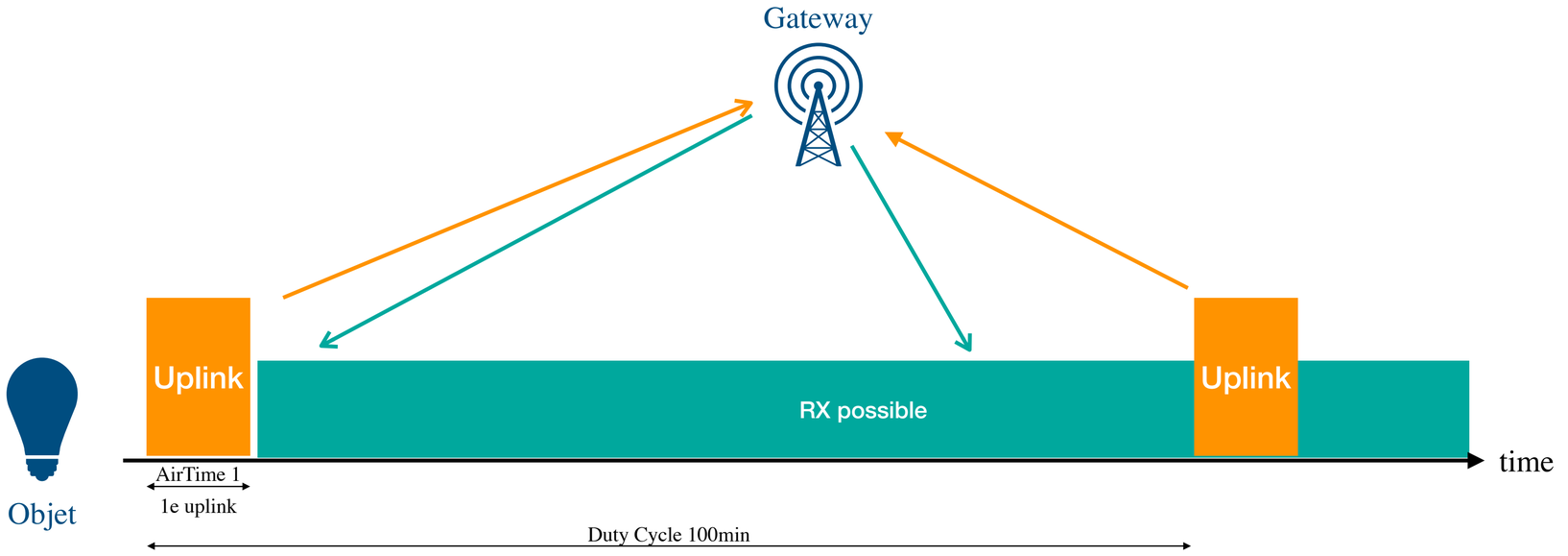
le nombre de PS : fenêtre de ping est configurable
+ il y en a plus l'objet sera joignable mais + vous consommerez de l'énergie

entre chaque slot uplink rx beacon et ping l'objet se met en mode stand-by

La classe C des Objets

explications centrées la modulation LoRa (RF)

- Mesures sensibles
- Lampadaires



Même principe que pour la Classe A avec la différence : dès qu'il y a eu un message uplink, la radio reste allumée pour recevoir des données en continu

La priorité sur la BP (car nous sommes en halfduplex) est sur le process lié à l'uplink

+++ L'objet sera toujours joignable

- - - ATTENTION, l'objet consommera beaucoup d'énergie

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

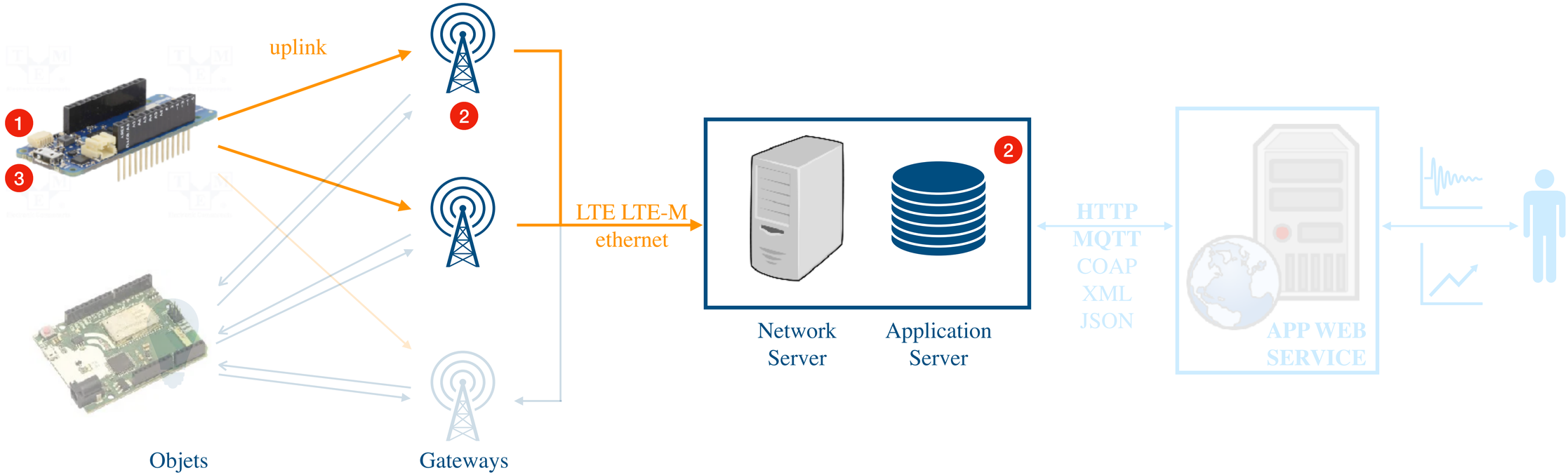
Partie

2

Activer un premier objet connecté LoRaWAN

Avec TTN

Activer un premier objet connecté LoRaWAN (message uplink)



Les étapes à suivre

1. Programmez et récupérez les informations de l'Objet, examples >> MKRWAN >> **FirstConfiguration**
2. créez et configurer (gateway aussi si inexistante) **Application** et **Enddevice** sous ttn console
3. Configurez l'objet MKRWAN 1310 (arduino) à partir des informations sous TTN >> utilisez le mode OTAA

Lancez l'ide Arduino !!!

Informations de l'objet : EUI et/ou AppEUI et/ou DevEUI

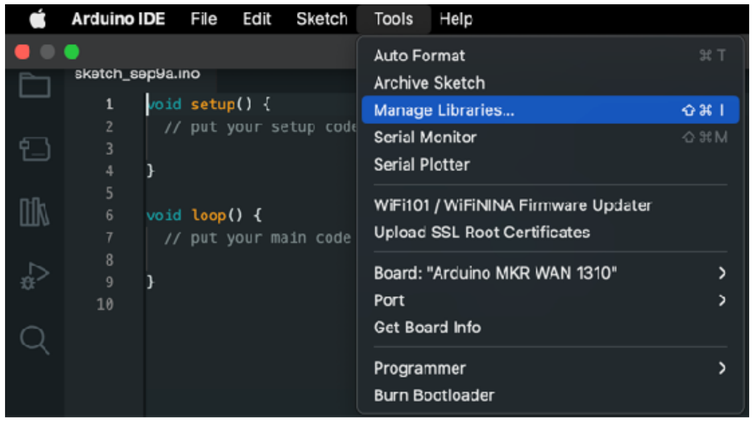
Objet

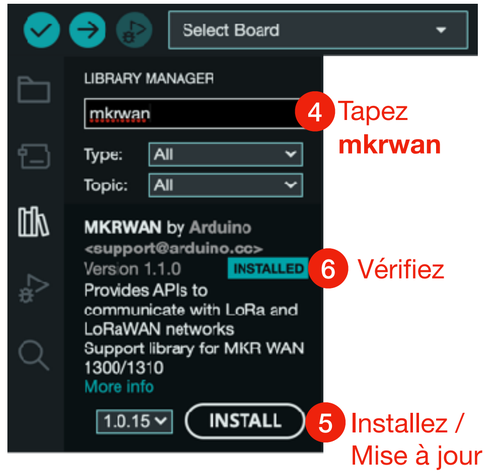
1 connectez MkrWan1310 au PC

arduino.app

lancez l'application ide arduino 2 chargement de l'application arduino ide2…

Menu > Outils > Gérer les bibliothèques 3



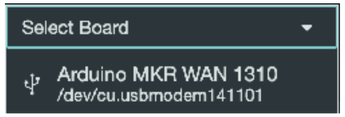


4 Tapez **mkrwan**

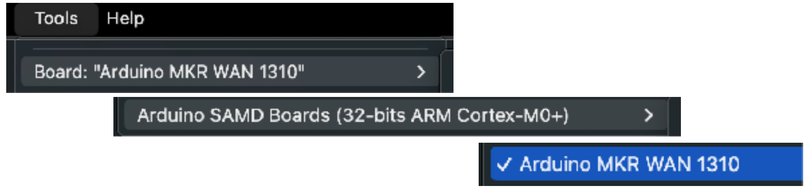
6 Vérifiez

5 Installez / Mise à jour

7 Outils > Port >

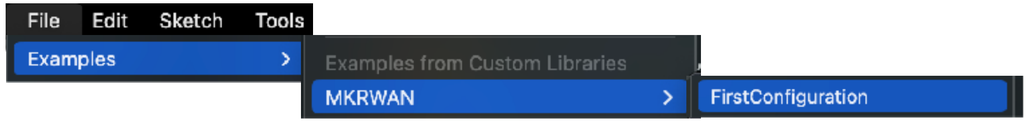


Outils > Type de carte >



Vérifiez (ou sélectionnez)

8 Fichier > Exemples > MKRWAN > FirstConfiguration

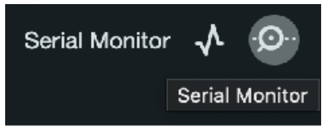


Compilez 9

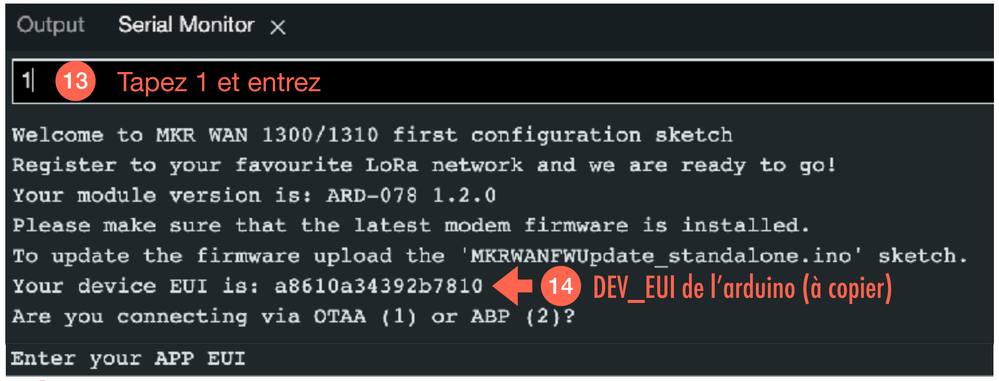
10 Téléversez

Paramètres importants

11 Lancez le terminal



12 Récupérez les informations obtenues et configurez…



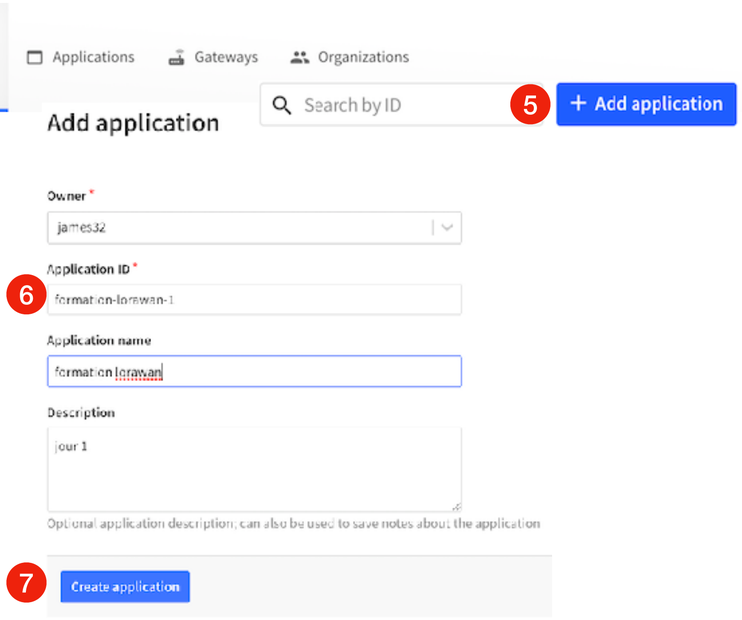
15 Rdv sous ttn pour enregistrer le device avec les informations APP EUI, APP KEY…

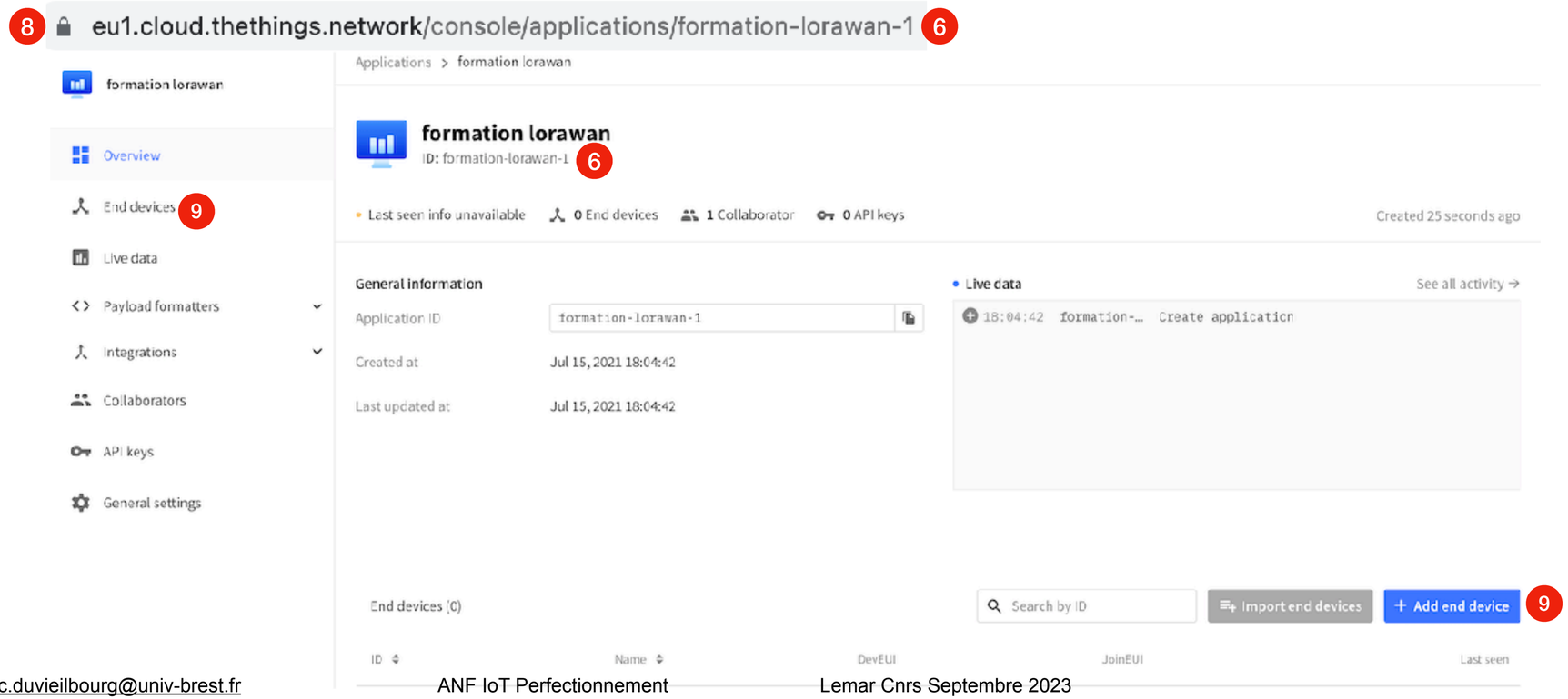
Configurer une application sur TTN 1

Login

Console

Europe1



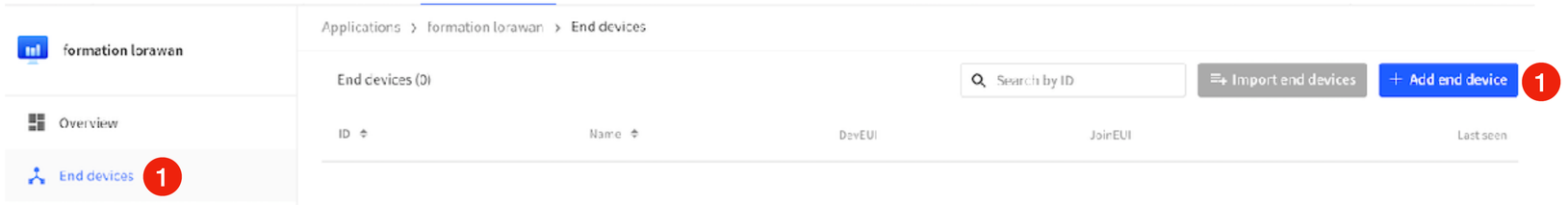


Créez un objet (End device)

Configurer un objet (End Device) sur TTN

Objet = Device = End Device

Objet = Device = End Device



End device type

Input method

Select the end device in the LoRaWAN Device Repository — Mode OTAA obligatoire

Enter end device specifics manually

End device brand *	Model *	Hardware Ver. *	Firmware Ver. *	Profile (Region) *
Arduino SA 2	Arduino MKR WA… 3	1.0 4	1.2.0 5	EU_863_870 6

Arduino MKR WAN 1310

LoRaWAN Specification 1.0.2, RP001 Regional Parameters 1.0.2, Over the air activation (OTAA), Class A

The Arduino MKR WAN 1310 is a development board that provides a practical and cost-effective solution to add LoRaWAN® connectivity for projects requiring long-range, low-power wireless communication. Sensors and actuators can be connected to the board through the analog, digital, UART, SPI, and I2C pins. The MKR WAN 1310 comes complete with an ATECC508 secure element, a battery charger, 2MByte SPI Flash, and power consumption as low as 104 uA.

Product website

Vérifiez

Frequency plan * : Europe 863-870 MHz (SF9 for RX2 - recommended) 7

Provisioning information

JoinEUI * : 00 00 00 00 00 00 00 ED

8 Libre arbitre ! Soyez imaginatif

This end device can be registered on the network

j'ai mis ici : 00 00 00 00 00 00 00 ED

DevEUI * : A8 61 0A 34 39 2B 78 10 Generate

9 collez le DEV_EUI de l'arduino

AppKey * : EE 34 62 67 5C F4 3E EF FC AF 21 F1 5F 53 34 0B Generate 10

copiez l'APP_KEY généré

End device ID * : eui-a8610a34392b7810

11 Généré automatiquement par TTN

This value is automatically prefilled using the DevEUI

mais vous pouvez le modifier !

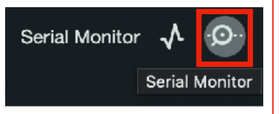
After registration

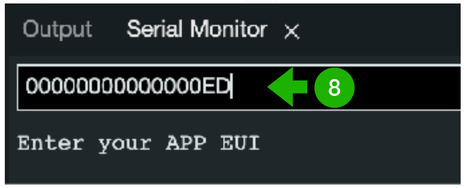
View registered end device

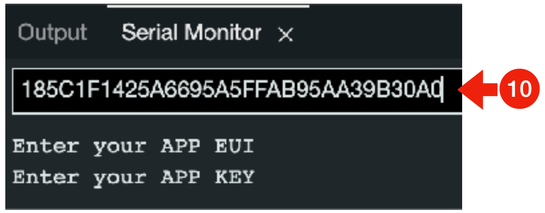
Register another end device of this type

Register end device 12 Termine la procédure d'enregistrement du device

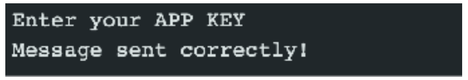
13 Retournez sur l'IDE Arduino







Tapez votre prénom et enter



Et au bout de quelques secondes ce message !

Premier objet connecté en classe A : testez le uplink et downlink

Objet = Device = End Device

Device Information

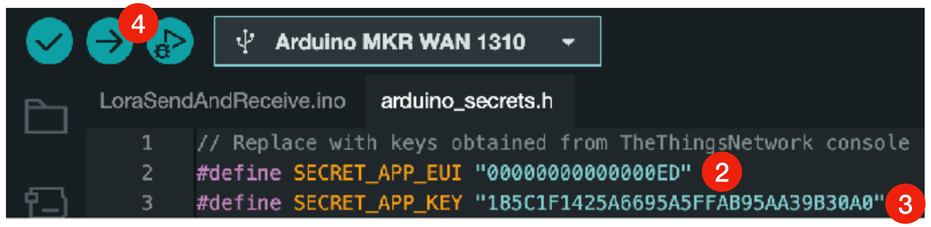
EUI: 0000000000000ED

AppEUI: 185C1F1xxxxx695A5xxxxxxxA39B30A0

DevEUI: A8610A34392B7810

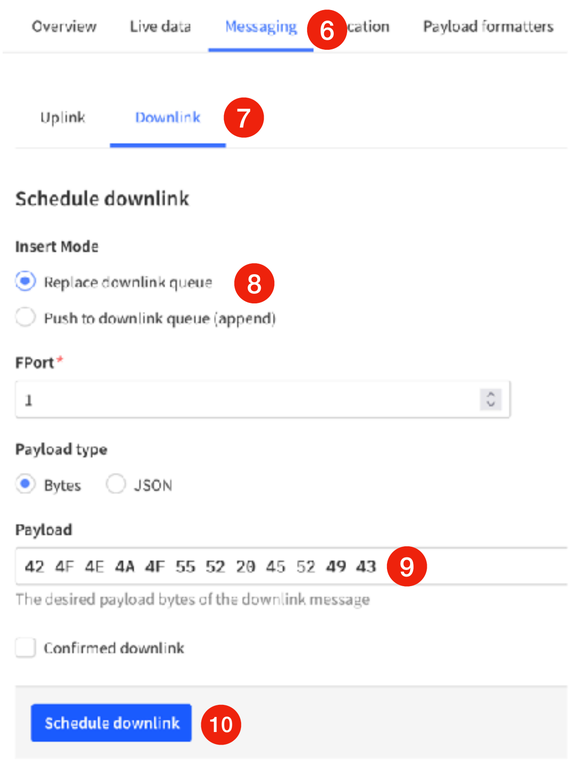
Use the EUI to register the device for OTAA

1 Sous Arduino : Fichier > Exemples > MKRWAN > LoraSendAndReceive



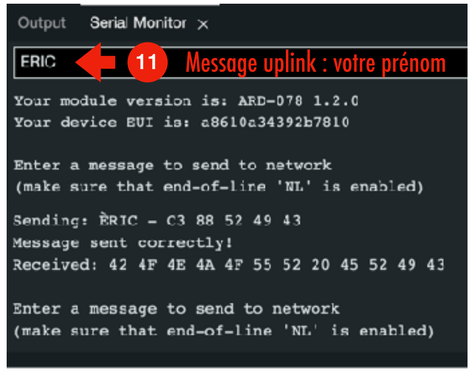
Classe A : les downlink possibles après (1s et 2s par défaut) récupèrent de l'application server les payload envoyés et les affiche 5

ttn end device > Messaging



ttn end device > Live data

Arduino terminal



Testez et Vérifiez !!!

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

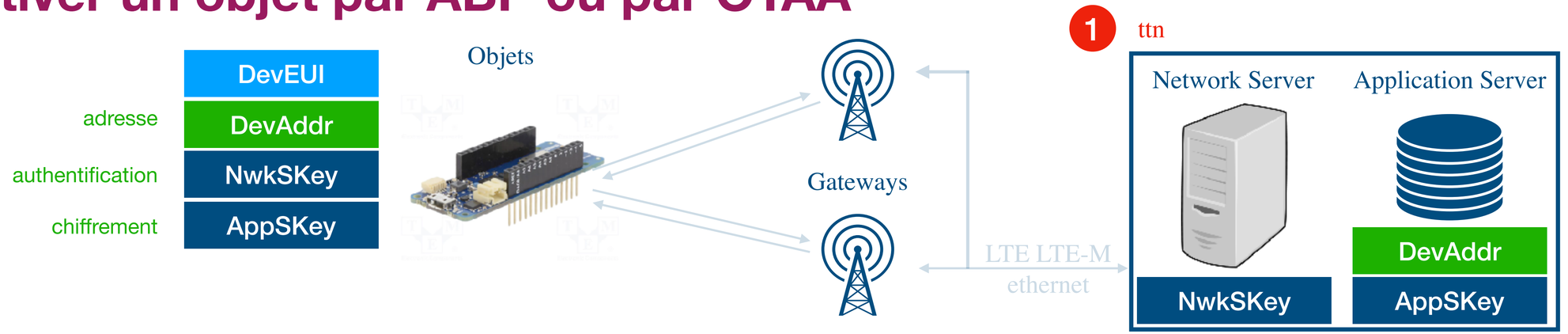
Partie

3

Configurer / comprendre

type d'activation

Activer un objet par ABP ou par OTAA



Comment activez l'objet

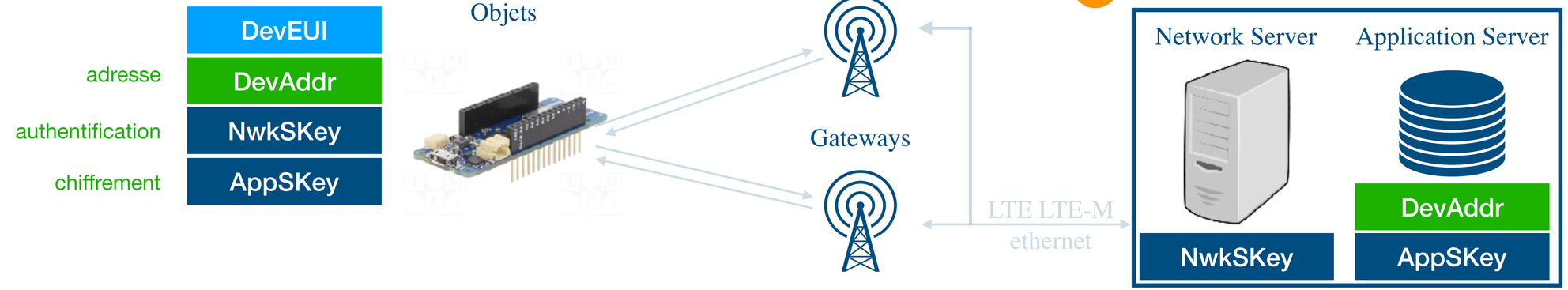
comment renseigner les même informations côté objet et côté servers (**N**etwork + **A**pplication) ?

2 méthodes :

- **OTAA** = Over The Air Activation
- **ABP** = Activation By Personalization

Objectif de ces 2 méthodes : même informations DevEUI / NwkSKey et AppSKey dans l'objet et les servers N + A

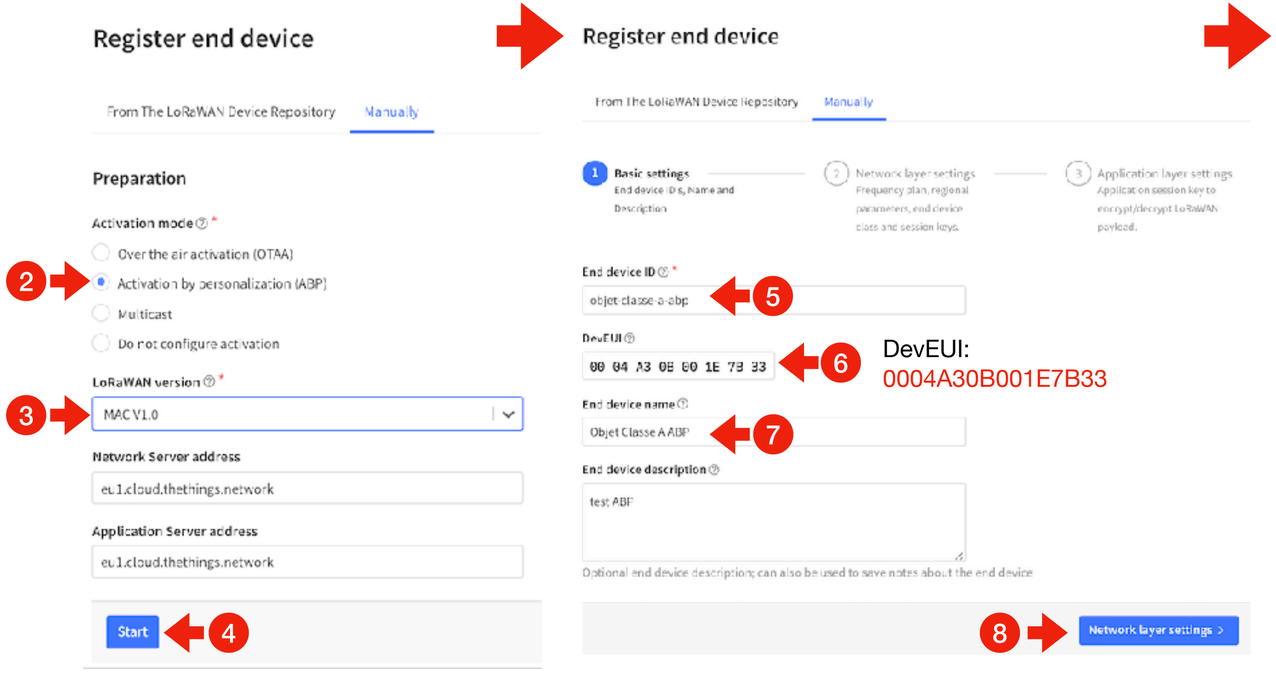
Activer un objet par ABP



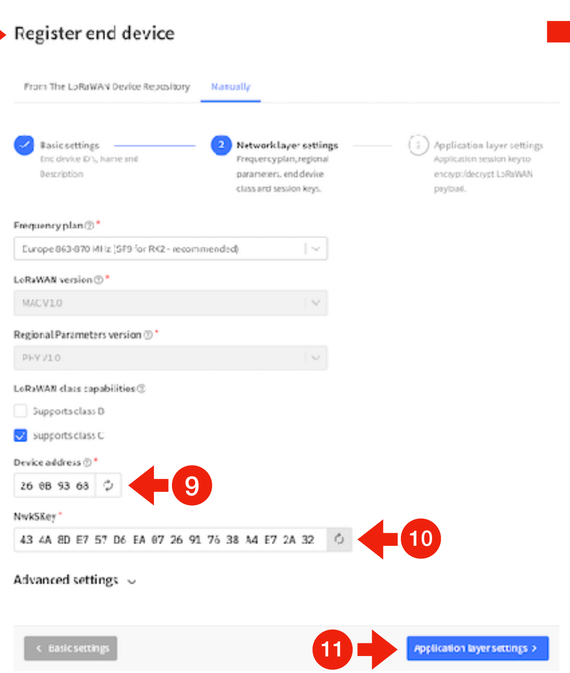
Comment un nouvel objet activé par ABP (même procédure pour chaque nouvel objet)

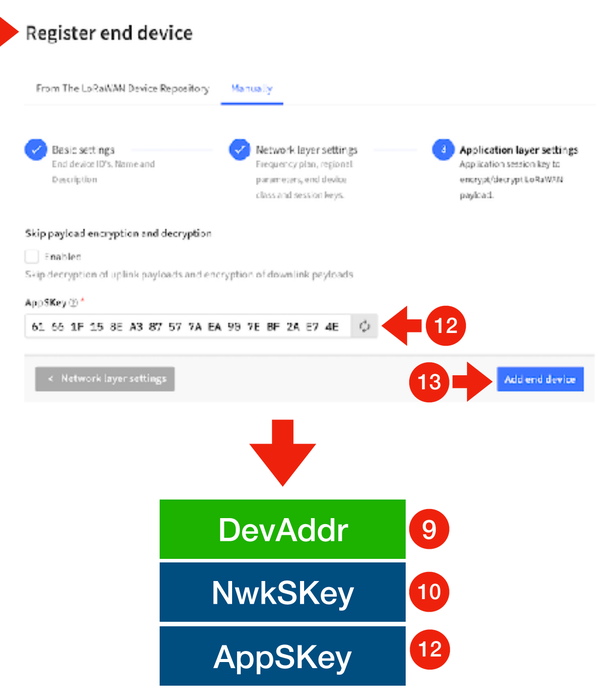
1 **sous ttn** console Applications > formation lorawan > **End devices** + Add end device

Applications > formation lorawan > End devices > **Register manually**



DevEUI:
0004A30B001E7B33





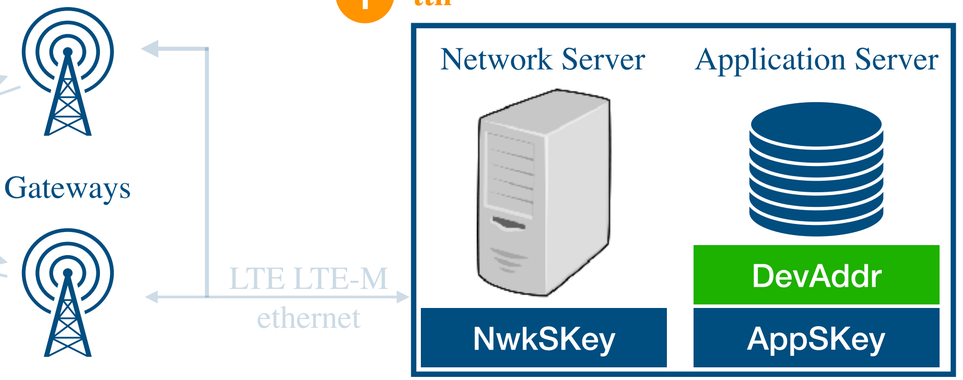
Activer un objet par ABP

2 Objets

adresse — DevAddr

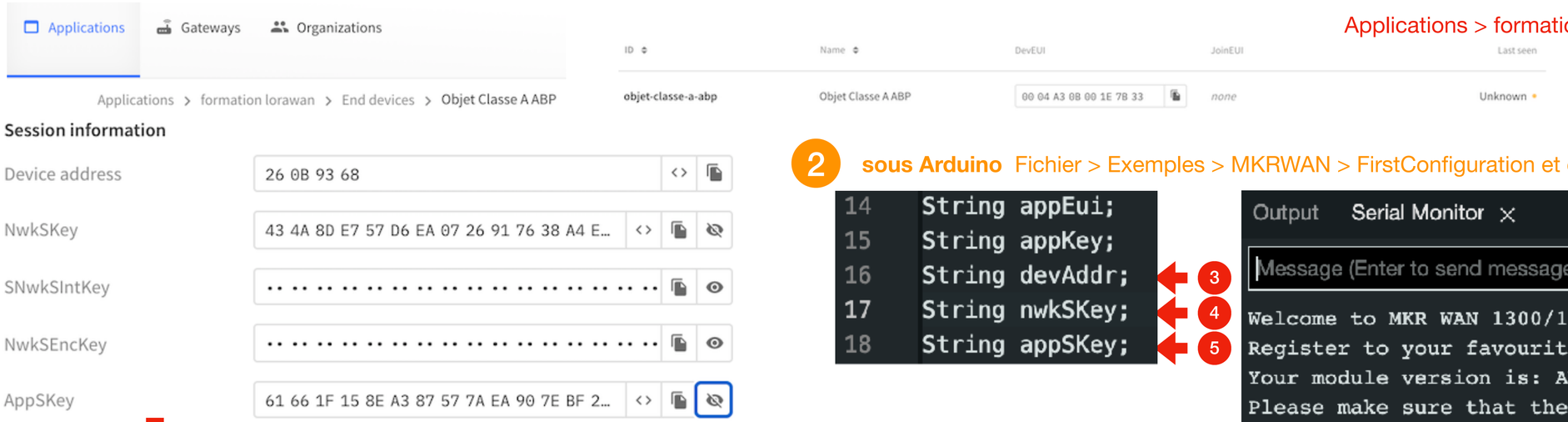
authentification — NwkSKey

chiffrement — AppSKey



Comment un nouvel objet activé par ABP (même procédure pour chaque nouvel objet)

1 **sous ttn** console Applications > formation lorawan > End devices > **Objet Classe A ABP**



Applications > formation lorawan > End devices > **Register manually**

Ne jamais dévoiler ces données

DevAddr	260B9368
NwkSKey	434A8DE757D6EA0726917638A4E72A32
AppSKey	61661F158EA387577AEA907EBF2AE74E

2 **sous Arduino** Fichier > Exemples > MKRWAN > FirstConfiguration et choisir 2 : **ABP**

```
14  String appEui;
15  String appKey;
16  String devAddr;
17  String nwkSKey;
18  String appSKey;
```

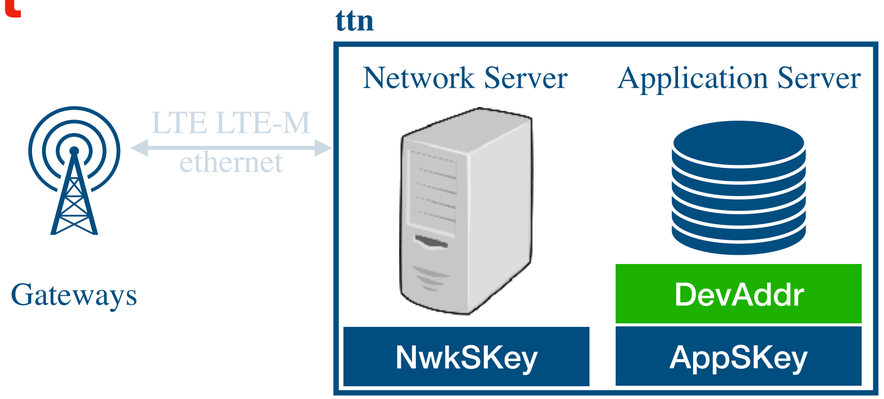
3 4 5

```
Output  Serial Monitor
Message (Enter to send message to 'Arduino MKR WAN 1310' o
Welcome to MKR WAN 1300/1310 first configuration
Register to your favourite LoRa network and we ar
Your module version is: ARD-078 1.2.0
Please make sure that the latest modem firmware i
To update the firmware upload the 'MKRWANFWUpdate
Your device EUI is: a8610a34392b7810
Are you connecting via OTAA (1) or ABP (2)?
Enter your Device Address   3
Enter your NWS KEY          4
Enter your APP SKEY         5
```

APB Live Data on ttn mise en évidence f_cnt

Applications > formation lorawan > Live data

Time	Entity ID	Type	Data preview
14:16:47	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 9.5 RSSI: -51 Bandwidth: 125000
14:16:31	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 8.5 RSSI: -47 Bandwidth: 125000
14:16:15	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 9.75 RSSI: -45 Bandwidth: 125000
14:15:59	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 8.5 RSSI: -45 Bandwidth: 125000
14:15:43	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 8.5 RSSI: -41 Bandwidth: 125000
14:15:27	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 8.75 RSSI: -45 Bandwidth: 125000
14:15:12	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 8.5 RSSI: -42 Bandwidth: 125000
14:14:55	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 7.75 RSSI: -41 Bandwidth: 125000
14:14:39	objet-classe-a-abp	Forward uplink data message	MAC payload: 00 FPort: 1 SNR: 9.5 RSSI: -51 Bandwidth: 125000



Gateways > edlemarconnect1 > Live data

Verbose stream Pause Clear

Time	Type	Data preview
14:13:51	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 17 t: 1 MAC payload: 02 Bandwidth: 125000 SNR: 10.25 RSSI: -45 Raw payload: 40 68 93 0B 26 81 11 00 08 01 02 CD 5C F4 F5
14:12:46	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 13 FPort: 1 MAC payload: C5 Bandwidth: 125000 SNR: 9.25 RSSI: -43 Raw payload: 40 68 93 0B 26 81 0D 00 08 01 C5 F3 6B 5D B0
14:12:30	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 12 FPort: 1 MAC payload: 1F Bandwidth: 125000 SNR: 8.25 RSSI: -44 Raw payload: 40 68 93 0B 26 81 0C 00 08 01 1F 1E 08 88 96
14:12:14	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 11 FPort: 1 MAC payload: 6C Bandwidth: 125000 SNR: 9.75 RSSI: -47 Raw payload: 40 68 93 0B 26 81 0B 00 08 01 6C CB FA B9 29
14:12:03	Send downlink message	Rx1 Delay: 1 Rx1 Data Rate Index: 5 Rx1 Frequency: 868100000 Rx2 Frequency: 869525000 Rx2 Data Rate Index: 3
14:11:51	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 9 FPort: 1 MAC payload: A9 Bandwidth: 125000 SNR: 8.75 RSSI: -45 Raw payload: 40 68 93 0B 26 00 09 00 01 A9 31 E3 06 3B
14:11:39	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 8 FPort: 1 MAC payload: EF Bandwidth: 125000 SNR: 9.5 RSSI: -47 Raw payload: 40 68 93 0B 26 00 08 00 01 EF 02 32 01 76
14:11:02	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 5 FPort: 1 MAC payload: F6 Bandwidth: 125000 SNR: 9.75 RSSI: -45 Raw payload: 40 68 93 0B 26 00 05 00 01 F6 C3 56 27 64
14:10:50	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 4 FPort: 1 MAC payload: BA Bandwidth: 125000 SNR: 9.75 RSSI: -45 Raw payload: 40 68 93 0B 26 00 04 00 01 BA 20 16 38 A6
14:10:38	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 3 FPort: 1 MAC payload: 56 Bandwidth: 125000 SNR: 9.75 RSSI: -43 Raw payload: 40 68 93 0B 26 00 03 00 01 56 F8 07 DD 83
14:10:26	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 2 FPort: 1 MAC payload: 27 Bandwidth: 125000 SNR: 8.5 RSSI: -41 Raw payload: 40 68 93 0B 26 00 02 00 01 27 B9 A2 ED 65
14:10:14	Receive uplink message	DevAddr: 26 0B 93 68 FCnt: 1 t: 1 MAC payload: F4 Bandwidth: 125000 SNR: 7.5 RSSI: -41 Raw payload: 40 68 93 0B 26 00 01 00 01 F4 E5 4C 4C C4

Activer un objet par OTAA Over The Air Activation (même finalité que ABP : DevAddr + NwkSKey + AppSKey)

2 ttn-mkrwan

config avant le Join-Request

AppEUI

AppKey

DevEUI

End Nodes LoRa

Gateways

1 ttn

Network Server Application Server

config avant le Join-Request

AppEUI id unique de l'appli.

AppKey clé pour générer les autres clés

DevEUI généré ici ou récupéré du Device

Comment fonctionne la création des DevAddr / NwkSKey / AppSKey via la méthode OTAA (même procédure pour chaque nouvel objet)

le Join-Request est le premier message uplink envoyé par le End Node LoRa. Il va déclencher la génération des DevAddr NwkSKey AppSKey côté servers

End Node LoRa

AppEUI

AppKey

DevEUI

1 Join-Request : AppEUI MIC(AppKey) DevEUI

Network Server

2 authentifié validé génération de

DevAddr

NwkSKey

AppSKey

Application Server

AppEUI

AppKey

DevEUI

4 Join-Accept : DevAddr + parameters

3

DevAddr

+ parameters génération dans End Node des

NwkSKey

AppSKey

Activer un objet par OTAA Over The Air Activation (même finalité que ABP : DevAddr + NwkSKey + AppSKey)

2 ttn-mkrwan

Device Information
EUI: 0004A30B001E7B33
AppEUI: 0000000000000000
DevEUI: 0004A30B001E7B33
Use the EUI to register
the device for OTAA

config avant le Join-Request

AppEUI
DevEUI
AppKey

End Nodes LoRa

Gateways

1 ttn

Network Server Application Server

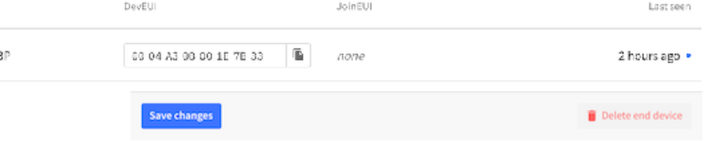
config avant le Join-Request

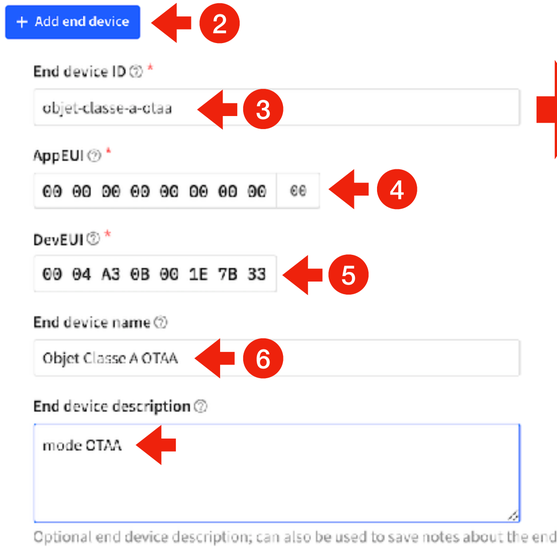
AppEUI id unique de l'appli.
DevEUI généré ici ou récupéré du Device
AppKey clé pour générer les autres clés

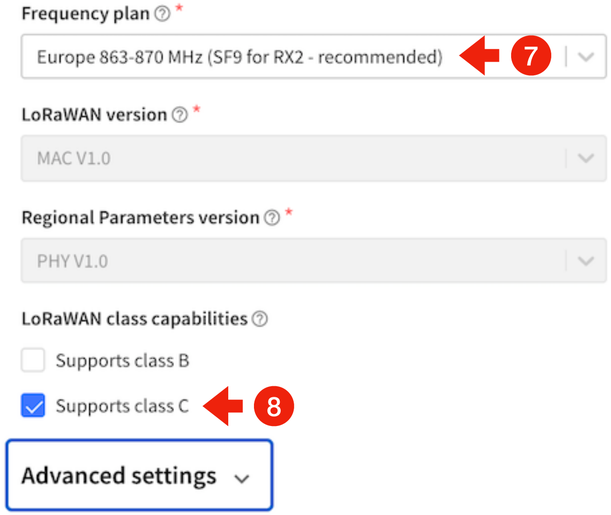
Comment créer un nouvel objet activé par OTAA (même procédure pour chaque nouvel objet)

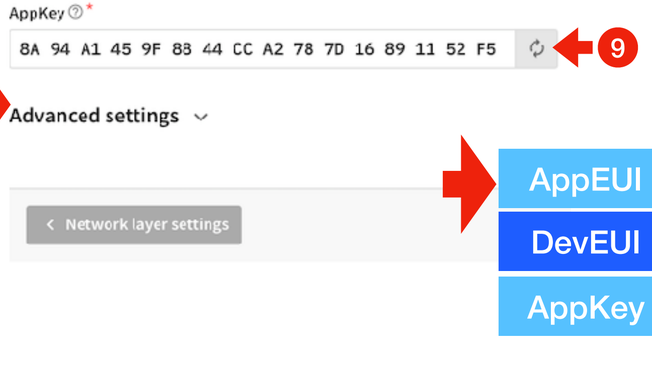
1 **sous ttn** console Applications > formation lorawan > **End devices**

Applications > formation lorawan > End devices > Register from The LoRaWAN Device Repository









AppEUI 0000000000000000
DevEUI 0004A30B001E7B33
AppKey 8A94A1459F8844CCA2787D16891152F5

Ne jamais dévoiler ces données

2 ttn-uno Arduino > Fichier > Exemple > TTN

Essayez de configurer le End Device et ensuite retournez sur ttn live data :
Repérez le Join-Accept

ABP ou OTAA ? Live Data on ttn mise en évidence

1 Sécurité OTAA == ABP

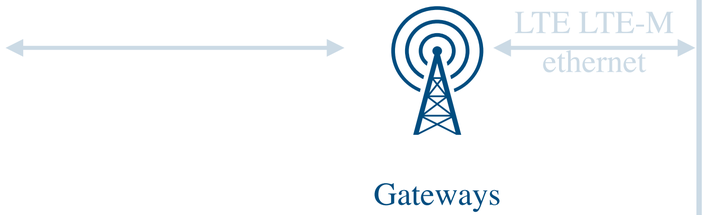
clé(s) stockée(s) en permanence (mem sécurisée)

	AppKey	OTAA
NwkSKey	AppSKey	ABP

clé sécurité fail possible

attention revers engineering ou bus prtcl sur le circuit

End Nodes LoRa



2 Frame Counter : **f_cnt** sous ttn V3 éviter les attaques par replay compteurs End Devices et Servers dépend cas ABP

NwkSKey

AppSKey

clés de chiffrement et d'authentification quasi inviolable

hacker qui écoute la trame raw payload ne peut pas comprendre la partie utile

exemple ouverture porte de garage même si on ne comprend pas on imagine qu'en rééméttant plus tard on pourra ouvrir la porte (sans les keys)

méthode : EndDevice n'émet jamais la même trame via l'incrémentation d'un compteur qui côté serveur s'incrément aussi à chaque uplink

Uplink	f_cnt_dev	receive_srv	f_cnt_srv	f_cnt_dev >= f_cnt_srv	valid uplink ?
1	0	OK	0	Oui	OK
2	1	OK	1	Oui	OK
3	2	X	X	X	X
4	3	OK	2	Oui	OK
1 (reset)	0	OK	3	Non	X

attaques par replay détectée = srv non valide

il faudra attendre 4 uplink avant que ça soit valide !!!

solution 1 : désactiver le f_cnt mais pas conseillé (sauf en mode développement => vérifier que cette option soit dispo)

2 : passer en mode OTAA à chaque Join-Request le server remet aussi son f_cnt à 0

3 : stocker dans sa mem non-volatile le f_cnt du End Device (lecture après un reset)

il y aussi des f_cnt pour les downlink

prob = récurent >> DEMO sur ttn dans la page gateway

1 ABP

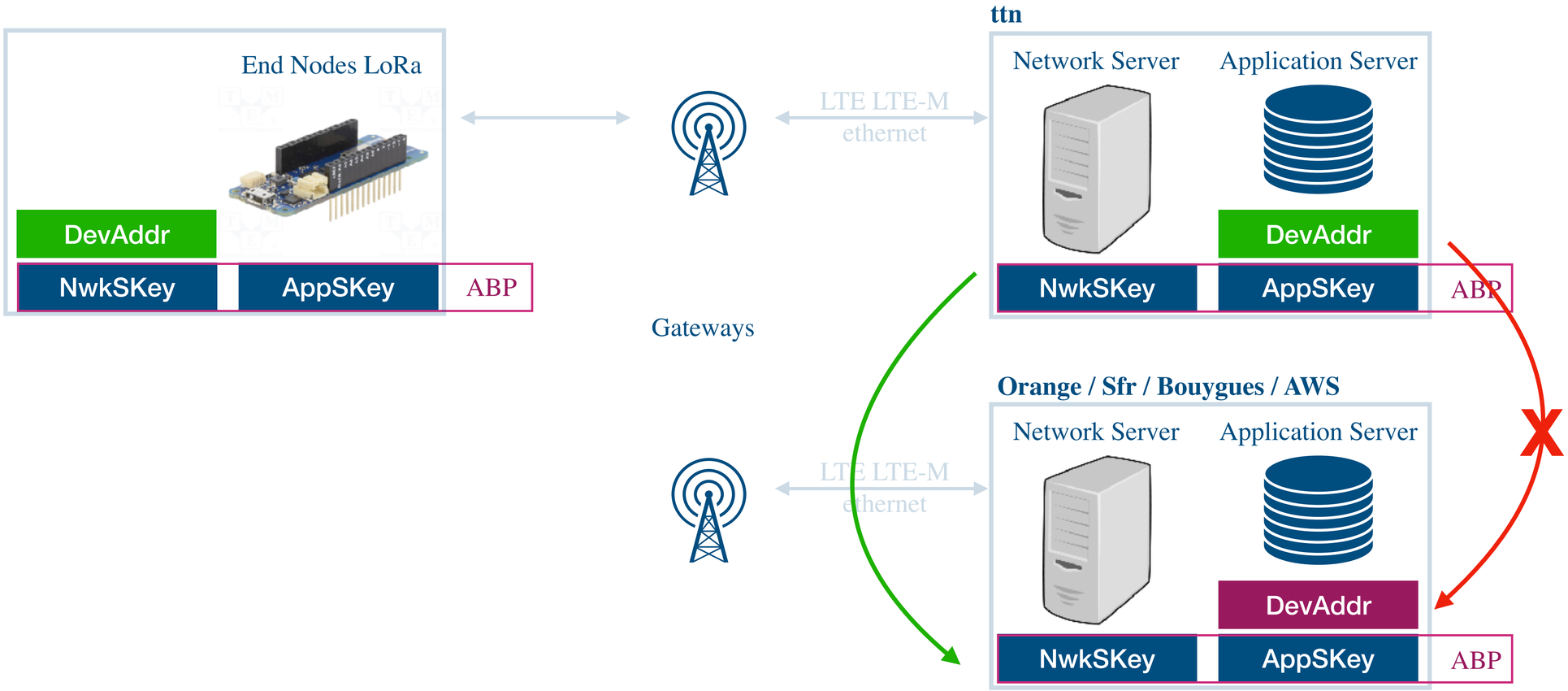
2 OTAA

https://v2console.thethingsnetwork.org/applications/ed29iot112018_1/devices/arduino_mkr_wan_1300_ed_iot

ABP ou OTAA ?

❸ Changement de réseau

End Nodes déjà déployés
Impossible de modifier



NwkSKey et AppSKey peuvent se copier mais pas DevAddr

DevAddr est générer par un processus interne aux servers
TTN les **DevAddr** commencent par le préfixe **0x26**
Les DevAddr des autres fournisseurs auront un autre préfixe

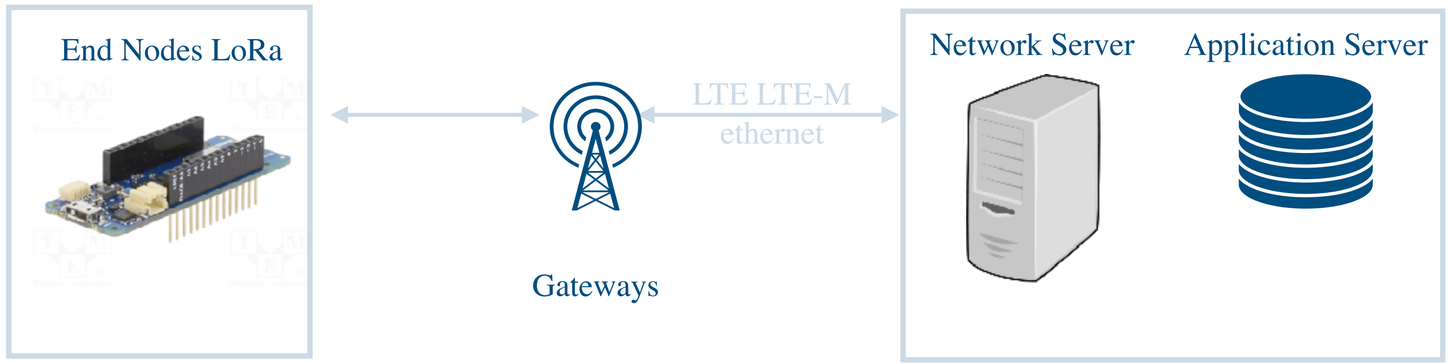
On peut pas configurer les End Nodes et les servers avec les mêmes Adresses

En OTAA les NwkSKey | AppSKey | DevAddr **sont régénérer et transmis au End Node => pas de souci**

OTAA plus adapté ici

ABP ou OTAA ?

4 Modification de paramètres

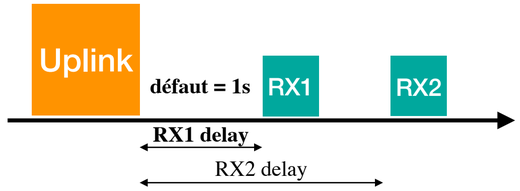


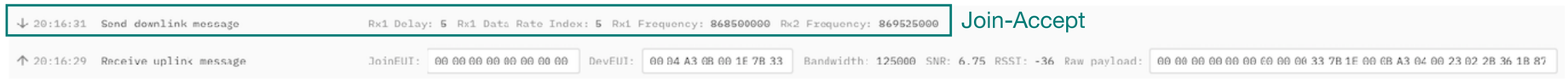
En ABP dès l'allumage du End Node les informations peuvent directement être transférées aux serveurs

En OTTA dès l'allumage du End Node il lancera une procédure de Join-Request qui sera suivi d'une procédure Join-Accept

Size (bytes)	3	3	4	1	1	(16) Optional
Join Accept	AppNonce	NetID	DevAddr	DLSettings	RxDelay	CFList

La trame Join-Accept peut transmettre des paramètres ex : "rx1_delay": 5, "rx1_data_rate_index": 5, "rx1_frequency": « 868500000", "rx2_frequency": "869525000"





La CFList : optional channel frequency list (p36 LoRaWAN Spec)

Canal (MHz)
868,10
868,30
868,50
Freq Ch4
Freq Ch5
Freq Ch6
Freq Ch7
Freq Ch8

p35

3 canaux connus obligatoirement par un End Node

configurable

En ABP ces 5 canaux sont préenregistrés dans le End Node

En OTAA ces 5 canaux sont programmables (24bits)

Size (bytes)	3	3	3	3	3	1
CFList	Freq Ch4	Freq Ch5	Freq Ch6	Freq CN7	Freq Ch8	RFU
		864,3				

Demo CFList

4 Modification de paramètres

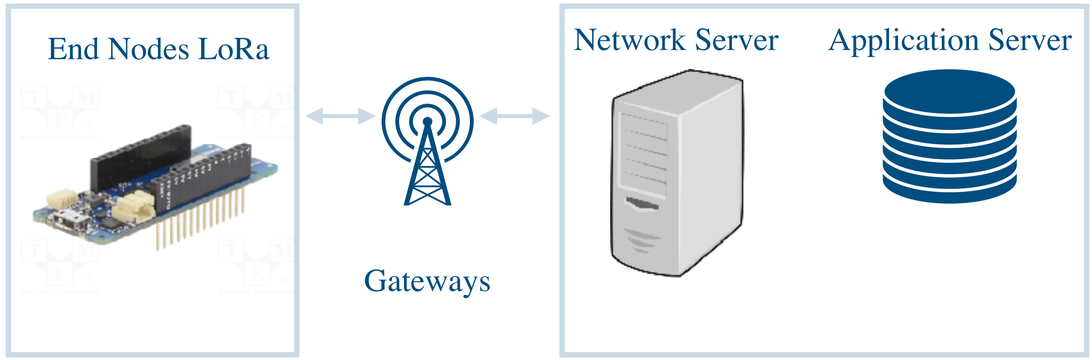
En mode ABP si vous utilisez un End Node :
- générique seules 3 fréq sont connues : **868,1 868,3 868,5 MHz**

- conçu pour des servers spec, les 5 fréq configurables seront codées en dur

exemple TTNUNO 867,1 867,3 867,5 867,7 867,9 **868,1 868,3 868,5**

```
void TheThingsNetwork::configureEU868()
{
  sendMacSet(MAC_RX2, "3 869525000");
  sendChSet(MAC_CHANNEL_DRRANGE, 1, "0 6");

  char buf[10];
  uint32_t freq = 867100000;
  uint8_t ch;
  for (ch = 0; ch < 8; ch++)
  {
    sendChSet(MAC_CHANNEL_DCYCLE, ch, "799");
    if (ch > 2)
    {
      sprintf(buf, "%lu", freq);
      sendChSet(MAC_CHANNEL_FREQ, ch, buf);
      sendChSet(MAC_CHANNEL_DRRANGE, ch, "0 5");
      sendChSet(MAC_CHANNEL_STATUS, ch, "on");
      freq = freq + 200000;
    }
  }
  sendMacSet(MAC_PWRIDX, TTN_PWRIDX_EU868);
}
```




```
int good = 0;
for(int i=0; i<10; i++) {
    delay(500);
    if (!modem.available()) {
      Serial.print("No downlink message received at this time. try ");
      Serial.println(i);
    }
    else good = 1;
}
if(!good) {
  return;
}
```

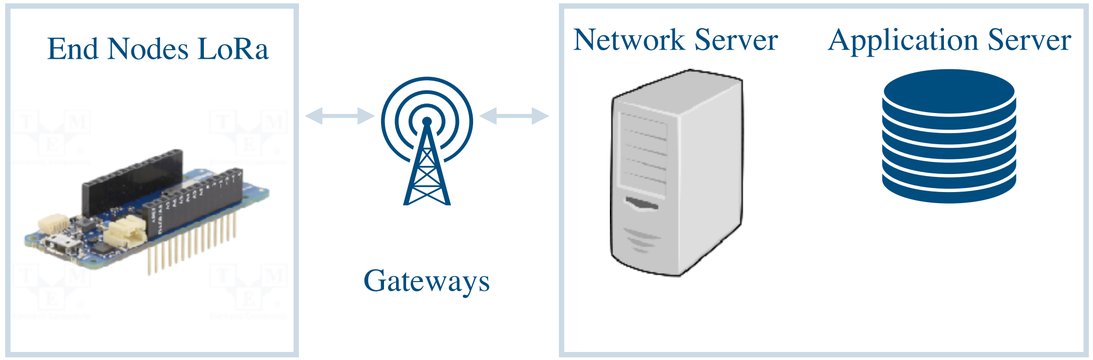
En OTTA dès l'allumage du End Node il lancera une procédure de Join-Request qui sera suivi d'une procédure Join-Accept (cryptée par l'AppKey)

C'est en analysant le (physical) payload du Join-Accept qu'on pourra en savoir plus, le Join-Accept utilisera forcément l'une des 3 fréquences obligatoires

```
"data": {
  "@type": "type.googleapis.com/ttn.lorawan.v3.DownlinkMessage",
  "raw_payload": "IEePoeC0VnhF8yb3aVTA9hWZ2sC4KRkfO40DzFgVLOsA",
```

Il faut utiliser un décodeur de trame LoRaWAN https://lorawan-packet-decoder-0ta6puiniaut.runkit.sh/

ABP ou OTAA ?



Bilan

	ABP	OTAA
Sécurité (fw)	NwkSKey AppSKey	AppKey
Frame Counter : f_cnt	gérer la mem volatile désactivation possible (Att. Replay)	oui par le Join-Accept
Autres serveurs (Net. App)	Impossible : DevAddr	oui par le Join-Accept
RX Delay param	non	oui par le Join-Accept
configuration des 5 canaux	non ou par firmware	oui par le Join-Accept

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

Partie

3 **Configurer / comprendre**

ADR

Adaptive Data Rate

End Nodes LoRa — LoRa — Gateways

Paramètres jouables
SF et Pe : meilleur couple

Image énergie perdu :
Pe élevé : personne qui hurle
SF élevé : personne qui articule trop

Si plusieurs personnes idéal c'est que les couples chuchotent entre eux pour garantir le dialogue entre les couples

Méthode 1 : empirique

1 Pe
SF
uplink

3 Pe ++
SF ++

5 Pe —
SF ++

n fois jusqu'à obtenir le meilleur couple Pe SF
méthode impossible sur
- une large flotte de End Node
- dès qu'un End Node change de place …

2 RSSI : mesuré et stocké > Sensibilité
SNR calculé > SNR min

4 RSSI : mesuré et stocké
SNR calculé

6 RSSI : mesuré et stocké
SNR calculé

Méthode 2 : adaptative

1 (Pe, SF)

uplink + ADR enabled

2 RSSI : mesuré
SNR calculé

3 Network Application Server

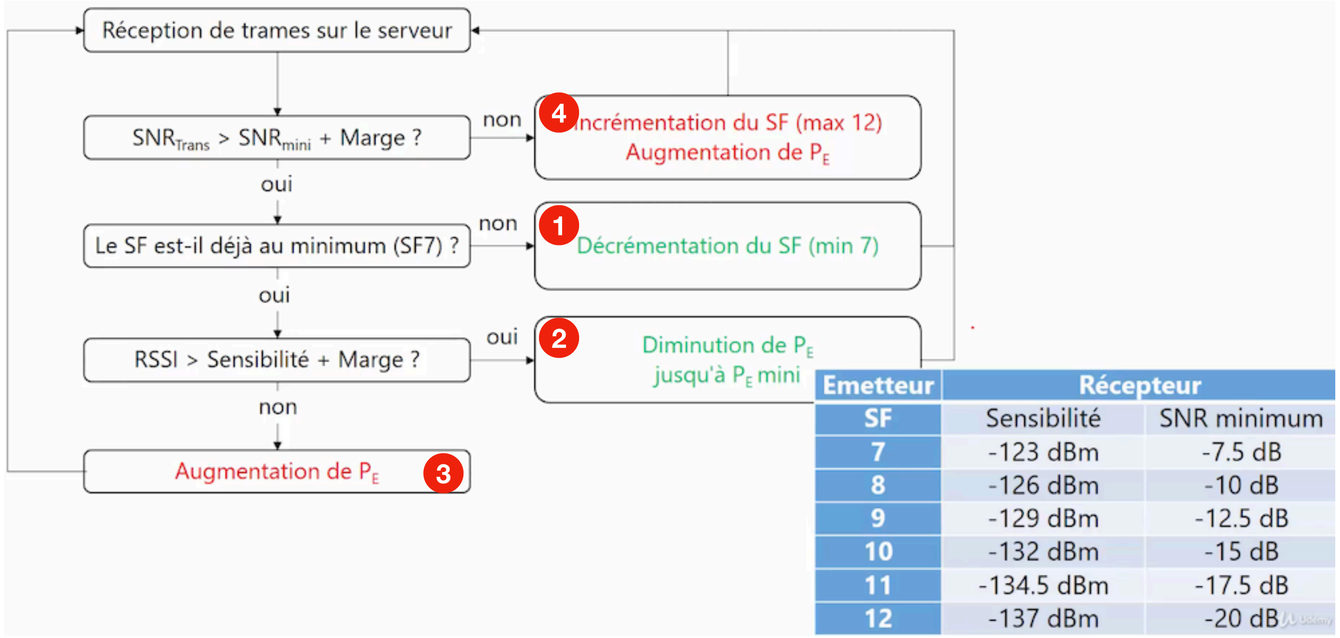
calcul de marge = f(sensibilité, SNR min)

4 downlink : Link ADReq
x N pour converger vers un bon résultat
Network server

5 adaptation du (Pe, SF)

uplink + ADRACKReq si EndNode à l'origine de la demande

Adaptive Data Rate



Emetteur	Récepteur	
SF	Sensibilité	SNR minimum
7	-123 dBm	-7.5 dB
8	-126 dBm	-10 dB
9	-129 dBm	-12.5 dB
10	-132 dBm	-15 dB
11	-134.5 dBm	-17.5 dB
12	-137 dBm	-20 dB

Démo : placer le End Node près de la Gateway
le End Node démarre tjs avec la Pe max et un SF de 9

1 2

Démo : dégrader la transmission : on débranche la gateway

3 4

Canaux	Spreading Factor	Bandwidth
868.1 Mhz	De SF7 à SF12	125 kHz
868.3 Mhz	De SF7 à SF12	125 kHz
868.3 Mhz	SF7	250 kHz
868.5 Mhz	De SF7 à SF12	125 kHz
867.1 Mhz	De SF7 à SF12	125 kHz
867.3 Mhz	De SF7 à SF12	125 kHz
867.5 Mhz	De SF7 à SF12	125 kHz
867.7 Mhz	De SF7 à SF12	125 kHz
867.9 Mhz	De SF7 à SF12	125 kHz

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

TX Power Index	Valeur en dBm
1	14 dBm
2	11 dBm
3	8 dBm
4	5 dBm
5	2 dBm

SF directement, Pe toutes les 20 trames

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

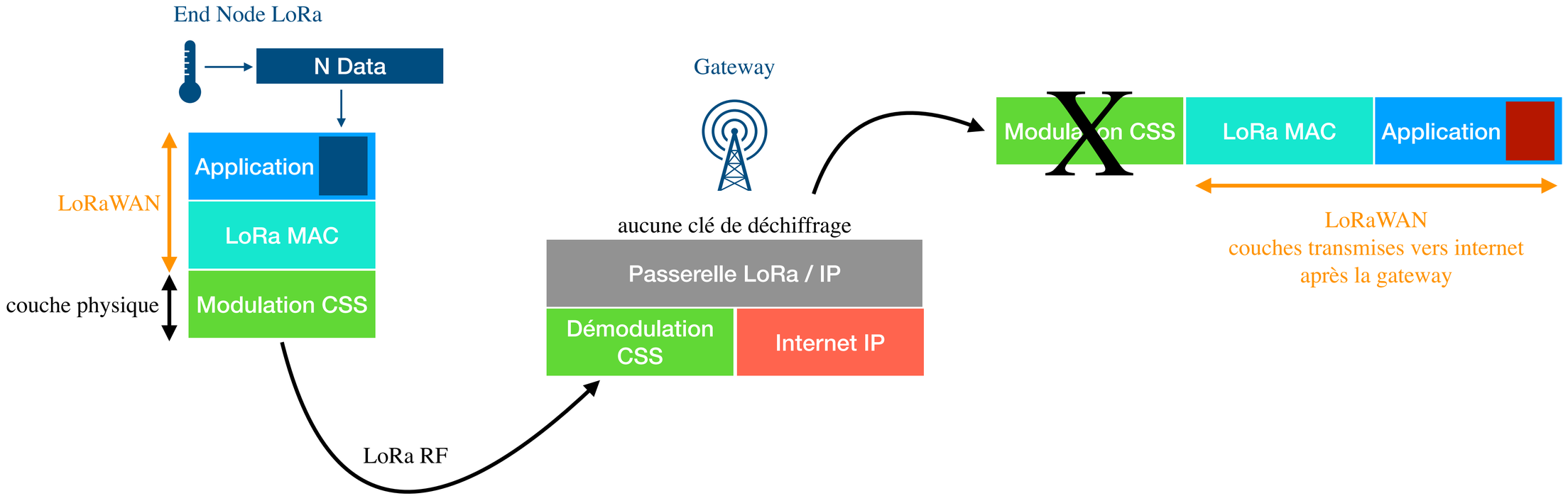
Partie

4

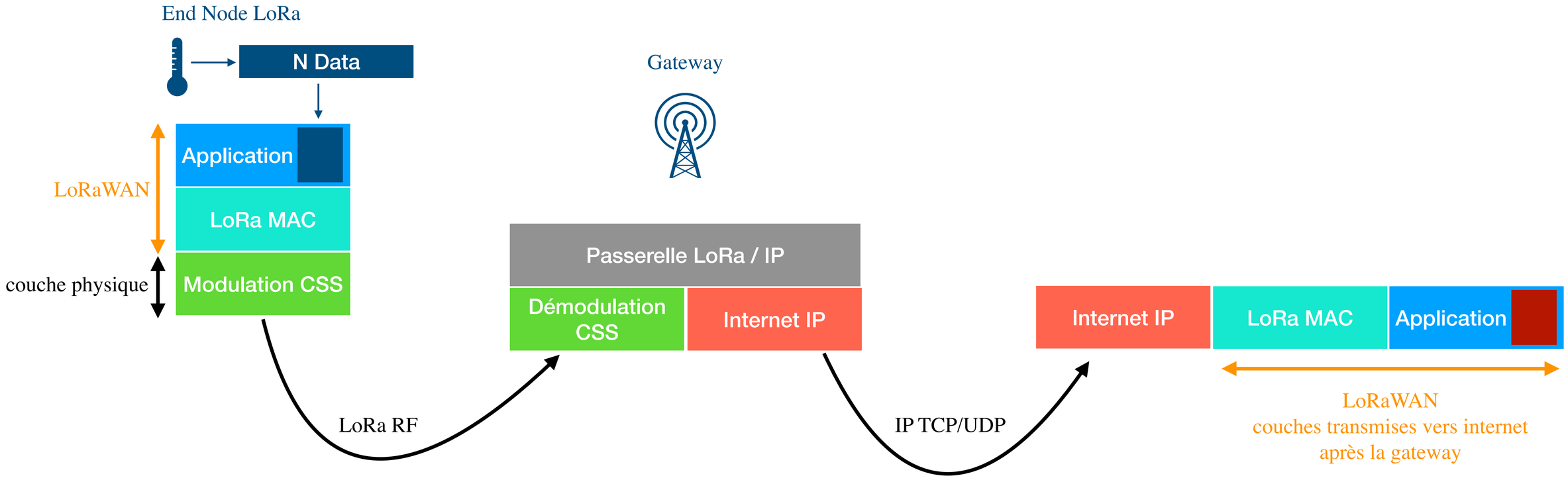
Analyser et optimiser les trames LoRaWAN

Packet Forwarder

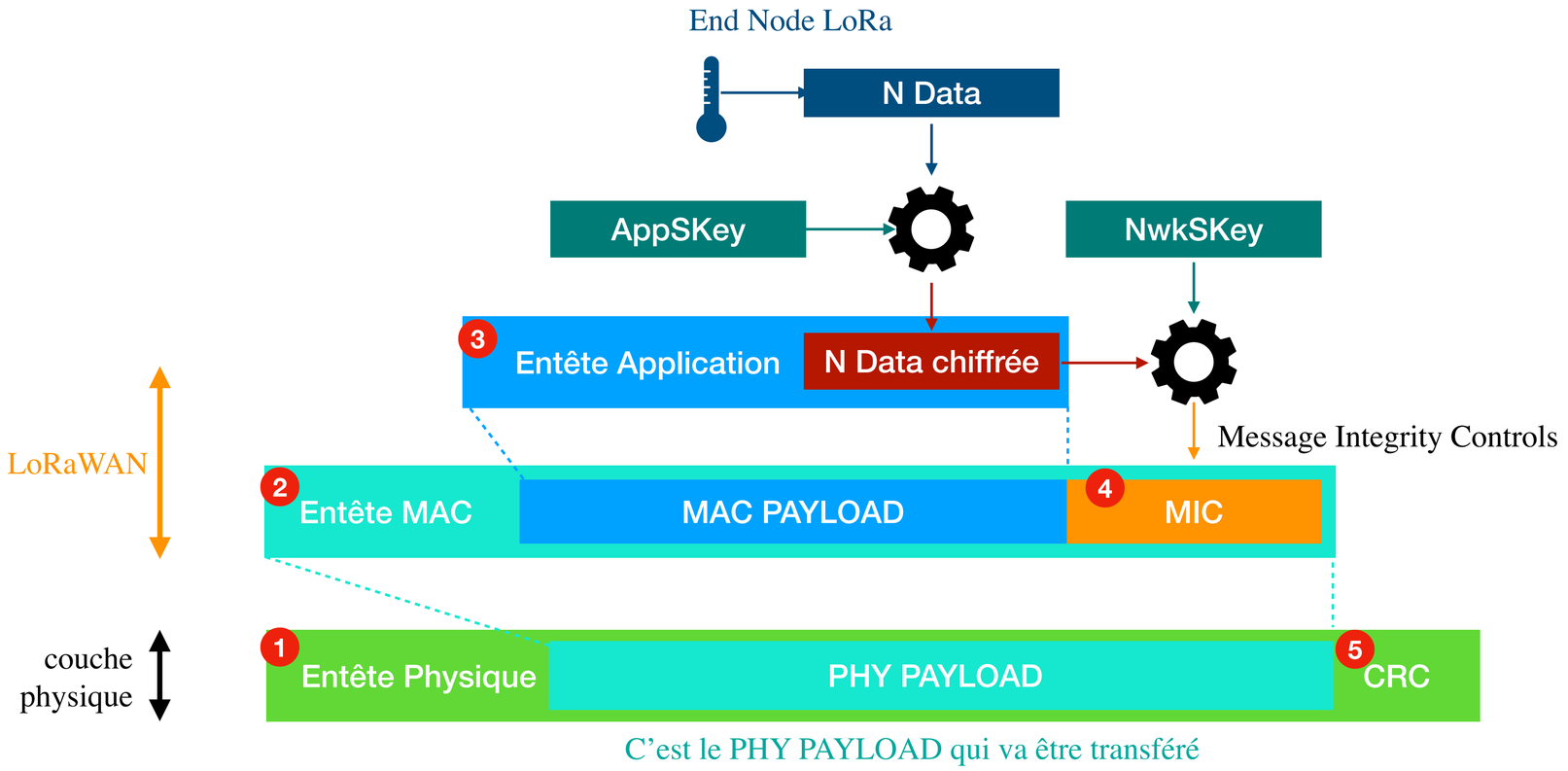
Analyser et optimiser les packets LoRaWAN



Analyser et optimiser les packets LoRaWAN



Analyser et optimiser les packets LoRaWAN



Architecture de Description de **LoRa**WAN avec The Things **N**etwork

Partie

4

Analyser et optimiser les trames LoRaWAN

Format payload : décoder, convertir, valider

Analyser et optimiser les packets LoRaWAN, décoder le Payload…

Si la donnée utile est codée sur 1 octet, le PhyPayload sera sur 14 octets min

```
↑ 18:47:08  Forward uplink data message   Payload: { bytes: […], len: 1, msg: "toto" } 01  FPort: 1  SNR: 9.25  RSSI: -33  Bandwidth: 125000
```

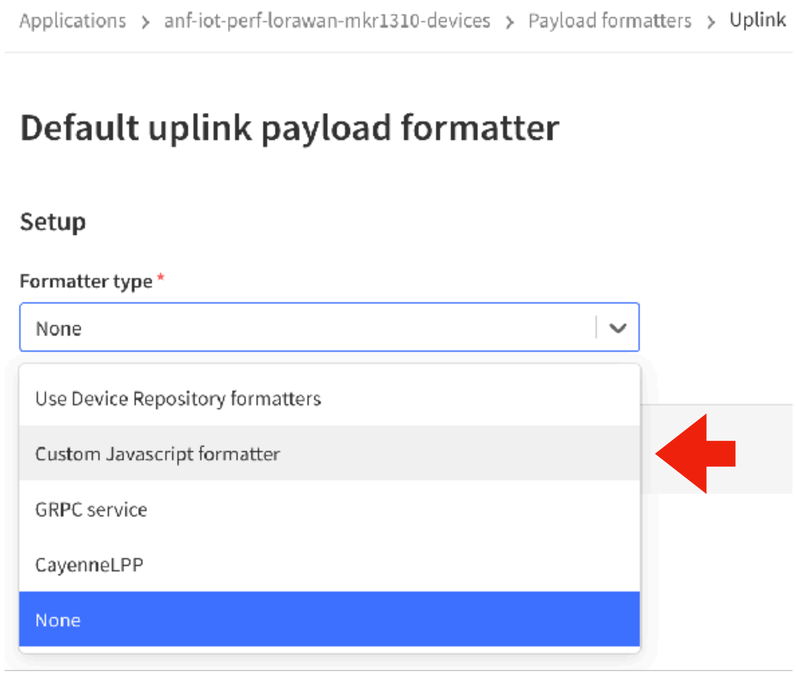
Pour exploiter les données issues des messages (uplink) on doit les décoder et le reformater !

Ce traitement de décodage / formatage du payload peut être fait en dehors d'un serveur LoRaWAN !

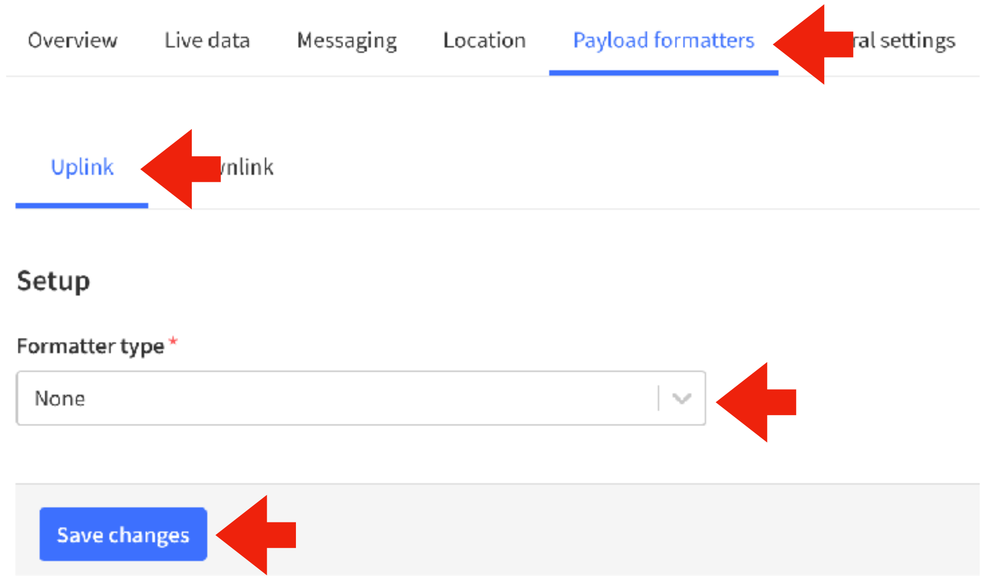
Les meilleurs serveurs LoRaWAN sur le marché proposent dans leur serveur d'application (Application Serveur) déjà des solutions, c'est le cas de TTN !!!

Comment accéder à cette fonctionnalité sous TTN :

Cas pour tous les messages uplink de tous vos Devices dans une application:



Cas pour tous les messages uplink d'un seul dévice dans une application:

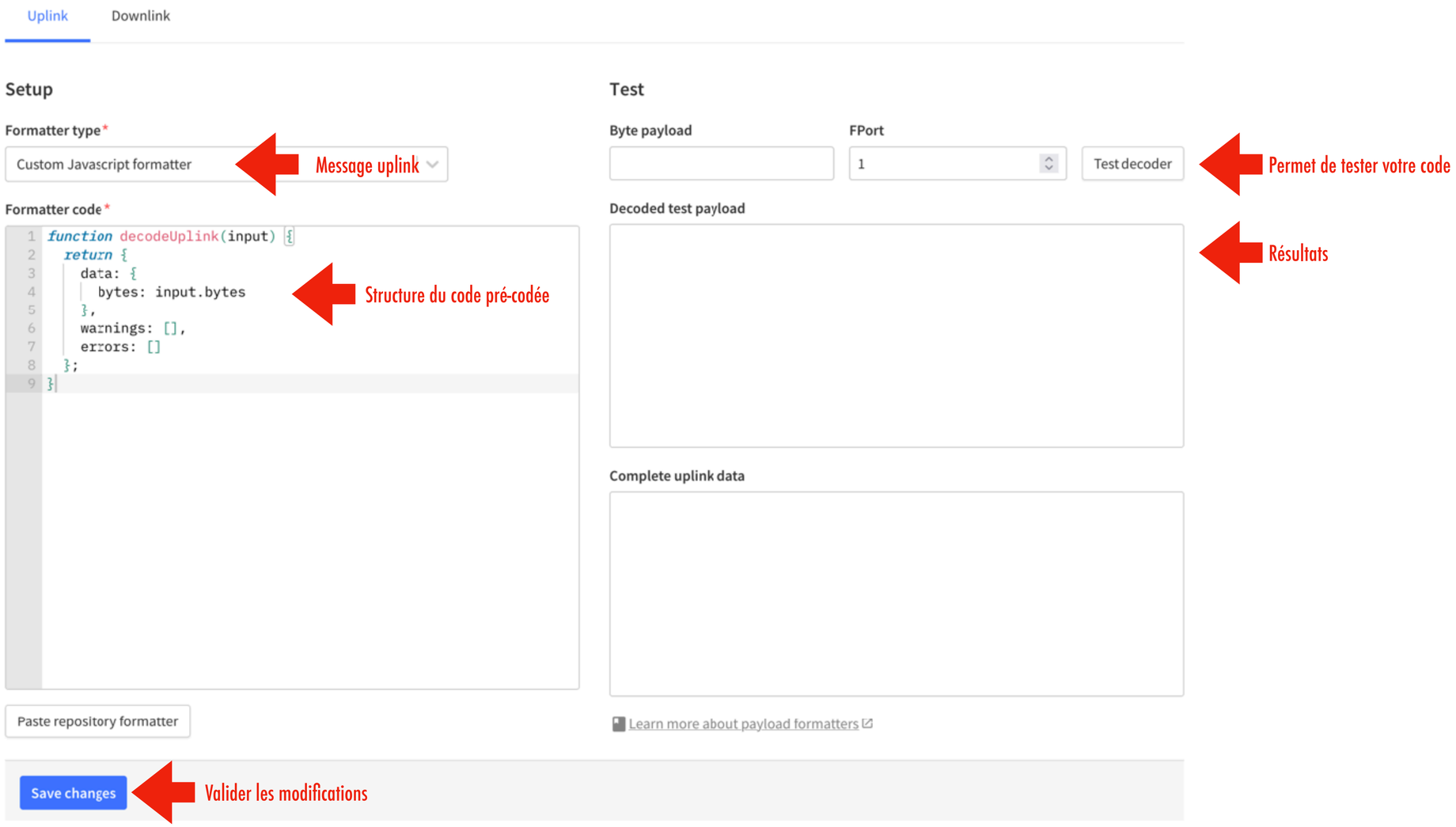


Le même principe s'appliquera pour les messages downlink

Analyser et optimiser les packets LoRaWAN, décoder le Payload…

Sous TTN les payload formatter peuvent déjà être soumis à des fonctions pré-codées afin d'attaquer des services existants (CAYENNE LPP) …

Mais vous pouvez vous même définir votre propre fonction dans le langage JavaScript, en plus vous pouvez tester directement votre code sous la même page !!!



Analyser et optimiser les packets LoRaWAN, décoder…

Exemples de codes JavaScript

```
function decodeUplink(input) {
  var data = {};
  var events = {
    1: "setup",
    2: "interval",
    3: "motion",
    4: "button"
  };
  data.event = events[input.fPort];
  data.battery = (input.bytes[0] << 8) +
input.bytes[1];
  data.light = (input.bytes[2] << 8) +
input.bytes[3];
  data.temperature = (((input.bytes[4] & 0x80
? input.bytes[4] - 0x100 : input.bytes[4]) <<
8) + input.bytes[5]) / 100;
  var warnings = [];
  if (data.temperature < -10) {
    warnings.push("it's cold");
  }
  return {
    data: data,
    warnings: warnings
  };
}
```

```
function decodeUplink(input) {

 return {
  data: {
   msg: 'toto',
   bytes: input.bytes,
   val: JSON.stringify(input.bytes),
   len: input.bytes.length
  },
  warnings: [],
  errors: []
 };
}
```

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

Partie

4

Analyser et optimiser les trames LoRaWAN

Exercice TP : réalisez un code Arduino et un décodage JavaScript TTN

Créez votre code côté Arduino

A partir de l'exemple LoraSendAndReceive.ino sous Arduino (mais vous pouvez vous inspirez d'autres exemples...

1 - Créez 3 variables pour l'acquisition du temps, de la température et de l'humidité (vous pouvez utiliser une structure)

```
uint32_t timestamp;  // 4 bytes
int16_t temperature;  // 2 bytes
uint8_t humidity;  // 1 byte
```

2 - Générez des données, aléatoires pour la température et l'humidité

Inspirez-vous de la fonction **random(-200, 400) ;** // ici on renvoie une valeur entre 200 et 399
Notez qu'il faut éviter d'envoyer des nombres flottants sur LoRaWAN pour éviter des pertes de précision
Pour le temps vous pouvez utilisez la fonction **millis**

3 - Faites une acquisition et un envoie des données toutes les 60 secondes

Evitez d'utilisez la fonction **delay();** qui est bloquante (mais ne passez pas trop de temps sur cette partie.

Créez votre code de décodage du payload côté TTN

4 - Vérifiez que vous recevez bien les données côté Server LoRaWAN TTN (LiveData)

5 - Créez la fonction *function decodeUplink(input) {* **... et vérifiez que vous récupérez bien les valeurs de vos 3 paramètres !**

Bon courage !!!

Architecture de Description de **LoRa**WAN avec The Things **N**etwork

Partie

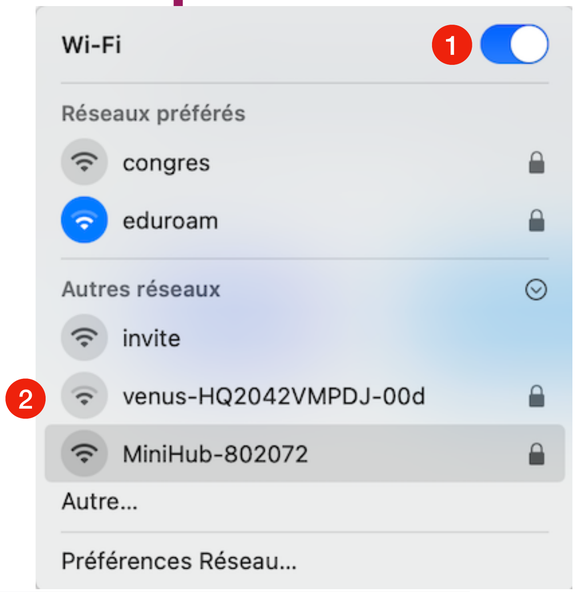
4

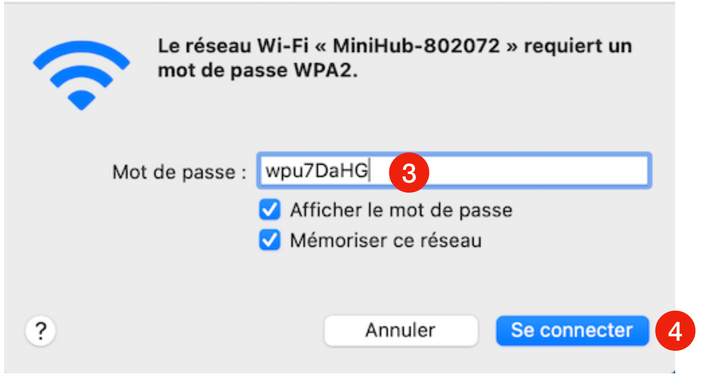
Mise en oeuvre passerelle LoRaWAN

Activer la passerelle TTIG
(https://www.thethingsnetwork.org/docs/gateways/thethingsindoor/)

Mise en oeuvre passerelle LoRaWAN

wifi de la TTIG



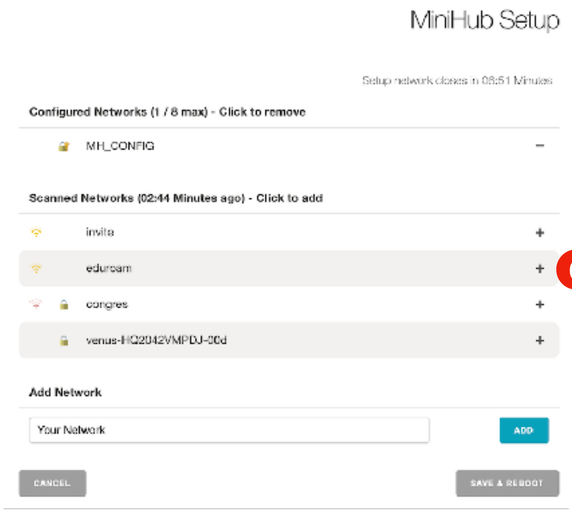


WiFi PW

https://www.thethingsindustries.com/docs/gateways/thethingsindoorgateway/#connecting-the-things-indoor-gateway

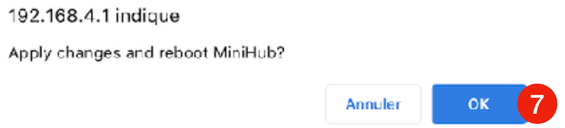
navigateur ⚠ Non sécurisé | 192.168.4.1

5 **@url** 192.168.4.1



6 sélectionnez un wifi perso

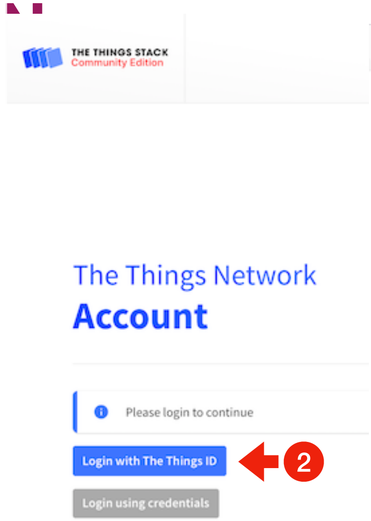
save & reboot >> cfg

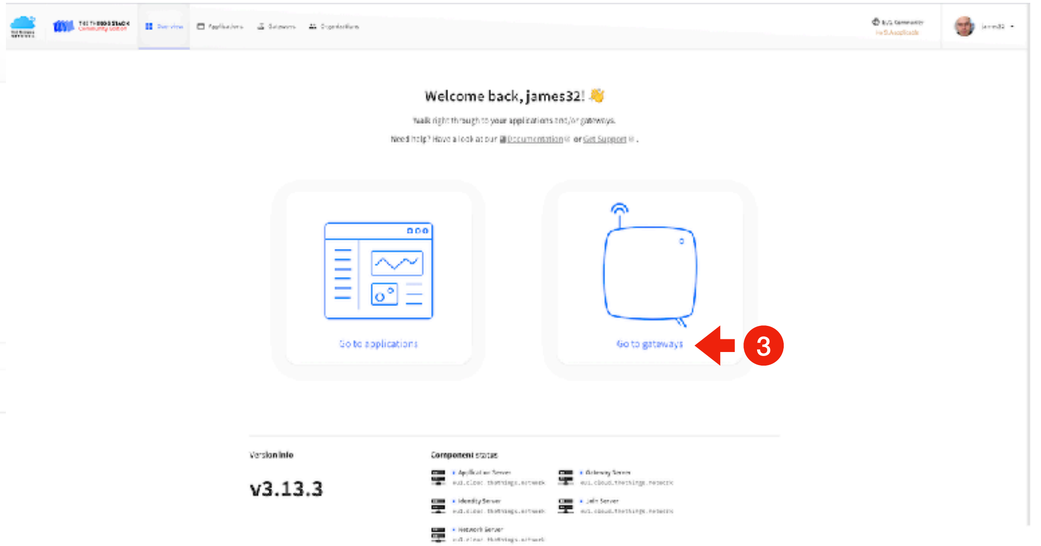


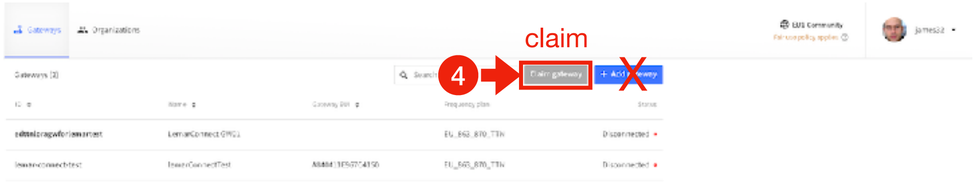
done

Gateway EUI58-A0-CB-FF-FE-80-20-72
WiFi AP MAC58:A0:CB:80:20:72
WiFi AP Passwpu7DaHG
WiFi STA MACCC:50:E3:D8:74:5F
Serial NumberTBMH100868006165
MFG date2019-11-10 05:43:47
FW Build2020-05-07 16:03:53
FW Version2.0.4
Core Version2.0.4(minihub/debug)

sous google : ttn puis console sous votre login







Gateway EUI [redacted] 5
WiFi AP MAC 58:A0:CB:80:20:72
WiFi AP Pass wpu7DaHG 6
WiFi STA MAC CC:50:E3:D8:74:5F
Serial Number TBMH100868006165
MFG date 2019-11-10 05:43:47
FW Build 2020-05-07 16:03:53
FW Version 2.0.4
Core Version 2.0.4(minihub/debug)