LoRa / LoRaWAN

Quelle: www.semtech.com

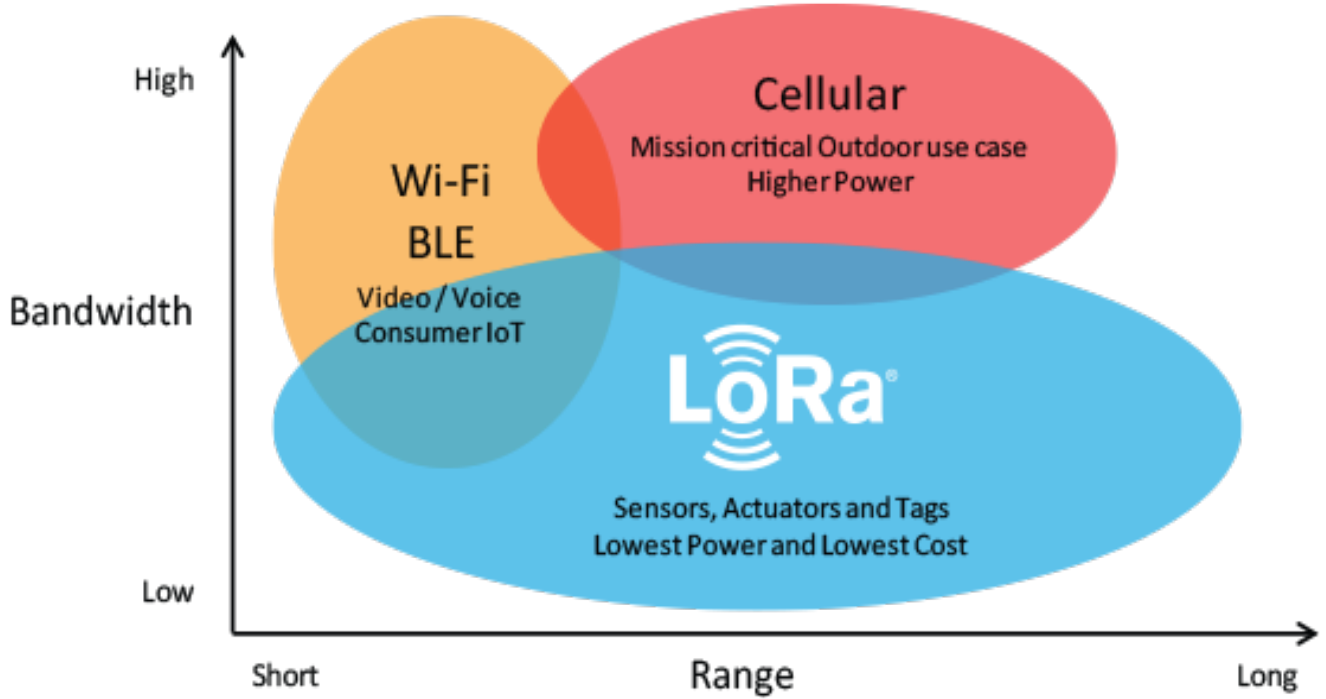
Erstellt von Karl-Heinz Lohner

Was ist LoRa / LoRaWAN

- LoRa® definiert die Übertragungsschicht (Long Range)
- LoRaWAN® definiert das Netzwerk

  (Long Range Wide Area Network)

Vergleich Funk-Übertragungstechniken



Quelle: www.semtech.com

Übersicht

- 1 - 9 LoRa Grundlagen
- 10 – 23 LoRaWAN Grundlagen
- 23 – 43 LoRaWAN – Aktivierung von End devices
- 44 – 50 TTN – Registrieren eines Gateyway
- 51 – 52 TTN – Registrieren eines End Device
- 53 – 65 Chirpstack – Add device profile, Add application, Add Gateway, Add device
- 66 – 67 Links, Quellen

LoRaWAN Einführung

- Die Basis von LoRaWAN ist LoRa. Es zählt zu den LPWAN (Low Power Wide Area Networks).
- Mit dieser Übertragungstechnik können IoT-Geräte mit niedrigen Datenraten bei geringem Energieverbrauch und hoher Reichweite vernetzt werden.
- Batterielebensdauer bis zu 10 Jahren, realistisch 1-2 Jahre
- Für LoRa können in Europa folgende lizenzfreie Frequenzbänder benützt werden (ISM-Band):

  433 MHz (unidirektional)
  868 MHz (bidirektional) (863 - 870 MHz)
  2,4 GHz
  5 GHz

Übertragungstechnik LoRa

- Bidirektional, Uplink bevorzugt. Empfangsquittungen möglich.
- Modulation CSS (Chirp-Spread-Spectrum) und FSK (Frequenz Shift Keying) . FSK ist nur über einen Kanal möglich.
- In Europa werden die Frequenzbereiche 433 MHz und 868 MHz benutzt.
- Für CSS werden nebeneinander liegende Frequenzen (Kanäle) verwendet.
- Die Kanalbelegungsdauer ist reglementiert. Duty Cycle 1% D.h. 36 Sekunden Sendezeit pro Stunde.

Übertragungstechnik LoRa

- Die erzielbare Reichweite beträgt ca. 1 km bis 15 km. Abhängig von der Bebauung und der Topografie.
- Der Ruhestromverbrauch reicht von einigen hundert nA bzw. uA bis einige mA.
- Begrenzung der Sendeleistung auf +14 dBm bzw. maximal 25 mW.
- In der EU ist Payload auf 51 Byte begrenzt. (64Byte Paketlänge - 13 Byte Header).

Frequenzplan EU868

- Uplink:
  - 868.1 - SF7BW125 to SF12BW125
  - 868.3 - SF7BW125 to SF12BW125 and SF7BW250
  - 868.5 - SF7BW125 to SF12BW125
  - 867.1 - SF7BW125 to SF12BW125
  - 867.3 - SF7BW125 to SF12BW125
  - 867.5 - SF7BW125 to SF12BW125
  - 867.7 - SF7BW125 to SF12BW125
  - 867.9 - SF7BW125 to SF12BW125
  - 868.8 – FSK
- Downlink:
  - Uplink channels 1-9 (RX1)
  - 869.525 - SF9BW125 (RX2)

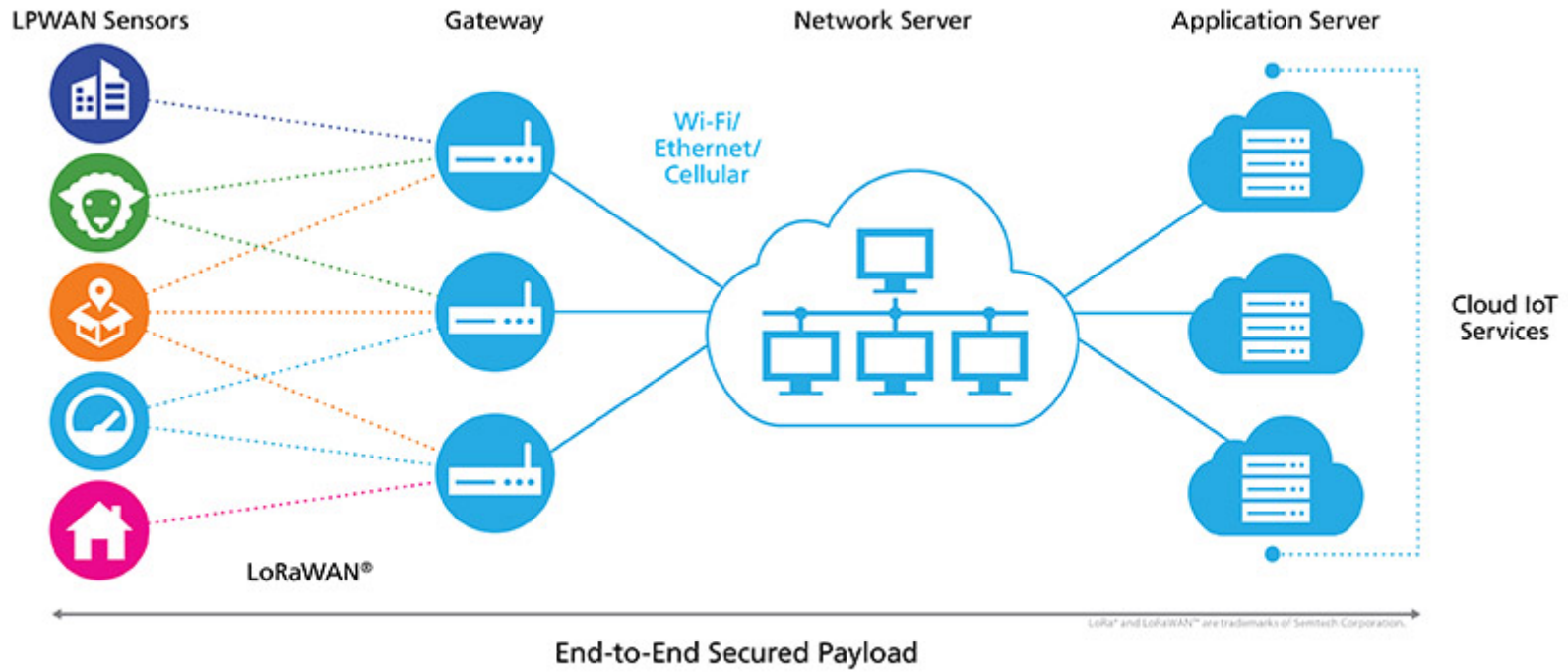
SF, Bandbreite, Bitrate

Spreizfaktor	Bandbreite	Bitrate	Dauer (64Bytes)	Reichweite
SF12	125 kHz	250 Bits/s	2470 ms	
SF11	125 kHz	440 Bits/s	1320 ms	> 2 km
SF10	125 kHz	980 Bits/s	700 ms	
SF9	125 kHz	1.760 Bits/s	390 ms	> 1 km
SF8	125 kHz	3.125 Bits/s	220 ms	
SF7	125 kHz	5.470 Bits/s	120 ms	100 – 500 m
	250 kHz	11.000 Bits/s		
FSK		50.000 Bits/s		

Anwendungsfälle von LoRaWAN

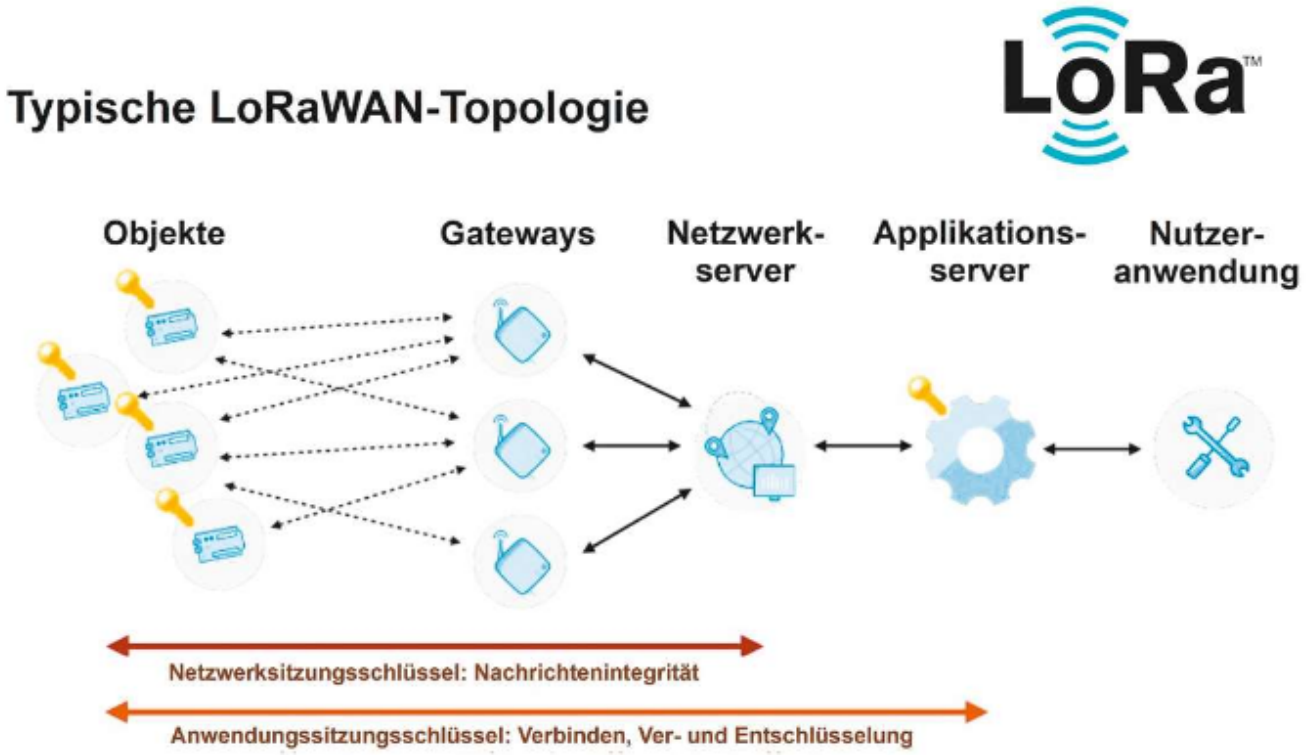
- Überwachung des Raumklimas (Temperatur, Luftfeuchte, co2-Gehalt, VoC)
- Fenstersensoren, Funksteckdosen usw.
- Landwirtschaft (Temperatur, Niederschlag, Bodenfeuchte usw.)
- Überwachung von Maschinen
- Smart City (z.B. Parkplätze, Verkehr)
- Überwachung der Kühlkette

LoRaWAN Architektur



Quelle: https://blog.semtech.com/

LoRaWAN Architektur



Quelle: [3]

LoRaWAN Komponenten

- Ein LoRaWAN Netzwerk besteht aus drei Komponenten:

  Sensoren (Nodes)

  Gateway

  Network- / Application-Server

- Die Nodes senden ihre Daten an alle Gateways in ihrer Umgebung. Die Gateways leiten diese an den/die Network-Server weiter. Diese schicken dann die Daten an Join- und Application-Server zur weiteren Verarbeitung.
- Die Datenübertragung erfolgt verschlüsselt. (nicht beim Join-Request)

LoRaWAN Komponenten

- Gateways
  - Ein oder mehrere Gateways empfangen die LoRa Nachrichten von den End-Devices
  - Es demoduliert die LoRa Nachricht
  - Leitet die Nachricht an den Network-Server weiter
- Network Server
  - Ist für die Authentifizierung und Identifizierung der End-Devices zuständig
  - Sendet die Antworten/Kommandos an die End-Devices
  - Kommuniziert mit Join-Server und Application-Server

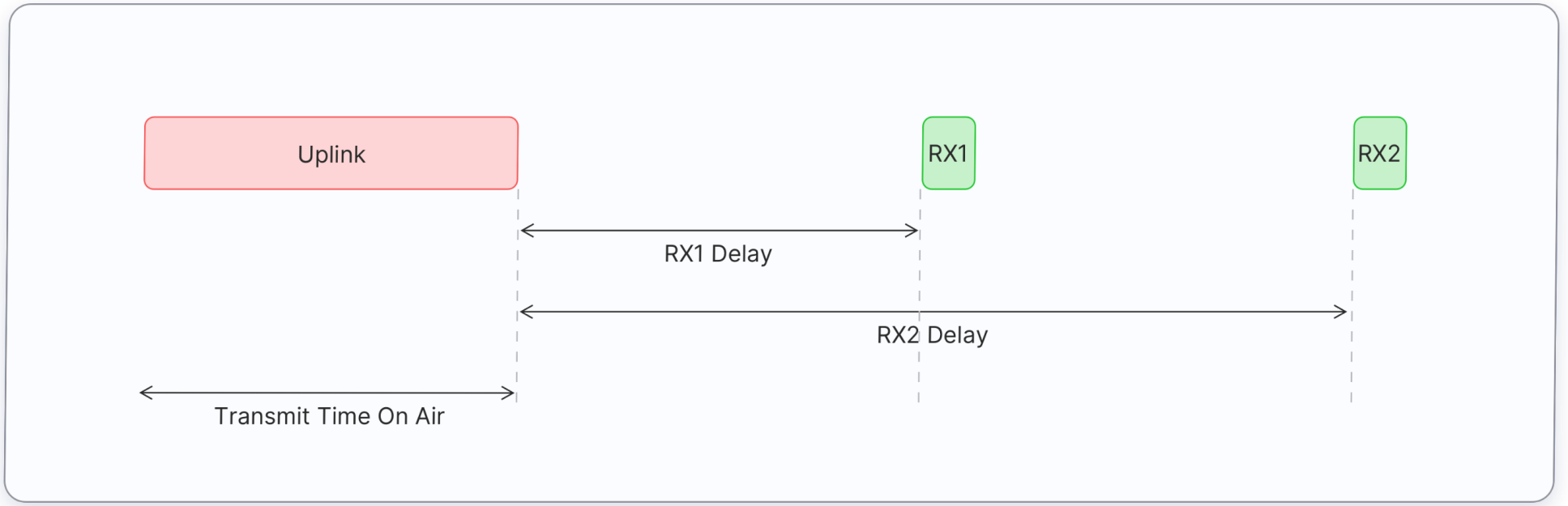
- Application Server
  - Ist für die Anwendungsschicht des Protokolls zuständig
  - Empfängt die Nachrichten vom Network Server und dekodiert diese
  - Verschlüsselt die Nachrichten und sendet sie zum Network Server für den Downlink
  - Bearbeitet AWS-, MQTT-, HTTP-Anfragen und Dashboard usw.

Geräte Klassen

- Es gibt drei Arten von Geräten: Klasse A, Klasse B und Klasse C. Alle LoRaWAN End-Devices müssen die Klasse-A-Implementierung unterstützen.
- End-Device der Klasse A, die oft batteriebetrieben sind und die meiste Zeit im Ruhezustand verbringen, können jederzeit eine Uplink-Nachricht senden. Sobald die Uplink-Übertragung abgeschlossen ist, öffnet das Gerät zwei kurze Empfangsfenster (RX1, RX2). Zwischen dem Ende der Uplink-Übertragung und dem Beginn der Empfangsfenster (RX1 oder RX2) besteht eine Verzögerung. Wenn der Netzwerkserver während dieser beiden Empfangsfenster nicht antwortet, erfolgt der nächste Downlink nach der nächsten Uplink-Übertragung.

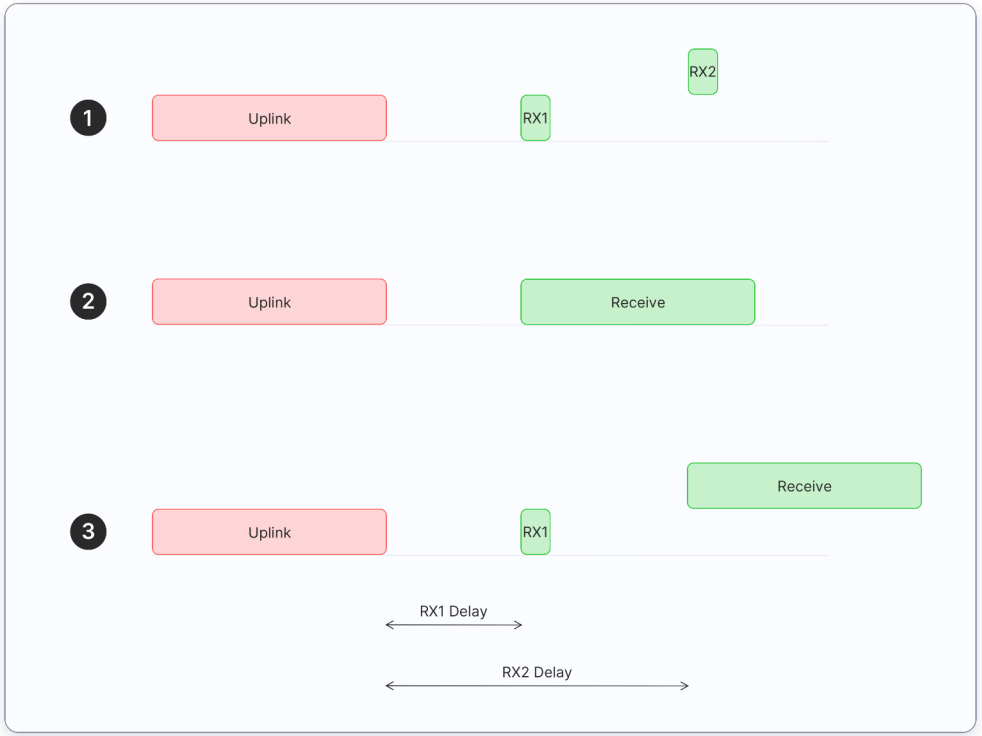
Quelle: [4]

Geräteklasse A



Quelle: [4]

Geräteklasse A



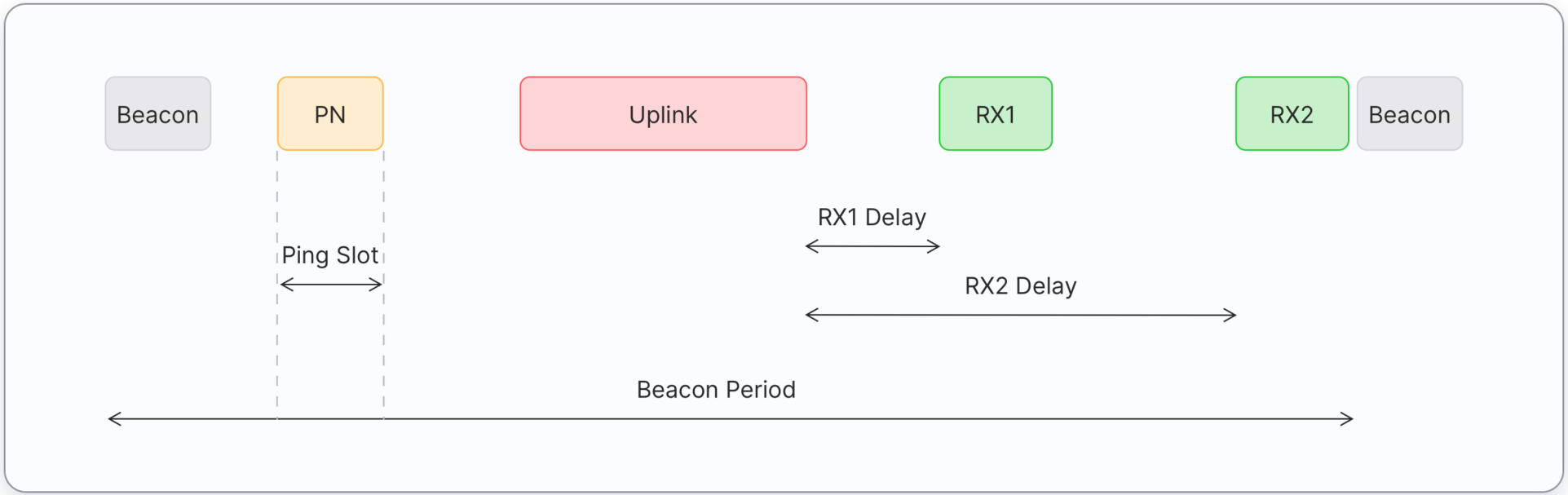
Quelle: [4]

Geräteklasse B

- Zusätzlich zu den Empfangsfenstern, die von Klasse A Geräten geöffnet werden, öffnen Geräte der Klasse B geplante Empfangsfenster, um Downlink-Nachrichten vom Netzwerkserver zu empfangen.
- Mit Hilfe von zeitsynchronisierten Beacons, die vom Gateway übertragen werden, öffnen die Geräte regelmäßig Empfangsfenster. Die Zeit zwischen zwei Beacons wird als Beacon Period bezeichnet. Das Gerät öffnet Downlink-"Ping-Slots" zu geplanten Zeiten für den Empfang von Downlink-Nachrichten vom Netzwerkserver.

Quelle: [4]

Geräteklasse B



- Die Beacon Period beträgt 128 Sekunden

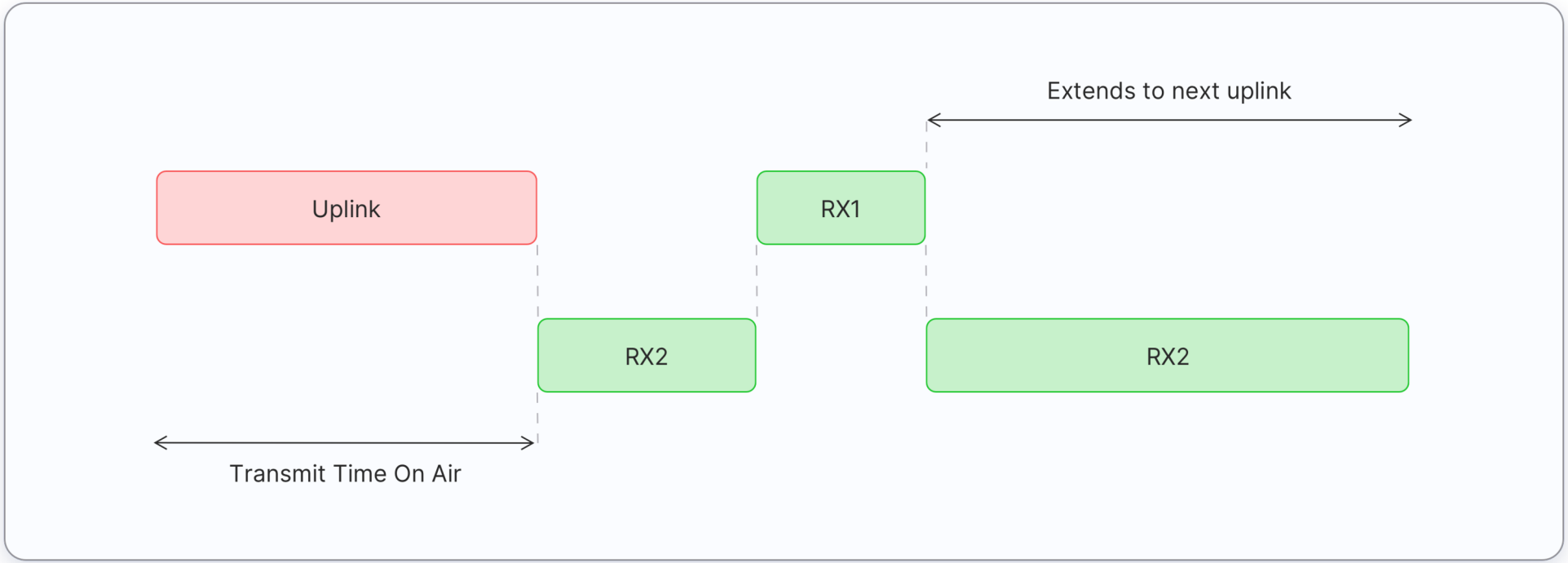
Quelle: [4]

Geräteklasse C

- Geräte der Klasse C erweitern die Klasse A, indem sie Empfangsfenster offen halten, wenn sie nicht senden, wie in der Abbildung unten dargestellt. Dies ermöglicht eine Kommunikation mit geringer Latenz, verbraucht aber ein Vielfaches an Energie im Vergleich zu Geräten der Klasse A.

Quelle: [4]

Geräteklasse C



Quelle: [4]

Aktivierung der End-Devices

- Jedes End-Device muss in einem Netz registriert werden, bevor es Nachrichten senden und empfangen kann. Dieser Vorgang wird als Aktivierung bezeichnet.
- Es sind zwei Aktivierungsmethoden verfügbar:
- Over-The-Air Activation (OTAA) - die sicherste und empfohlene Aktivierungsmethode für End-Devices.
  - Während der Aktivierung wird dem Gerät eine dynamische Geräteadresse zugewiesen und die Keys zur Verschlüsselung erzeugt.

Quelle: [4]

Aktivierung der End-Devices

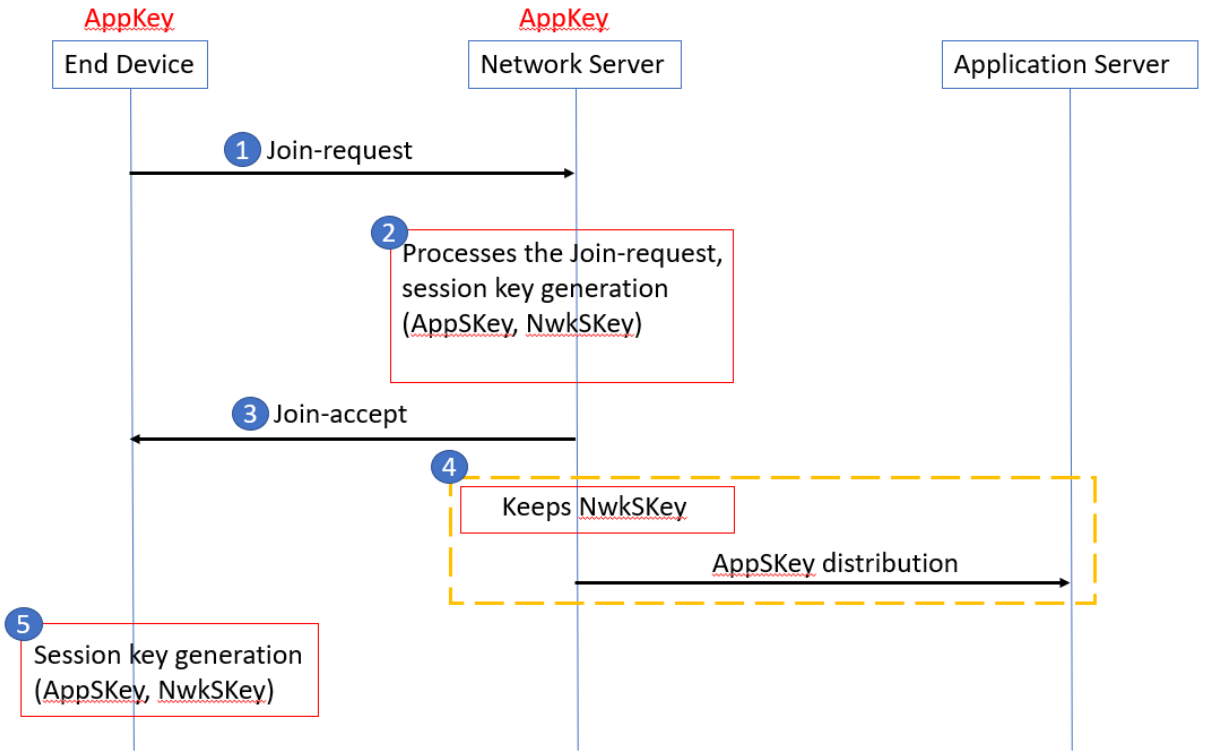
- Aktivierung durch Personalisierung (ABP) - erfordert die harte Codierung der Geräteadresse und der Sicherheitsschlüssel im Gerät. ABP ist weniger sicher als OTAA.
- Ein ABP-Gerät in einem öffentlichen Netzwerk muss die vom Netzwerkserver bereitgestellte DevAddr verwenden. Dies bedeutet, dass das Gerät an dieses Netzwerk gebunden ist und nicht in ein anderes Netzwerk verschoben werden kann.

Quelle: [4]

OTAA Aktivierung LoRaWAN Version 1.0.x

- 'join request' vom End-Device zum Network-Server
- 'join-accept' vom Network-Server zum End-Device
- Vor der Aktivierung müssen **AppEUI**, **DevEUI** und **AppKey** dem End-Device zugewiesen sein.
- Der **AppKey** ist ein AES-128 bit Schlüssel, er wird auch als 'root key' bezeichnet. Der gleiche **AppKey** muss im End-Device und im Network-Server bekannt sein. Der **AppKey** wird nicht übertragen.
- Die 'join request' Nachricht mit einer beliebigen Datenrate über die regionalen 'Join Kanäle' gesendet werden. In Europa sind dies 868,10 MHz, 868,30 Mhz oder 868,50 Mhz.

OTAA Aktivierung LoRaWAN Version 1.0.x



Quelle: [2]

OTAA Aktivierung LoRaWAN Version 1.0.x

- Der Join Prozess wird immer vom End-Device gestarted.
  - Das End-Device sendet dafür folgende Daten zum Network-Server

    8 bytes AppEUI | 8 bytes DevEUI | 2 bytes DevNonce (Number used once)
  - **AppEUI** – Ein global eindeutiger 64-Bit 'application identifier' im IEEE EUI64 Adressraum
  - **DevEUI** – Ein global eindeutiger 64-Bit 'device identifier' im IEEE EUI64 Adressraum
  - **DevNonce** – Ein eindeutiger, zufälliger 2 Byte Wert, vom End-Device erzeugt. Der Network-Server verfolgt anhand des DevNonce die 'join-requests' vom End-Device. Bekommt der Network-Server einen 'join-request' der schon einmal von diesem Gerät benutzt wurde, weist er diesen mit 'reject' ab.
  - Die Daten des 'join request' sind nicht verschlüsselt.

OTAA Aktivierung LoRaWAN Version 1.0.x

- Der Join-Server bearbeitet den 'join request' und generiert zwei 'session keys', (**NwkSKey** and **AppSKey**) und die 'join accept' Nachricht, falls das End-Device dem Netzwerk beitreten darf.

3 bytes	3 bytes	4 bytes	1 byte	1 byte	16 bytes (optional)
AppNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

- **AppNonce** – Eine zufällige und eindeutige ID vom Network-Server. Wird vom End-Device benutzt, um AppSKey und NwkSKey daraus abzuleiten.
- **NetID** – Ein 24-bit eindeutiger 'network identifier'.
- **DevAddr** – Eine 32-Bit 'device address' – vom Network-Server zugewiesen, um das End-Device im aktuellen Netzwerk zu identifizieren.
- **DLSettings** – Ein 1 Byte Feld das die 'downlink settings', die das Gerät benützen soll, beschreibt.

OTAA Aktivierung LoRaWAN Version 1.0.x

- **RxDelay** – Die Verzögerung zwischen TX und RX.
- **CFList** – Ein optionale Liste mit Frequenzen die das End-Device benutzen darf. Ist Regionsspezifisch.
- **MIC** -- (Message Integrity Code) wird mit dem AppKey über alle Felder der 'join accept' Nachricht berechnet und an die 'join accept' Nachricht angehängt.
- Die 'join accept' Nachricht ist mit dem AppKey verschlüsselt.
- Der Network-Server sendet die 'join accept' Nachricht als ganz normalen Downlink.
- Falls der 'join request' nicht akzeptiert wird, wird keine Nachricht an das End-Device gesendet.
- Der Network-Server merkt sich den NwkSKey und schickt den AppSKey zum Application-Server

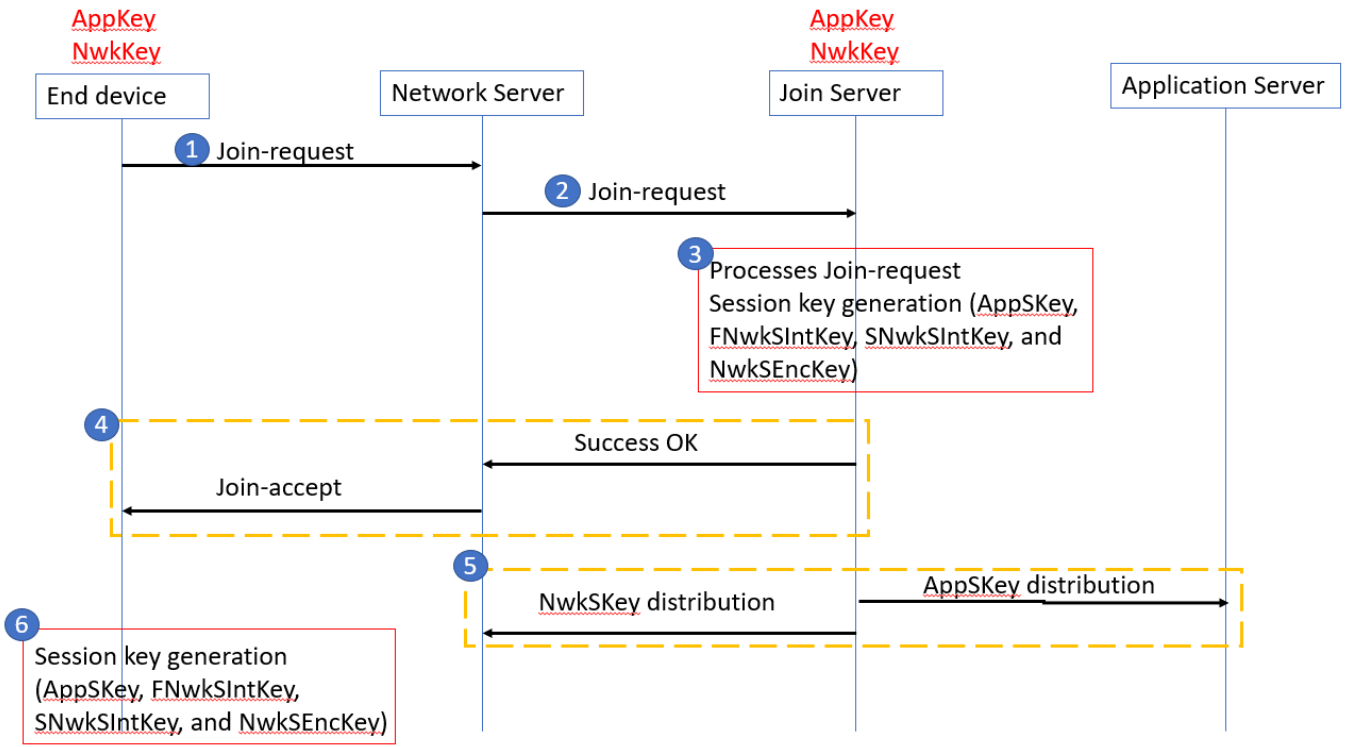
OTAA Aktivierung LoRaWAN Version 1.0.x

- Das End-Device ist jetzt im Netzwerk aktiviert.
- Folgende Informationen werden im End-Device gespeichert
  - **DevAddr** – Eine 32-Bit 'device address' – vom Network-Server zugewiesen, um das End-Device im aktuellen Netzwerk zu identifizieren.
  - **NwkSKey** – Der 'network Session Key' wird vom End-Device und vom Netzwerkserver verwendet, um den Message Integrity Code (MIC) aller Datennachrichten zu berechnen und zu überprüfen, um die Nachrichtenintegrität sicherzustellen. Der NwkSKey wird auch zum Verschlüsseln und Entschlüsseln von Nutzdaten mit MAC-Befehlen verwendet.
  - **AppSKey** – Der 'Application Session Key' wird zum Verschlüsseln und Entschlüsseln der Anwendungdaten (payload) vom / zum 'application server' verwendet. Damit wird die Vertraulichkeit der Daten sichergestellt.

OTAA Aktivierung LoRaWAN Version 1.1

- 'join request' vom End-Device zum Network-Server
- 'join accept' vom Network-Server zum End-Device
- Vor der Aktivierung müssen **JoinEUI**, **DevEUI**, **AppKey** und **NwkKey** müssen im End-Device gespeichert sein.
- **AppKey** und **NwkKey** sind AES-128 Bit Schlüssel und werden als **root keys** bezeichnet.
- Ein MIC (Message Integrity Code) wird mit dem NwkKey über alle Felder der 'join request' Nachricht berechnet und an die 'join request' Nachricht angehängt.
- **AppKey** und **NwkKey** werden **niemals** über das Netzwerk gesendet.

OTAA Aktivierung LoRaWAN Version 1.1



Quelle: [2]

OTAA Aktivierung LoRaWAN Version 1.1

- Der Join Prozess wird immer von End-Device gestarted.
  - Das End-Device sendet dafür folgende Daten zum Network-Server

    8 bytes JoinEUI | 8 bytes DevEUI | 2 bytes DevNonce
  - **JoinEUI** – Ein global eindeutiger 64-Bit 'application identifier' im IEEE EUI64 Adressraum. Der 'Join Server' bearbeitet den 'join request' und erzeugt die Session Keys.
  - **DevEUI** – Ein global eindeutiger 64-Bit 'device identifier' im IEEE EUI64 Adressraum
  - **DevNonce** – Ein eindeutiger, zufälliger 2 Byte Wert, von End-Device erzeugt. Der Network-Server verfolgt anhand des DevNonce die 'join-requests' vom End-Device. Bekommt der Network-Server einen 'join request' der schon einmal von diesem Gerät benutzt wurde, weist er diesen mit 'reject' ab. Das **DevNonce** soll 'replay attacken' verhindern.
  - Die Daten des 'join request' sind nicht verschlüsselt.

OTAA Aktivierung LoRaWAN Version 1.1

- In LoRaWAN 1.1 Ist die **AppEUI** durch **JoinEUI** ersetzt.
- Die 'join request' Nachricht mit einer beliebigen Datenrate über die regionalen 'Join Kanäle' gesendet werden. In Europa sind dies 868,10 MHz, 868,30 Mhz oder 868,50 Mhz.
- Die 'join request' Nachricht kann ein oder über mehrere Gateways zum 'network server' übertragen werden.
- Wenn der 'join request' nicht akzeptiert wird, wird keine Antwort zum 'end device' geschickt.

OTAA Aktivierung LoRaWAN Version 1.1

- Der 'network server' leitet die 'join request' Nachricht an den zuständigen 'join server' weiter.
- Der 'join jerver' bearbeitet den 'join request'. Der 'join server' generiert alle erforderlichen session keys (**AppSKey**, **FNwkSIntKey**, **SNwkSIntKey**, und **NwkSEncKey**) falls das 'end device' in das Netzwerk integriert wird.
- Wenn der 'join request' erfolgreich ist, sendet der 'network server' eine 'join accept' Nachricht mit folgendem Inhalt an das 'end device':

1 byte	3 bytes	4 bytes	1 bytes	1 bytes	16 bytes
JoinNonce	NetID	DevAddr	DLSettings	RXDelay	CFList

- **JoinNonce** – Ein gerätespezifischer Zähler der vom 'join server' vergeben ist und vom 'end device' benutzt wird, um die 'session keys' **FNwkSIntKey**, **SNwkSIntKey**, **NwkSEncKey und AppSKey** abzuleiten.
- **NetID** – Ein 24-bit eindeutiger 'network identifier'.
- **DevAddr** – Eine 32-Bit 'device address' – vom Network-Server zugewiesen, um das End-Device im aktuellen Netzwerk zu identifizieren.

OTAA Aktivierung LoRaWAN Version 1.1

- **DLSettings** – Ein 1 Byte Feld das die 'downlink settings', die das Gerät benützen soll, beschreibt.
- **RxDelay** – Die Verzögerung zwischen TX und RX.
- **CFList** – Ein optionale Liste mit Frequenzen die das End-Device benutzen darf. Ist Regionsspezifisch.
- **MIC** -- Der Message Integrity Code (MIC) wird über alle Felder in der 'join accept' Nachricht mit dem NwkKey (für LoRaWAN 1.0 devices) oder JSIntKey (für LoRaWAN 1.1 devices) errechnet. Der errechnete MIC wird an die 'join accept' Nachricht angehängt.
- Die 'join accept' Nachricht wird dann mit dem **NwkKey** (falls ein 'join request' angefordert wurde) oder dem **JSEncKey** (bei einem 'rejoin request') verschlüsselt. Der 'Network Server' benutzt eine AES decrypt operation in ECB mode um die 'join accept' Nachricht zu verschlüsseln.
- Dann sendet der 'Network Server' die verschlüsselte 'join accept' Nachricht zum 'end device' als ganz normalen Downlink.
- Wenn der 'join request' nicht akzeptiert wird, wird keine Nachricht zum 'end device' gesendet.

OTAA Aktivierung LoRaWAN Version 1.1

- Der 'Join Server' sendet den **AppSKey** zum 'Application Server', und die drei network session keys (**FNwkSIntKey**, **SNwkSIntKey**, and **NwkSEncKey**) zum 'Network Server'.
- Das 'end-device' entschlüsselt die 'Join-accept' Nachricht mit einer AES decrypt operation. Das 'end device' benutzt **AppKey**, **NwkKey**, und **JoinNonce** um die 'session keys' zu generieren.
- Für **LoRaWAN 1.0.x** Geräte:
  - Der **AppSKey** ist vom **NwkKey** abgeleitet.
  - **FNwkSIntKey**, **SNwkSIntKey**, und **NwkSEncKey** werden auch vom **NwkKey** abgeleitet.
- Für **LoRaWAN 1.1** Geräte:
  - Der **AppSKey** ist vom **AppKey** abgeleitet.
  - **FNwkSIntKey**, **SnwkSIntKey** und **NwkSEncKey** werden vom **NwkKey** abgeleitet.
- Das End-Device ist jetzt im Netzwerk aktiviert.

OTAA Aktivierung LoRaWAN Version 1.1

Nach der Aktivierung sind folgende Informationen im End-Device gespeichert:

- **DevAddr** - Eine 32-Bit 'device address' – vom Network-Server zugewiesen, um das End-Device im aktuellen Netzwerk zu identifizieren.
- **FNwkSIntKey** – Ein 'network session key' der vom End-Device verwendet wird, um den MIC (teilweise) von allen Uplink Nachrichten zu berechnen.
- **SNwkSIntKey** – Ein 'network session key' der vom End-Device verwendet wird, um den MIC (teilweise) von allen Uplink Nachrichten und den MIC Downlink Daten Nachrichten zu berechnen.
- **NwkSEncKey** -- Ein 'network session key' der benutzt wird, um die Nutzdaten von MAC Kommandos zu verschlüsseln und zu entschlüsseln.
- **AppSKey** – Ein 'session key'. Vom 'Application Server' und vom 'end device' verwendet, um die 'application data' zu verschlüsseln und zu entschlüsseln. Damit wird die Vertraulichkeit sicher gestellt.

ABP – Activation By Personalization

- Bei der Aktivierung durch Personalisierung (ABP) wird ein End-Device direkt an ein vorab ausgewähltes Netzwerk gebunden, wobei das drahtlose Aktivierungsverfahren umgangen wird. Die Aktivierung durch Personalisierung ist die weniger sichere Aktivierungsmethode und hat außerdem den Nachteil, dass Geräte den Network-Provider nicht wechseln können, ohne die Schlüssel im Gerät manuell zu ändern. Ein Join-Server ist am ABP-Prozess nicht beteiligt.
- Ein mit der ABP-Methode aktiviertes End-Device kann nur mit einem einzigen Netzwerk arbeiten und behält während seiner gesamten Lebensdauer dieselbe Sicherheitssitzung bei.

ABP in LoRaWAN 1.0.x

- Die **DevAddr** und die zwei session keys **NwkSKey** und **AppSKey** sind anstatt **DevEUI**, **AppEUI** und der **AppKey** sind im End-Device dauernhaft gespeichert. Jedes End-Device muss einen eindeutigen Satz **NwkSKey** und **AppSkey** haben. Dieselbe **DevAddr** und derselbe **NwkSKey** müssen dem 'Network Server' bekannt sein. Der **AppSKey** muss im 'Application Server' gespeichert sein.

ABP in LoRaWAN 1.0.x

Quelle: [2]

ABP in LoRaWAN 1.1

- Anstatt **DevEUI**, **JoinEUI**, **AppKey** und **NwkKey** müssen die **DevAddr** und die vier 'session keys' **FNwkSIntKey**, **SNwkSIntKey**, **NwkSEncKey** und **AppSKey** im 'end device' gespeichert sein.
- Die selben **DevAddr**, **FNwkSIntKey**, **SNwkSIntKey** und **NwkSEncKey** müssen auch im 'Network Server' gespeichert sein. Der **AppSKey** muss im 'Application Server' gespeichert sein.

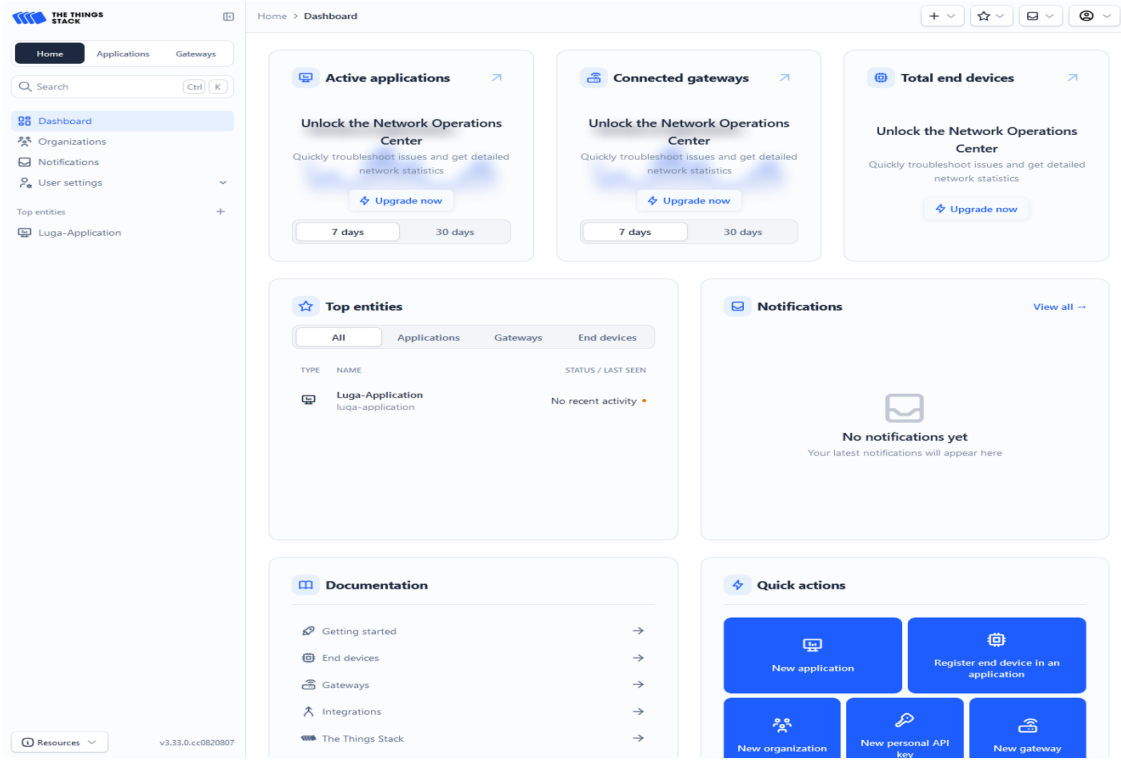
ABP in LoRaWAN 1.1

Quelle: [2]

Network Server Konfiguration

- Damit ein LoRaWAN Netzwerk funktioniert, muss der Network Server konfiguriert werden.
- Die notwendigen Schritte sind auf den folgenden Seiten beschrieben.
  - Konfiguration eines evtl. vorhandenen Gateway
  - Aufnahme eines End-Device ins LoRaWAN Netzwerk

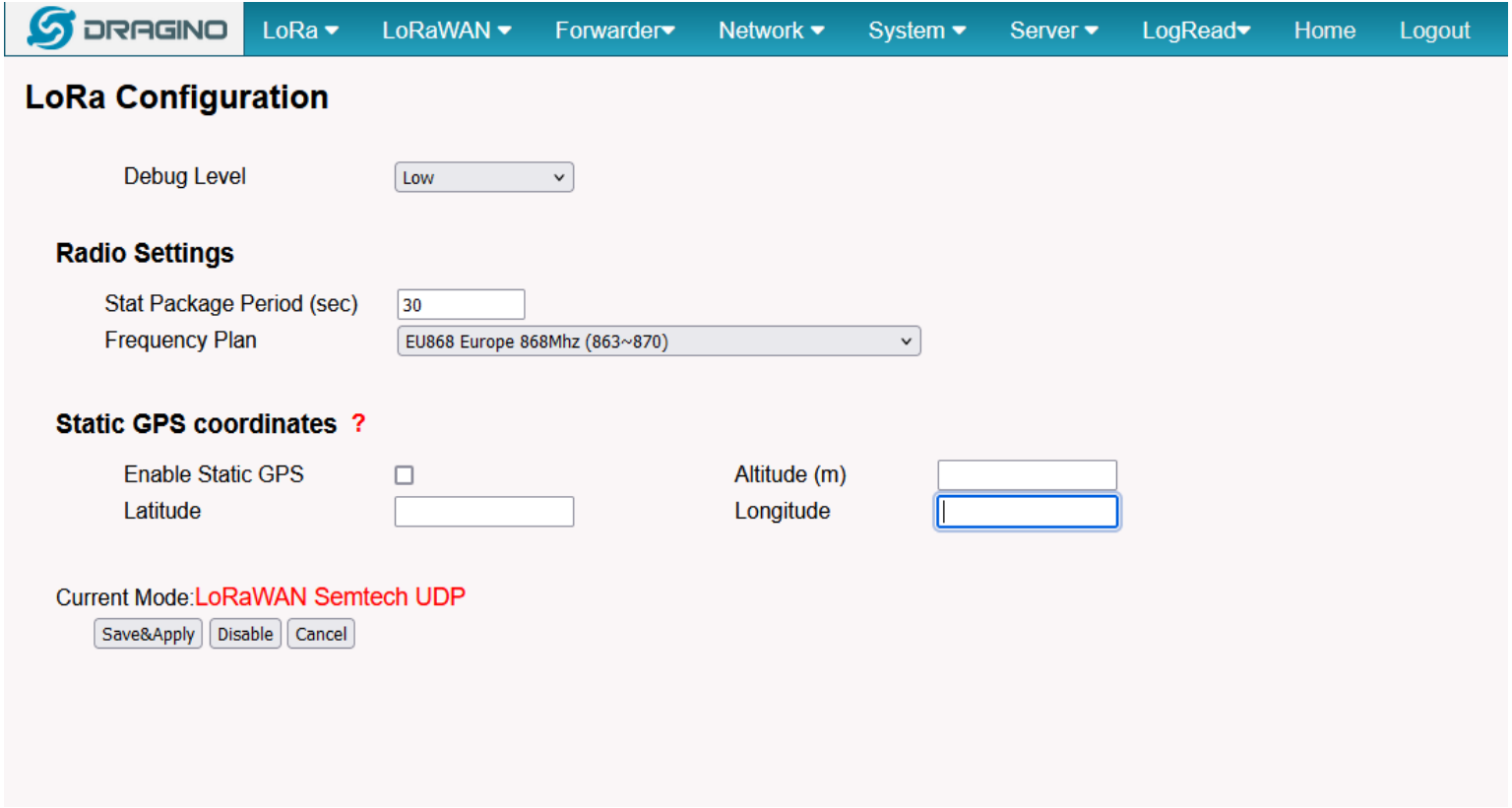
The Things Network



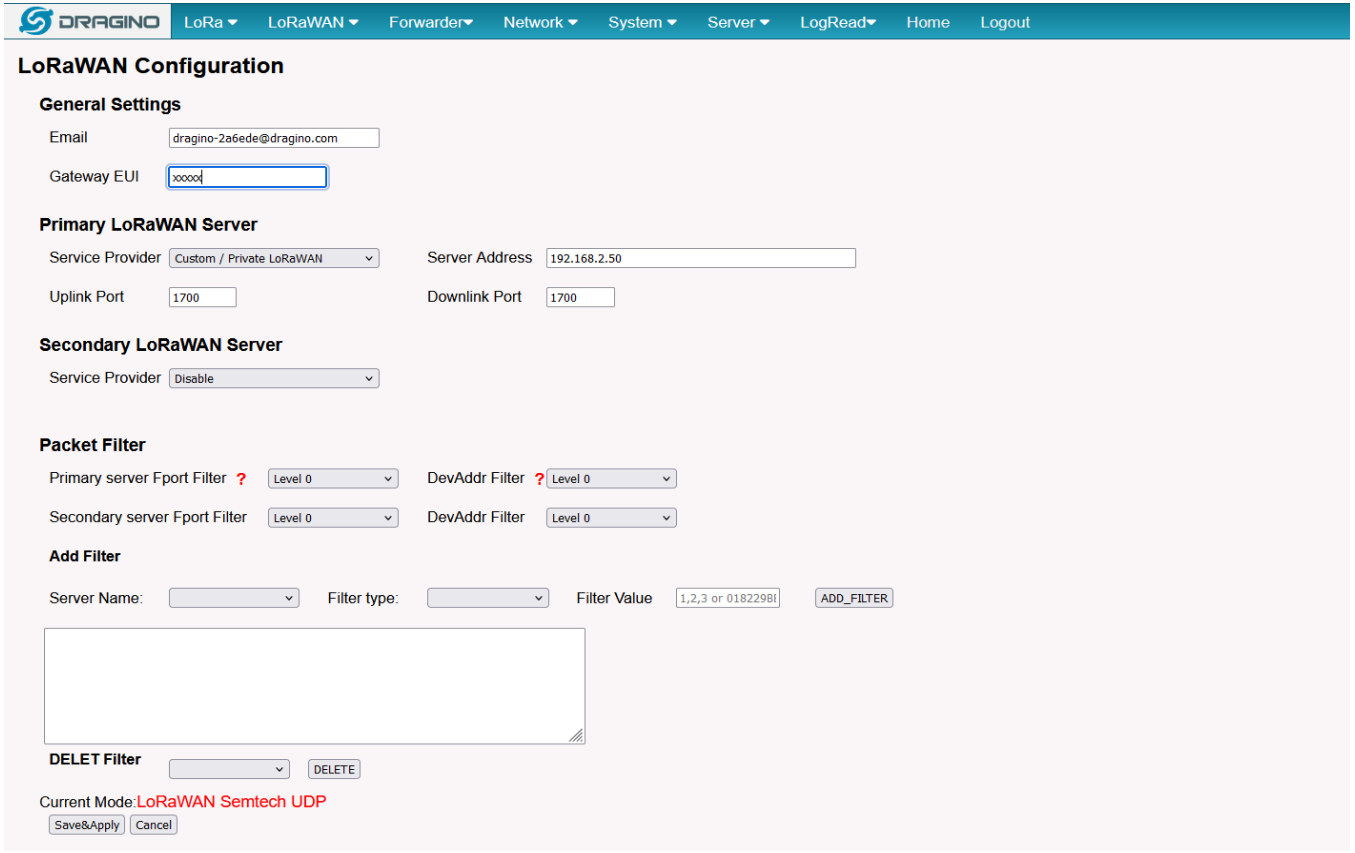
The Things Network

- Ein Praxisbeispiel mit ‚The Things Network‘ als Network-Server.
- Konfiguration eines Gateway für TTN
- Registrieren eines Gateway
- Registrieren eines End-Device

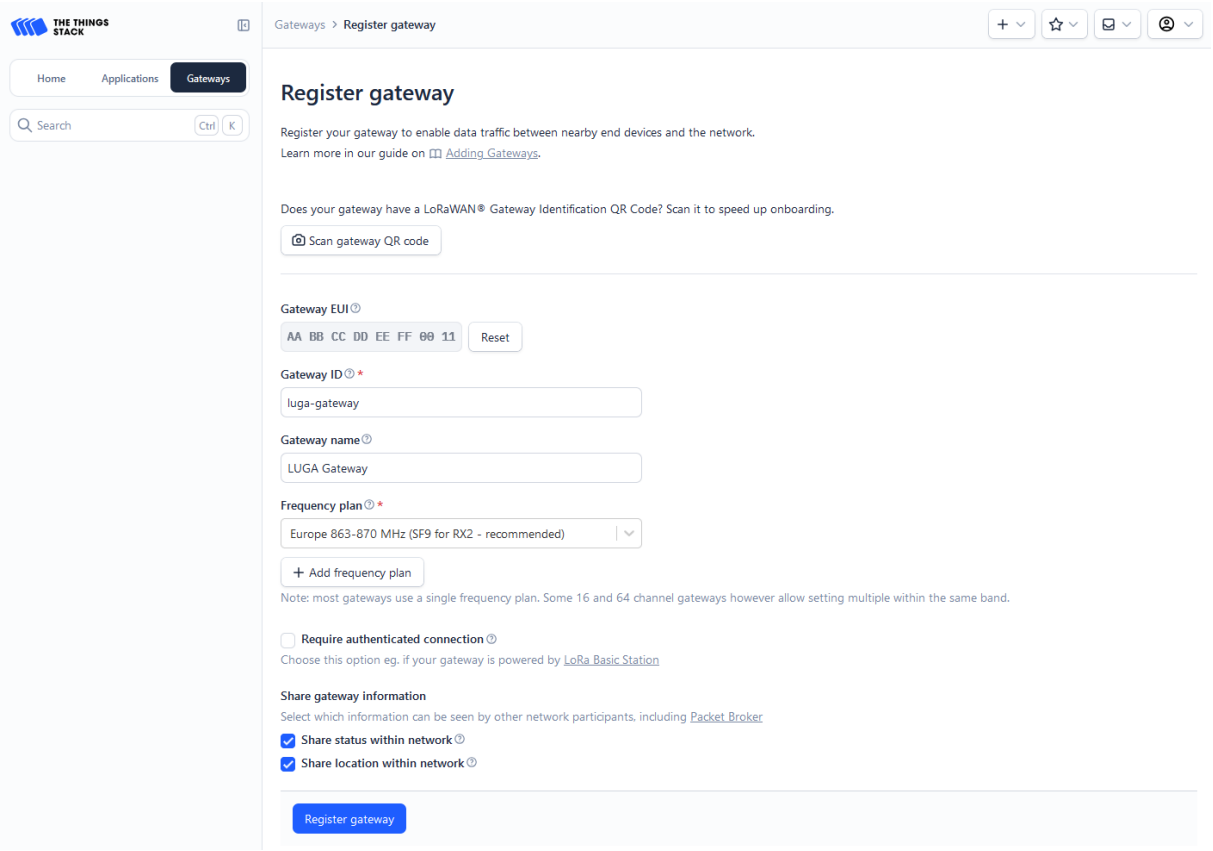
Registrieren eines Gateway



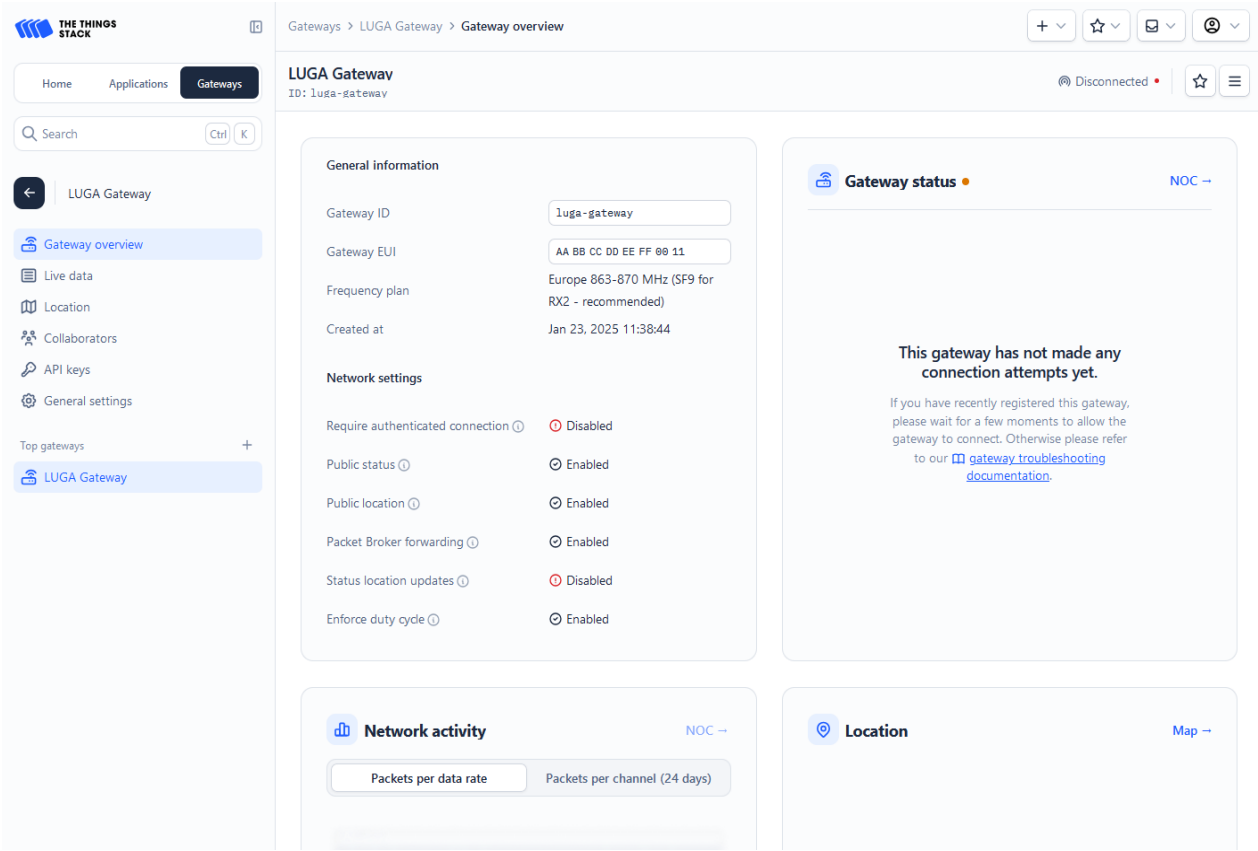
Konfiguration eines Gateway



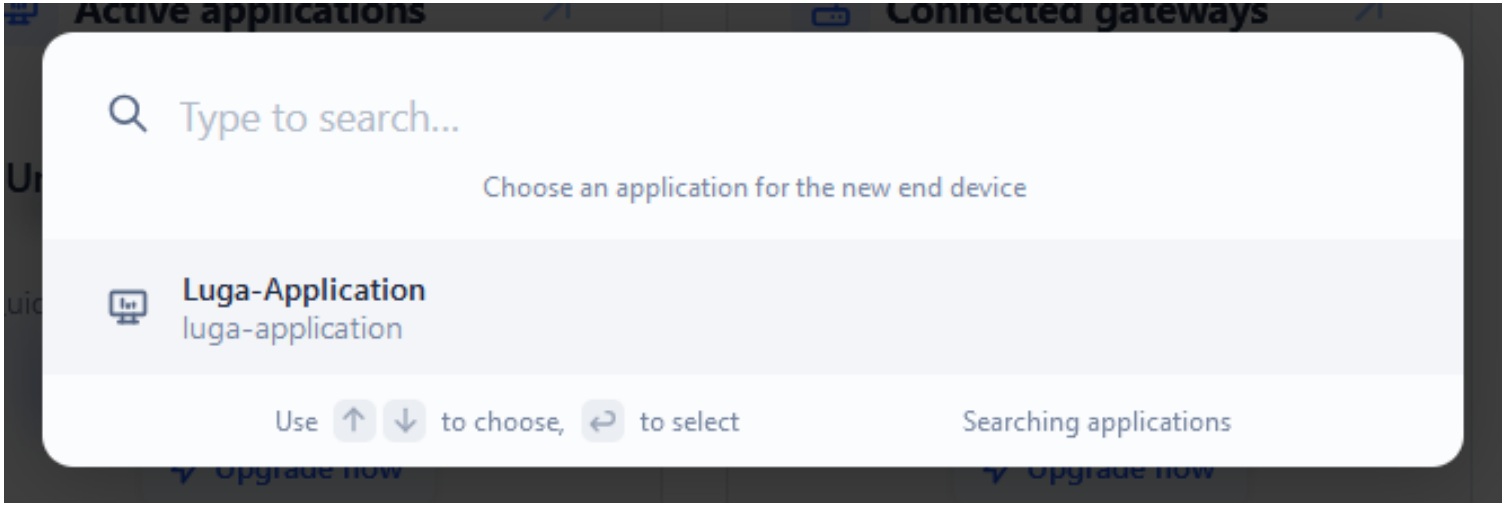
Registrieren des Gateway



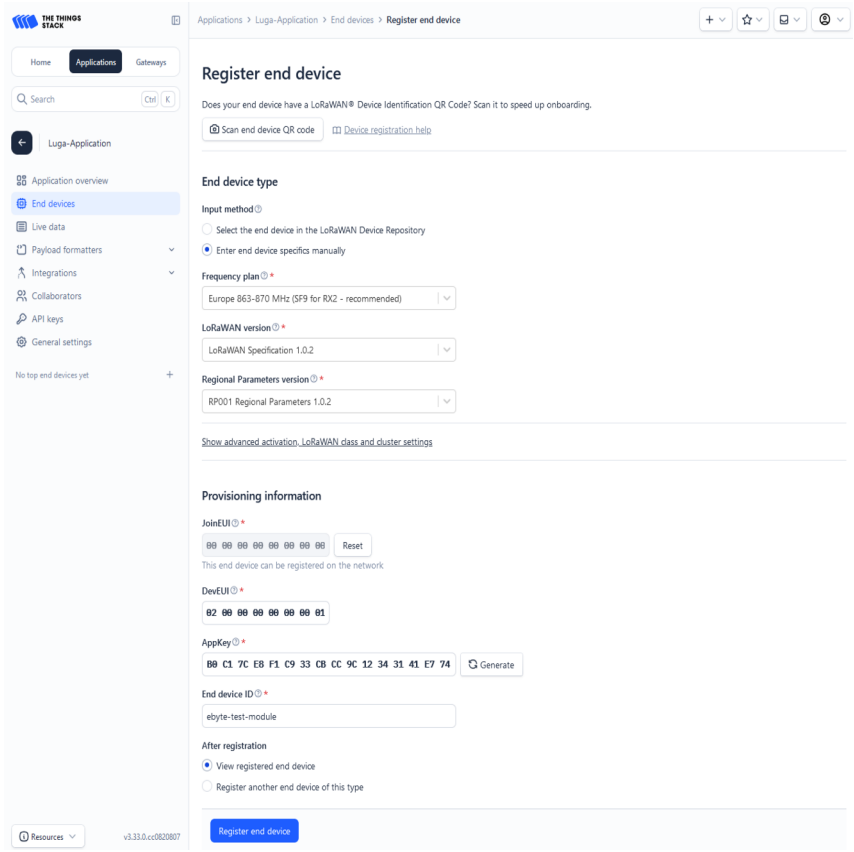
Das Gateway ist registriert



Register end Device in an application



Register End Device



ChirpStack

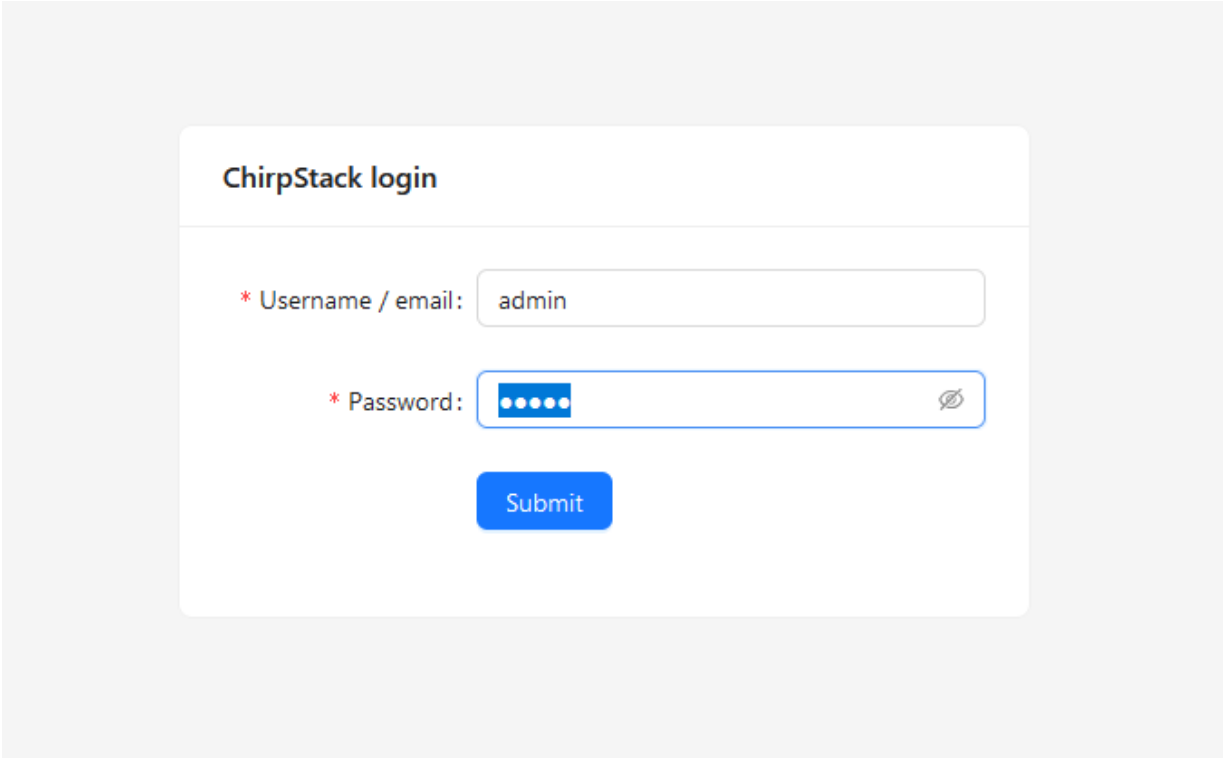
- Alternativ kann der Chirpstack als Network-Server dienen.
- In diesem Beispiel wurde eine Dockerinstanz auf einem Raspberry Pi installiert.

1) Klonen Github Repository:
   git clone https://github.com/chirpstack/chirpstack-docker.git
2) Cd ./chirpstack-docker
3) Container starten:

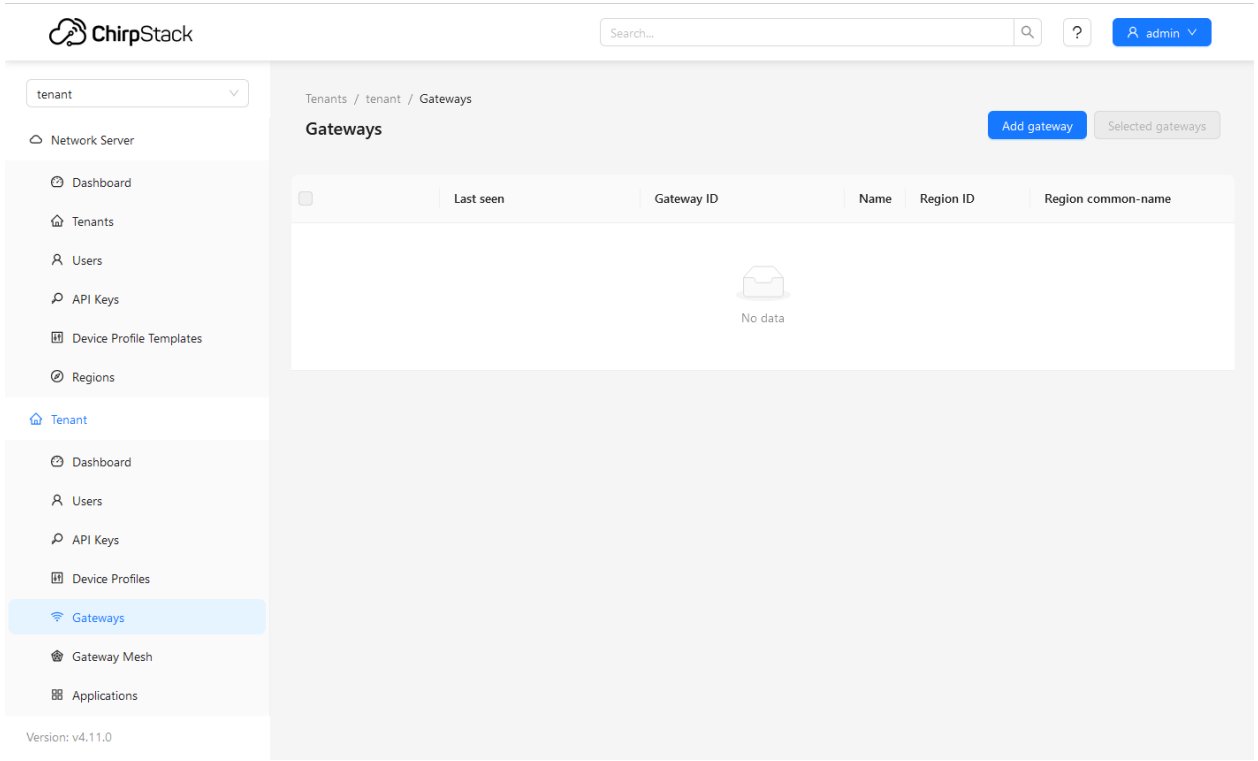
   docker compose up -d
4) Wenn der Stack läuft (docker compose ps, docker compose logs), mit dem Chirpstack verbinden:

   https://192.168.178.xx:8080
5) Mit User: ‚admin' und Passwort: ‚admin' anmelden.

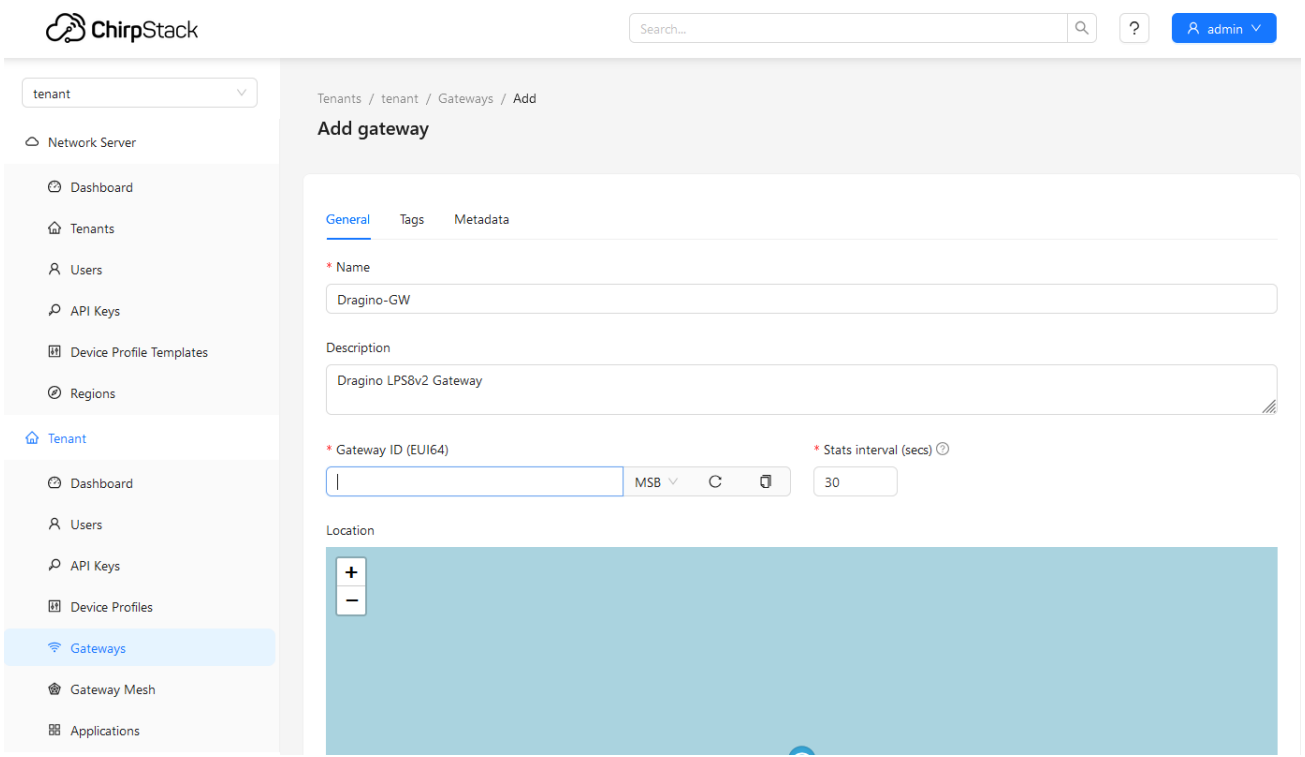
ChirpStack



ChirpStack – Add Gateway



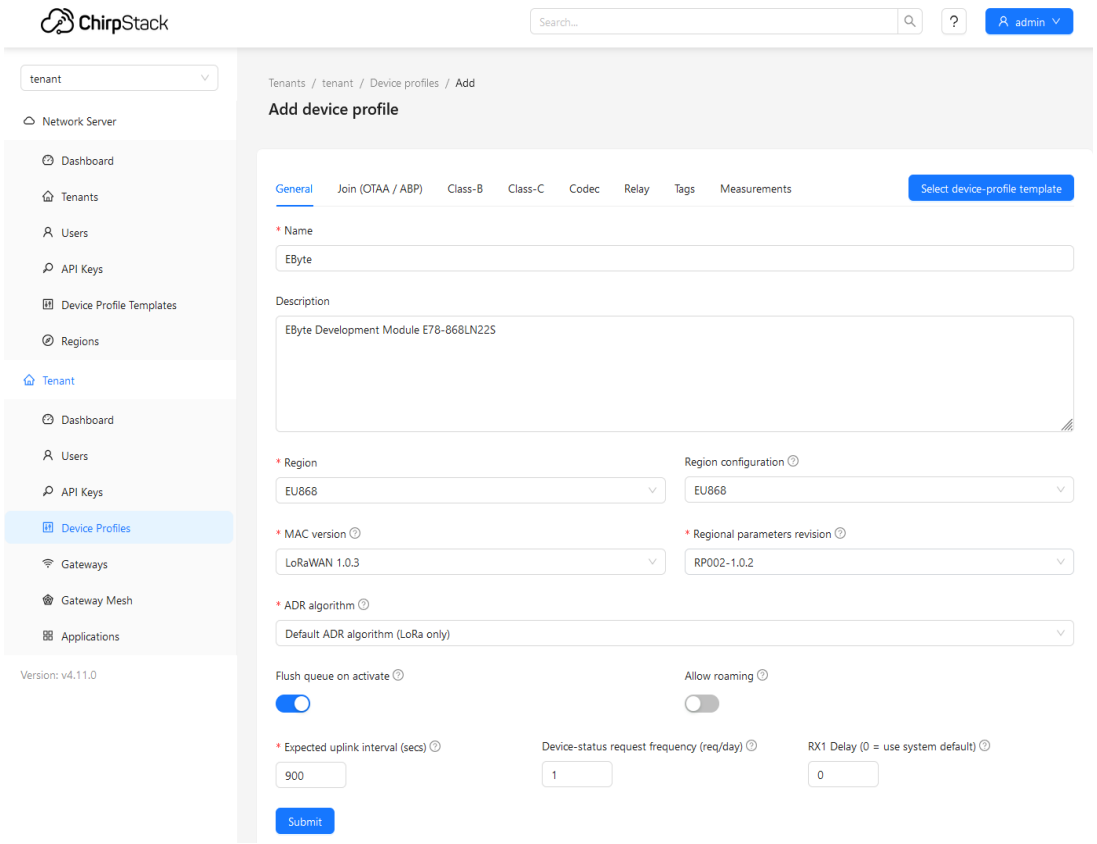
ChirpStack – Add Gateway



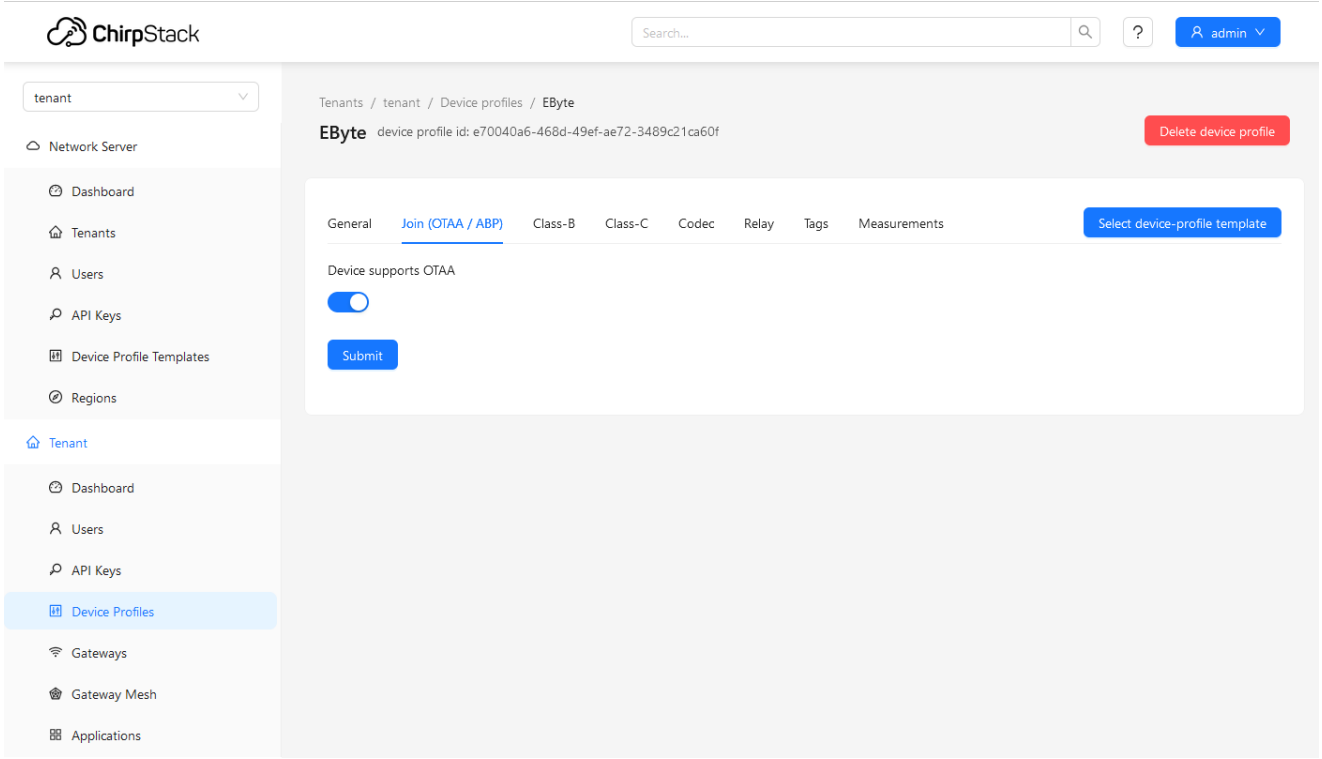
ChirpStack – Add End-Device

- Damit ein End-Device im ChirpStack registiert werden kann, müssen folgende Schritte durchgeführt werden.
  - Add device profile
  - Add application
  - Add device mit Auswahl eines device profile

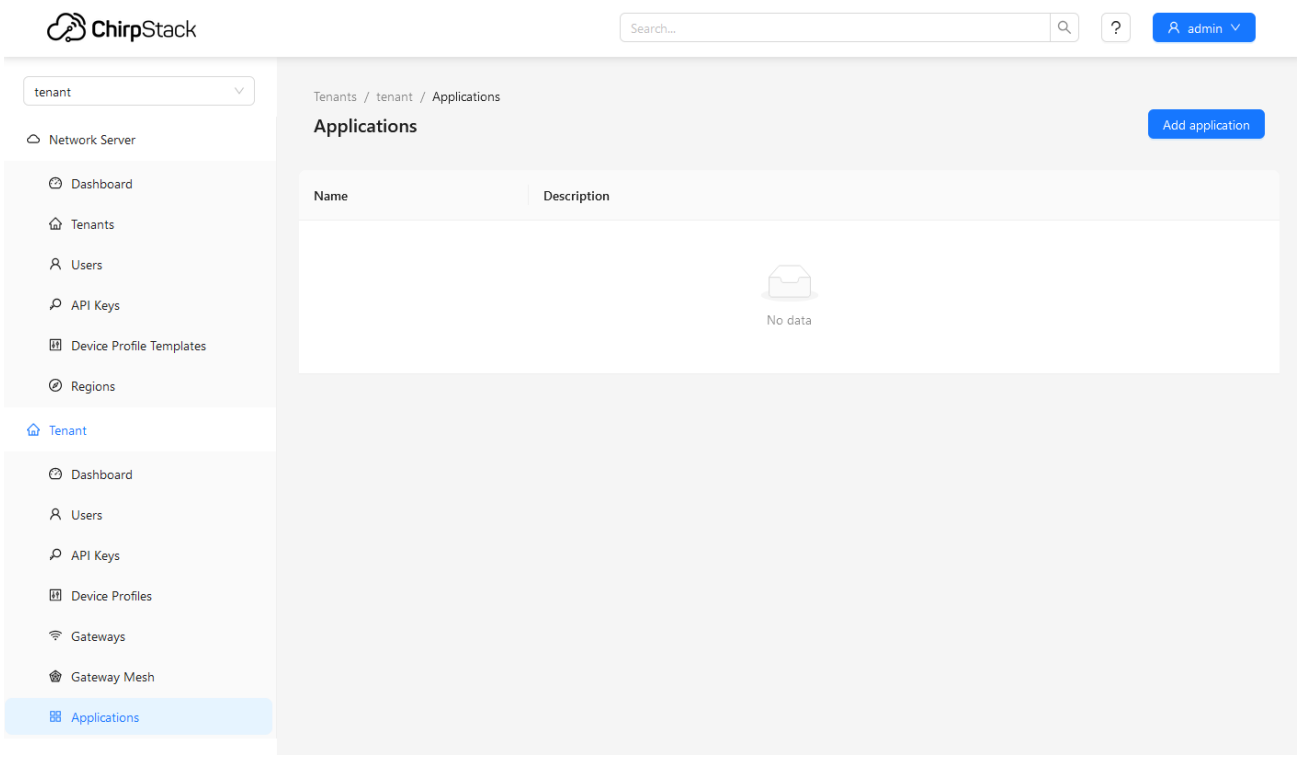
ChirpStack – Add device profile



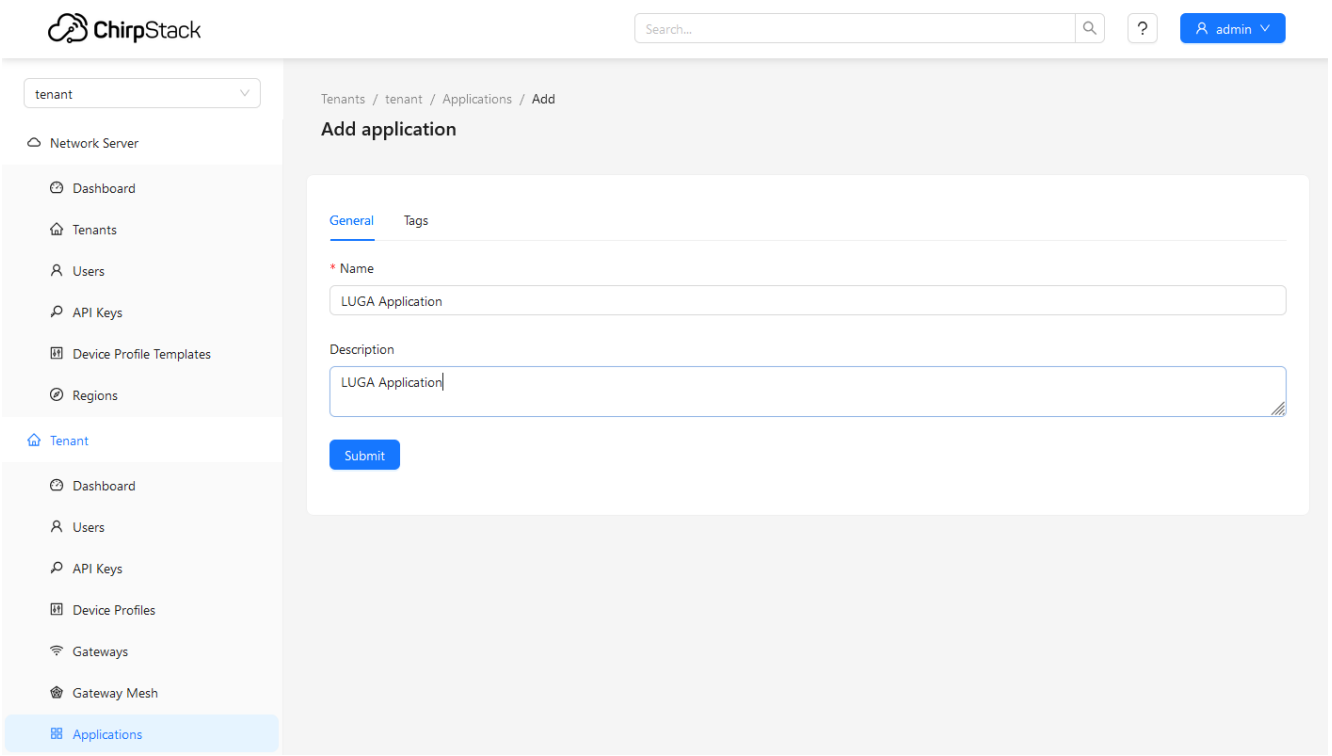
ChirpStack – Add device profile



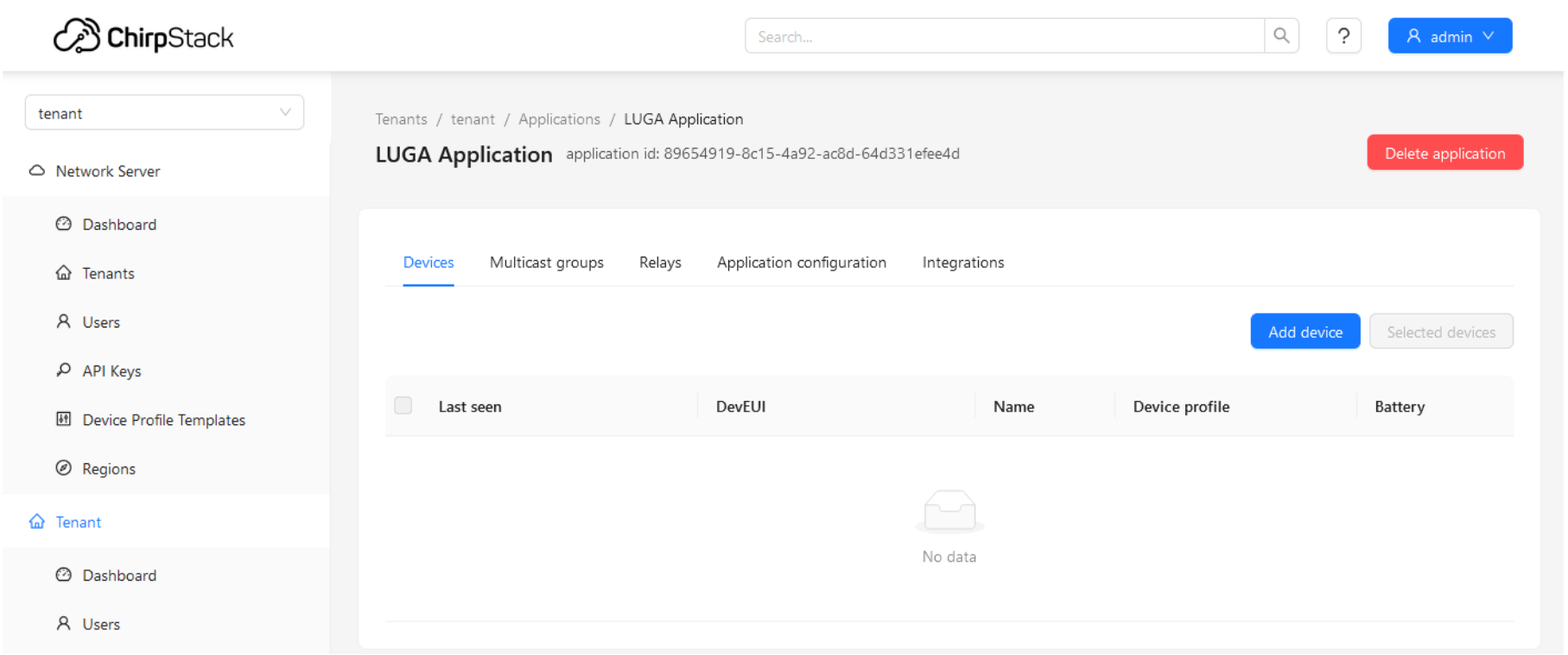
ChirpStack – Add application



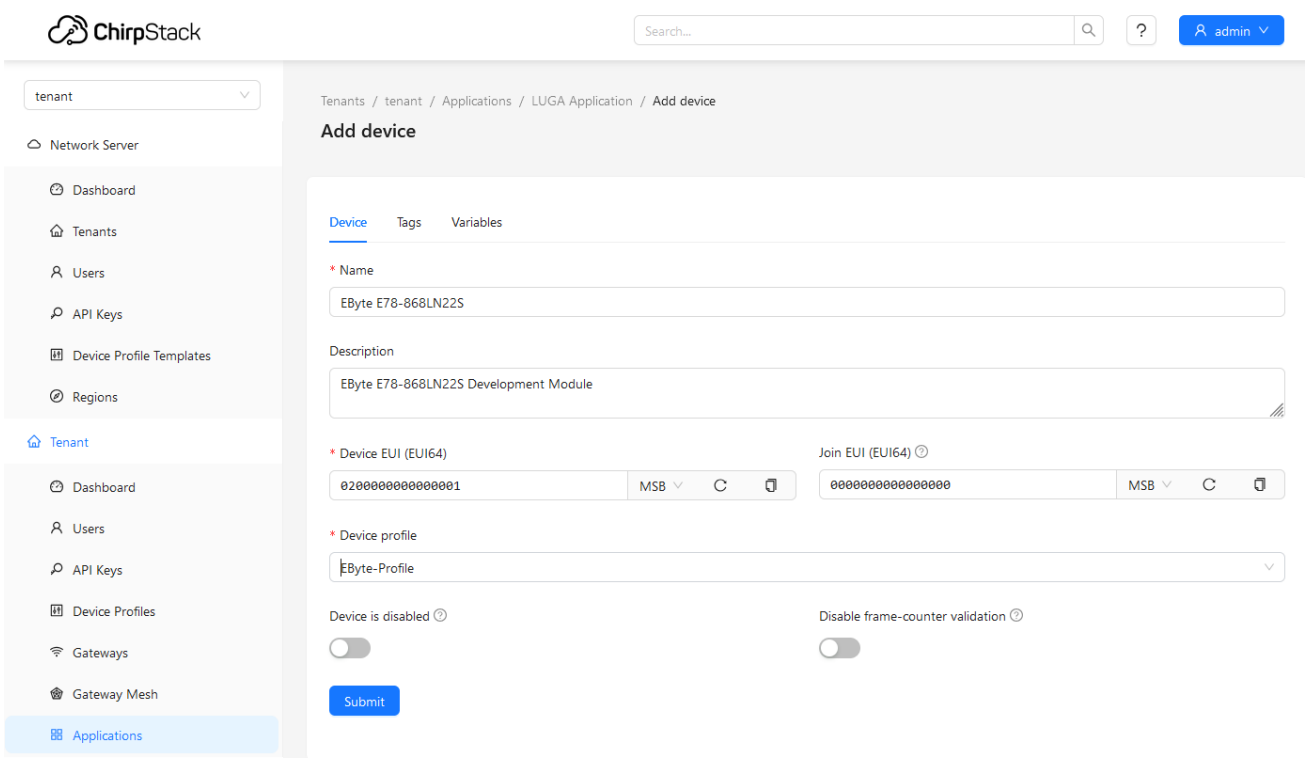
ChirpStack – Add application

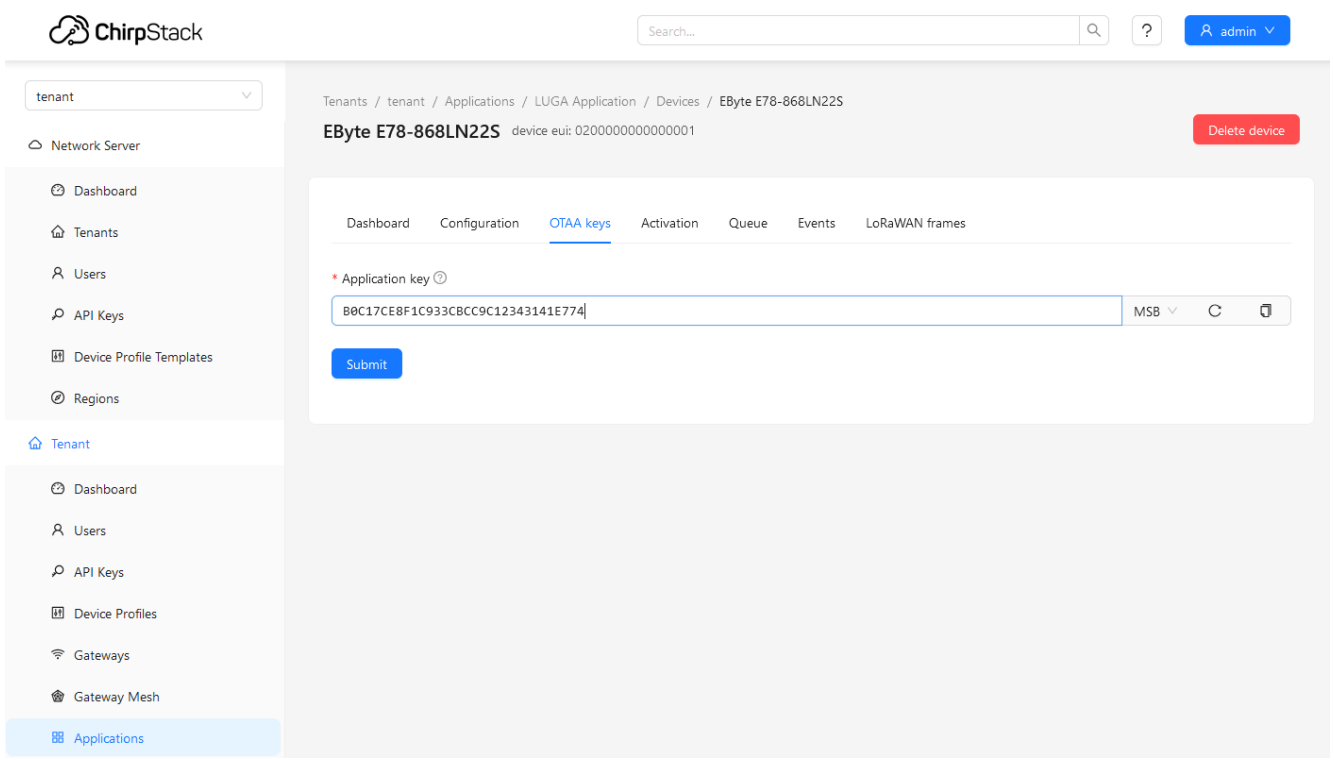


ChirpStack – Add application

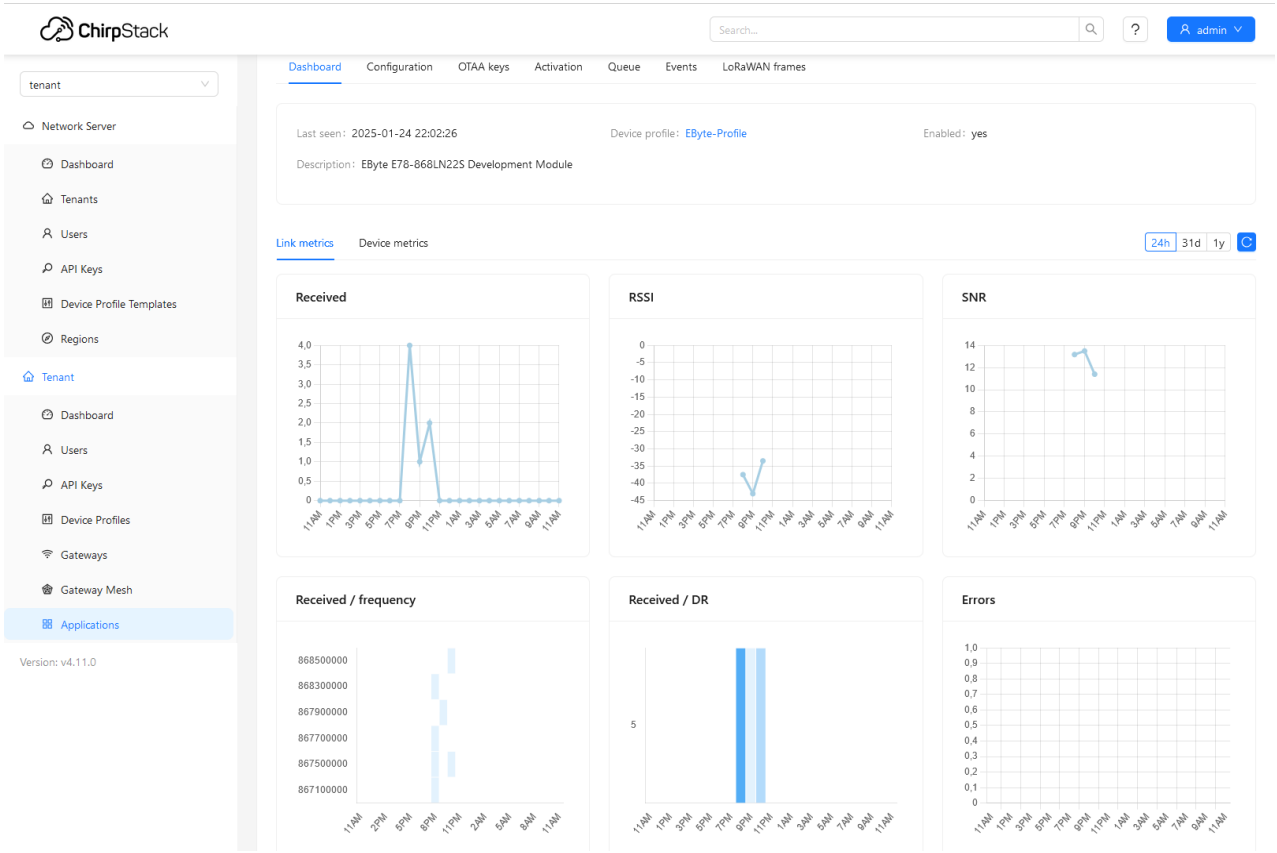


ChirpStack – Add device

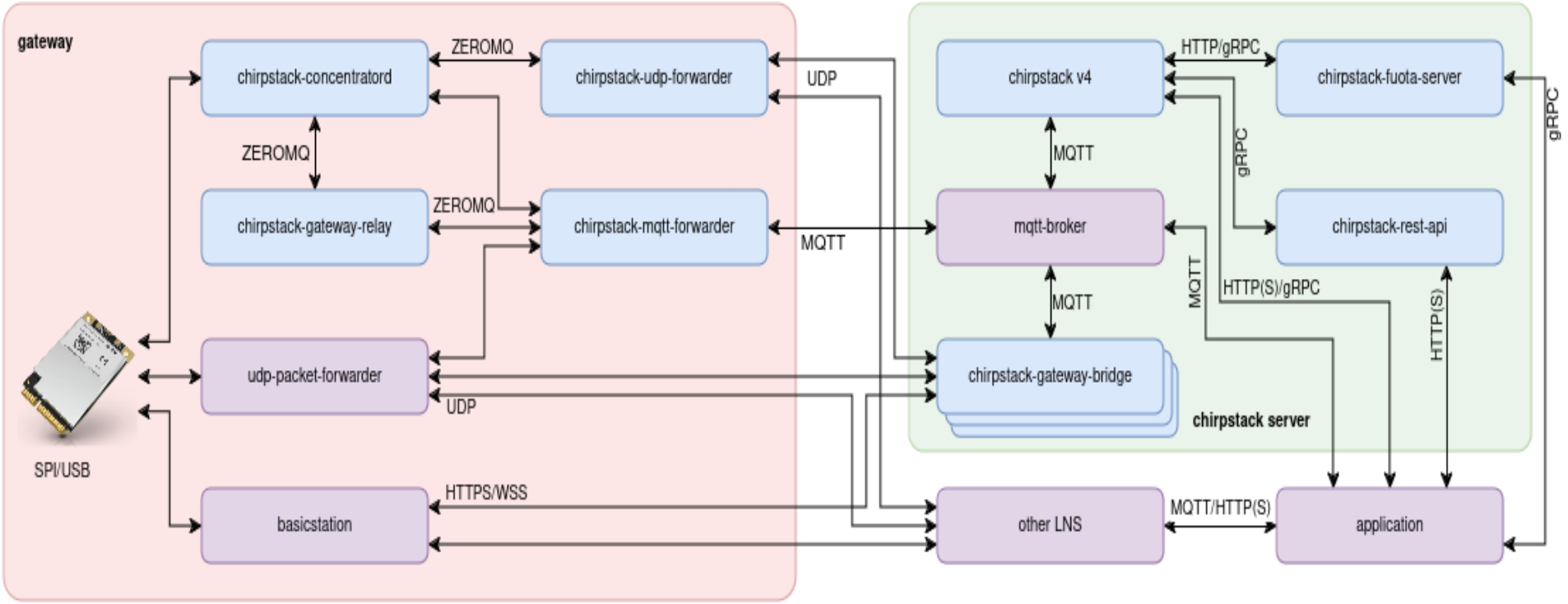



ChirpStack
Search...
admin
tenant
Network Server
Dashboard
Tenants
Users
API Keys
Device Profile Templates
Regions
Tenant
Dashboard
Users
API Keys
Device Profiles
Gateways
Gateway Mesh
Applications
Tenants / tenant / Applications / LUGA Application / Devices / EByte E78-868LN22S
EByte E78-868LN22S device eui: 0200000000000001
Delete device
Dashboard
Configuration
OTAA keys
Activation
Queue
Events
LoRaWAN frames
* Application key
B0C17CE8F1C933CBCC9C12343141E774
MSB
Submit

ChirpStack – Application link metrics



Chirpstack Architecture



Network Server for Self Hosting

- The Things Stack, an open source LoRaWAN Network Server

  https://github.com/TheThingsNetwork/lorawan-stack

- ChirpStack open-source LoRaWAN(R) Network Server

  https://github.com/chirpstack/chirpstack
  https://github.com/chirpstack/chirpstack-docker

- LoRaWAN 1.0.4 Specification in Depth For End Device Developers

  https://learn.semtech.com/course/view.php?id=6

- LoRaWAN airtime calculator

  https://www.thethingsnetwork.org/airtime-calculator

- [1] https://blog.semtech.com/
- [2] https://www.thethingsnetwork.org/docs/lorawan
- [3] https://media.elv.com/file/251814_lpwan.pdf
- [4] https://www2.htw-dresden.de/~jvogt/projektseminar/public/
- [5] https://learn.semtech.com/course/view.php?id=6

Become a LoRaWAN expert

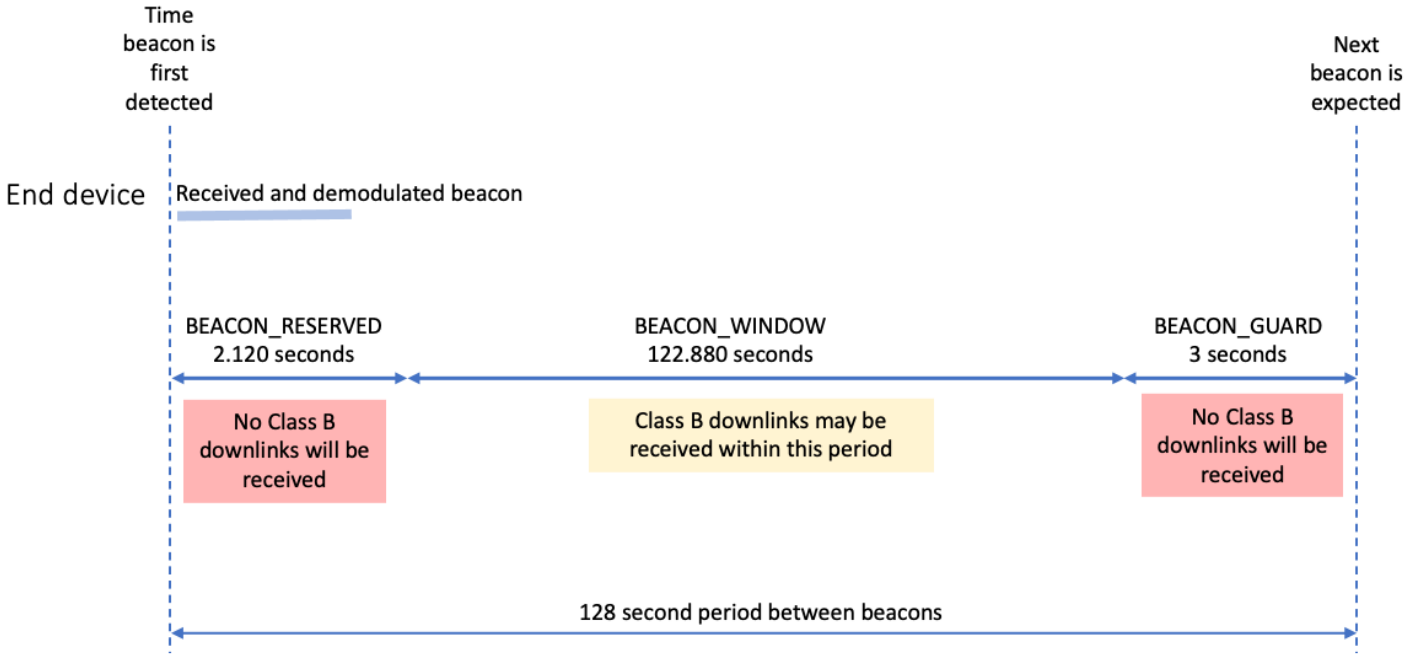
- https://www.univ-smb.fr/lorawan/en/

Quelle: [4]

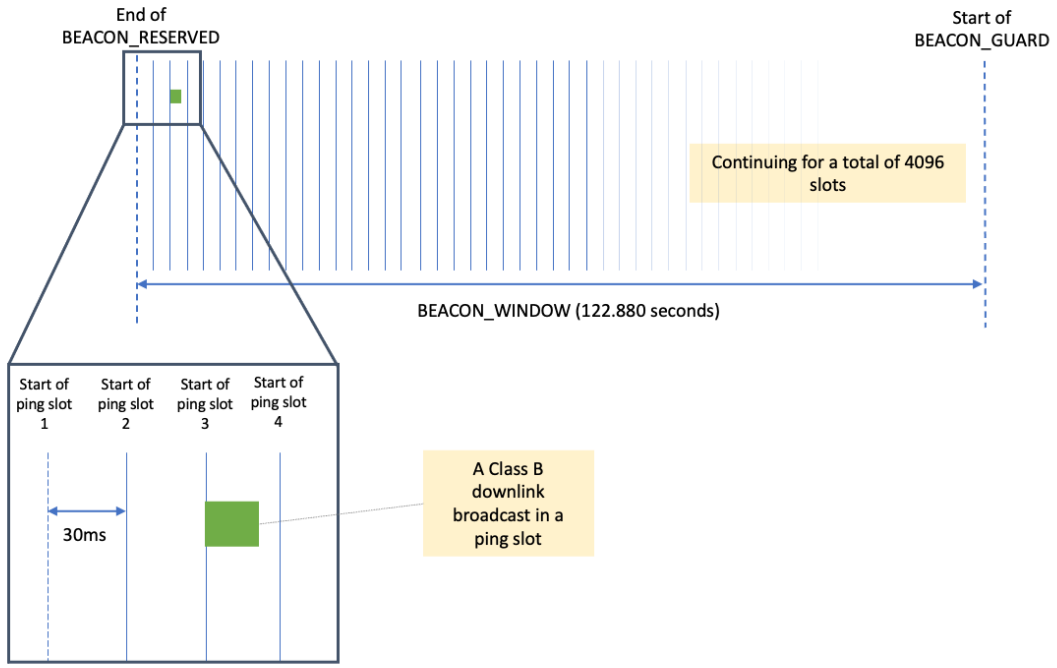
Glossar

- ECB — AES electronic codebook mode encryption
- ISM Band - Industrial, Scientific and Medical Band

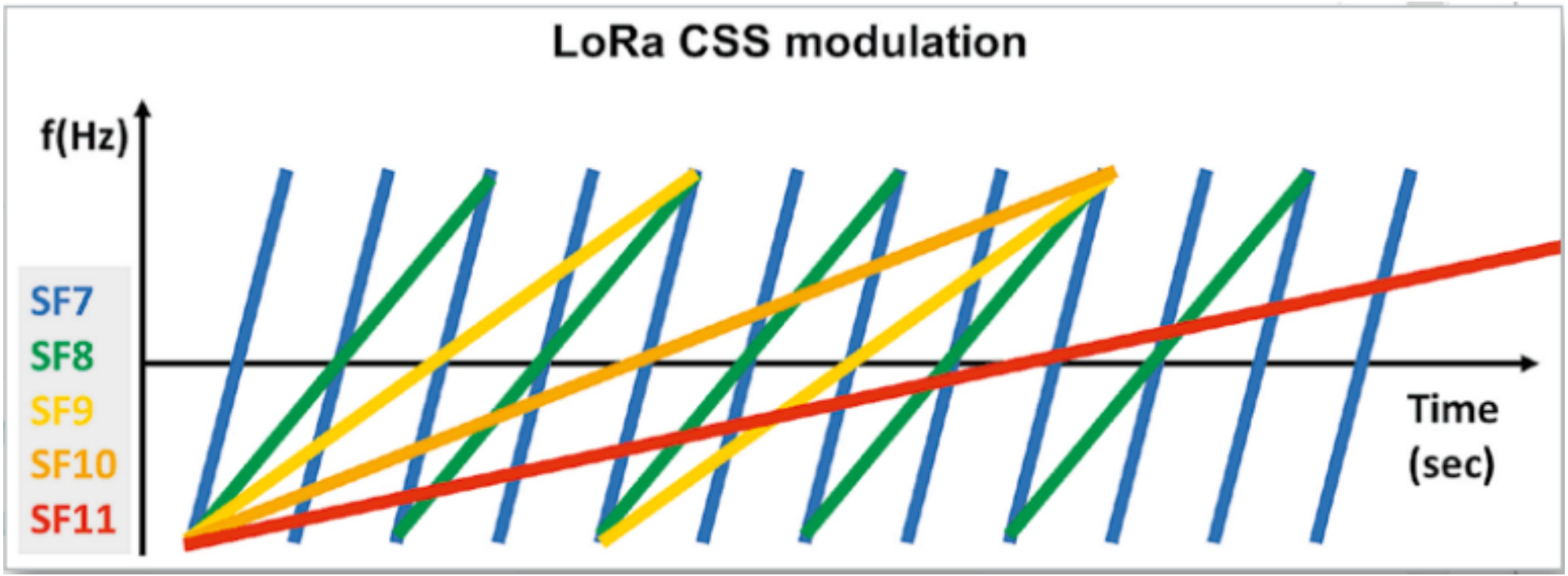
Anhang


Time beacon is first detected
Next beacon is expected
End device
Received and demodulated beacon
BEACON_RESERVED
2.120 seconds
BEACON_WINDOW
122.880 seconds
BEACON_GUARD
3 seconds
No Class B downlinks will be received
Class B downlinks may be received within this period
No Class B downlinks will be received
128 second period between beacons

Anhang

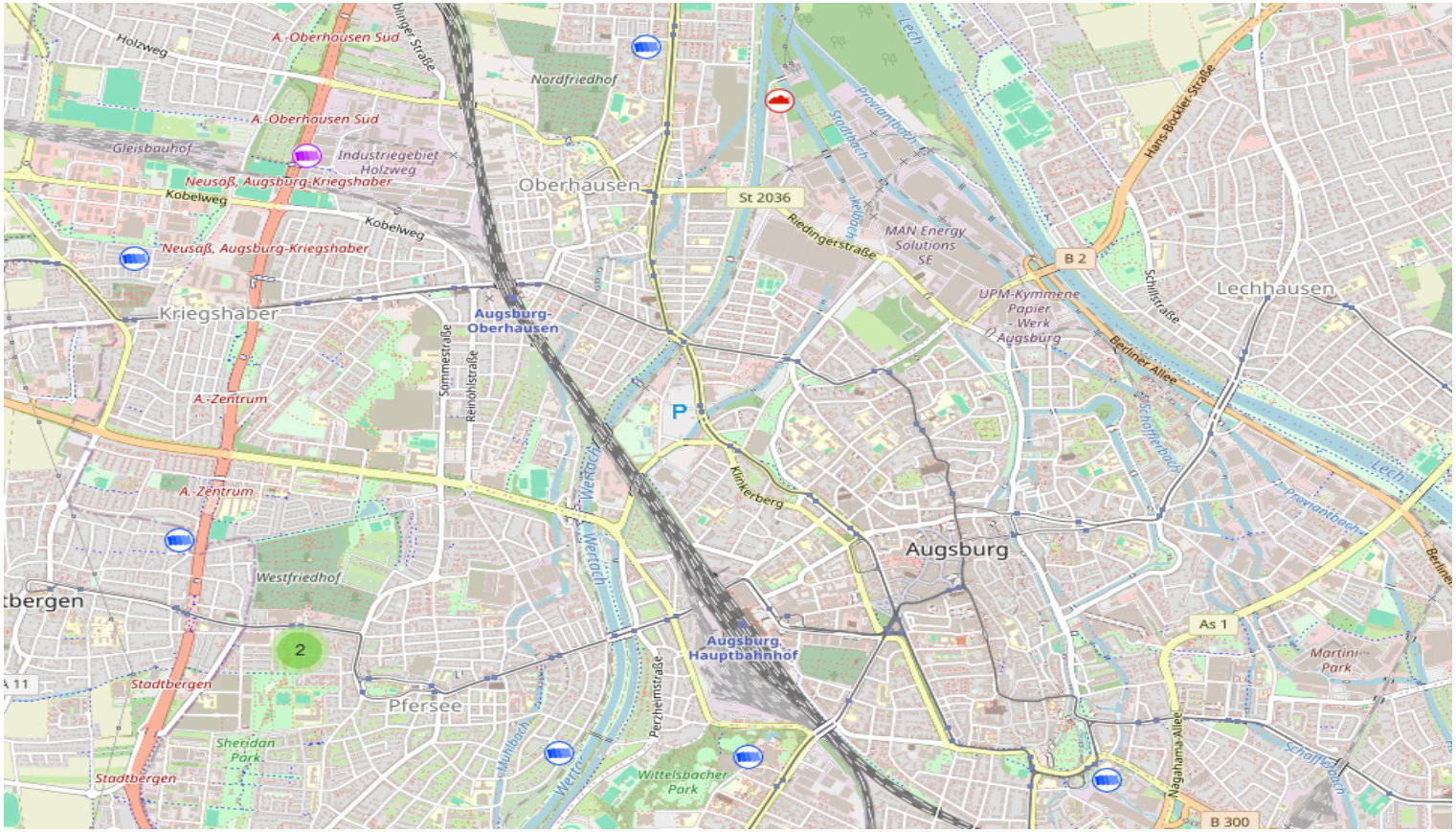

End of
BEACON_RESERVED
Start of
BEACON_GUARD
Continuing for a total of 4096
slots
BEACON_WINDOW (122.880 seconds)
Start of
ping slot
1
Start of
ping slot
2
Start of
ping slot
3
Start of
ping slot
4
30ms
A Class B
downlink
broadcast in a
ping slot

LoRa CSS Modulation



Quelle: [3]

Gatway Map Augsburg



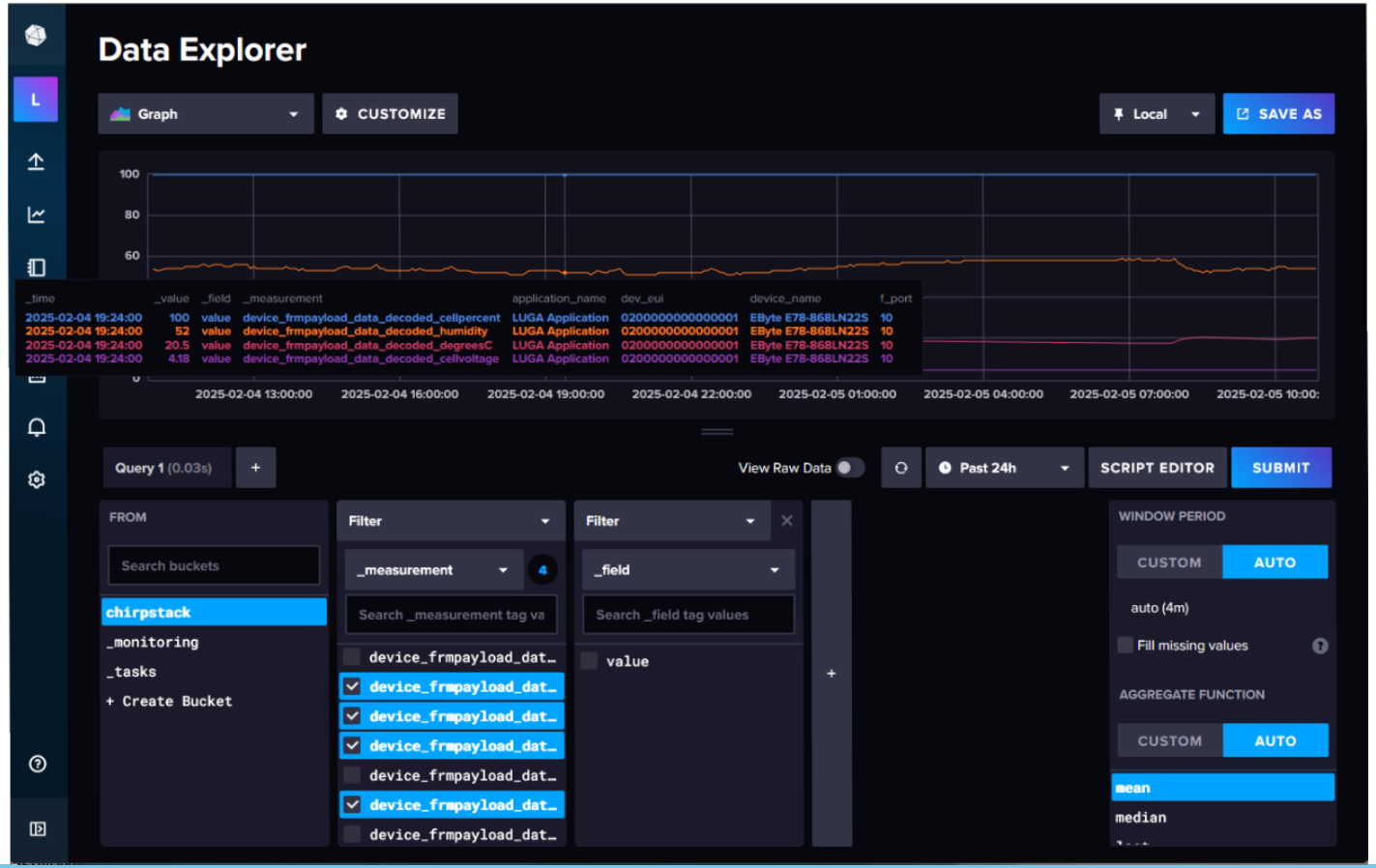
Beispiele LoRaWAN Devices

Dragino Gateway LPS8N

Dragino Node LHT65N – Temp, Hum

LoRaWAN Gateway Beispiel-Installation

Influxdb Data Explorer



Grafana Dashboard