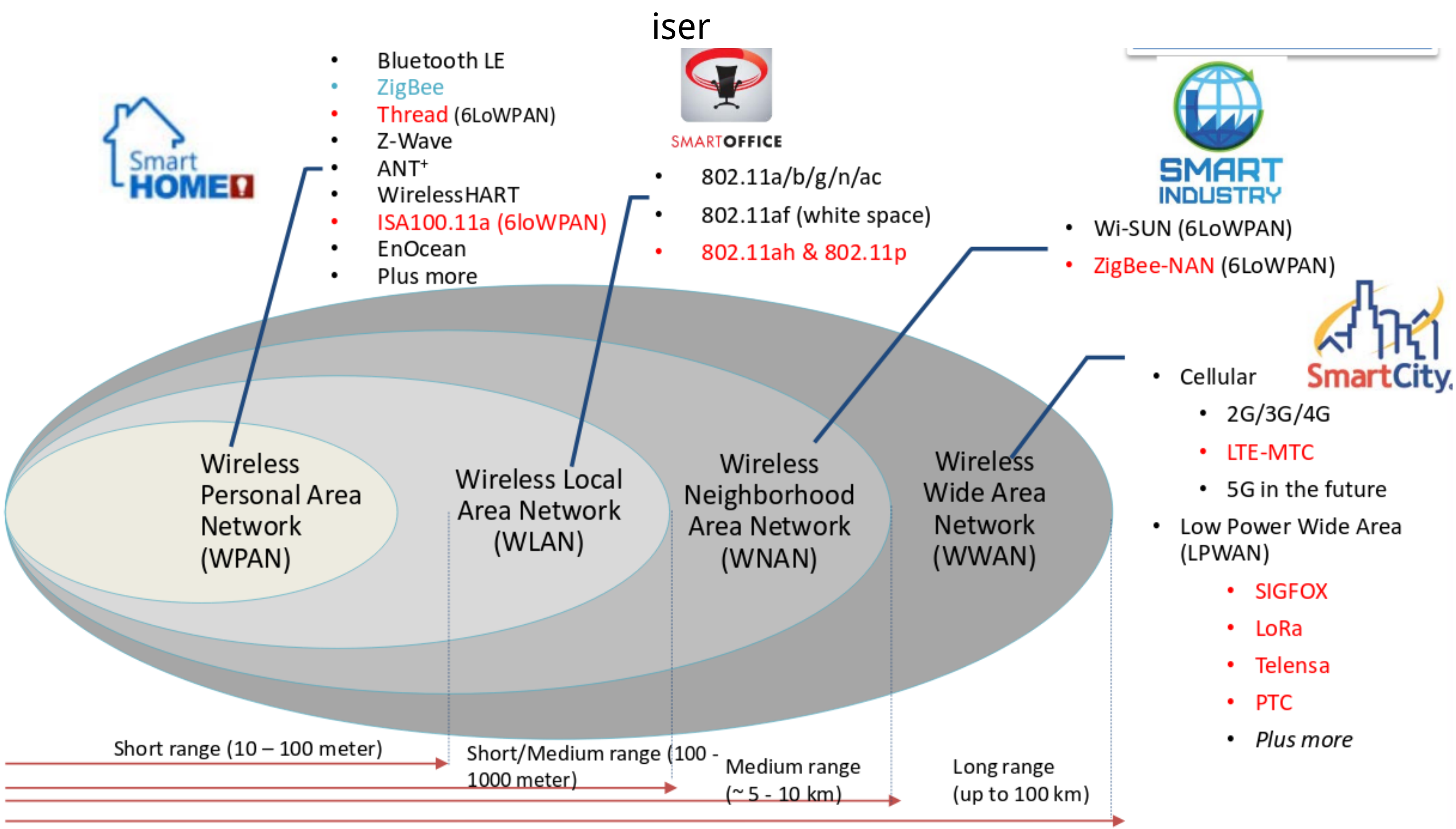
LoRaWAN

IoT protocols



Overview of LoRaWAN

- Designed by Semtech and promoted by the LoRa Alliance

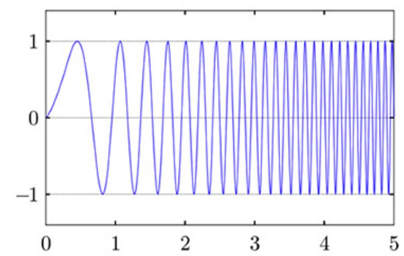
- First release 1.0 of the LoRaWaN specification in 2015
- Latest release 1.1 in 2018
- Based on long range radio communication modulation, LoRa
- Star network topology ⇒ devices talks to the network via gateways

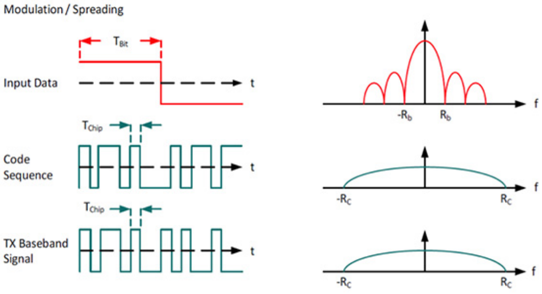
A few words on LoRa

- Long range radio technology
- Spread Spectrum modulation:

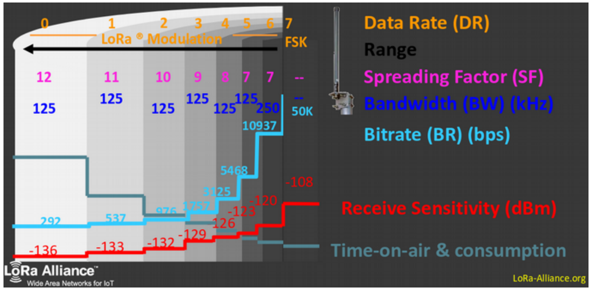
  ⇒ "Chirp Spread Spectrum"

- Very robust to noise

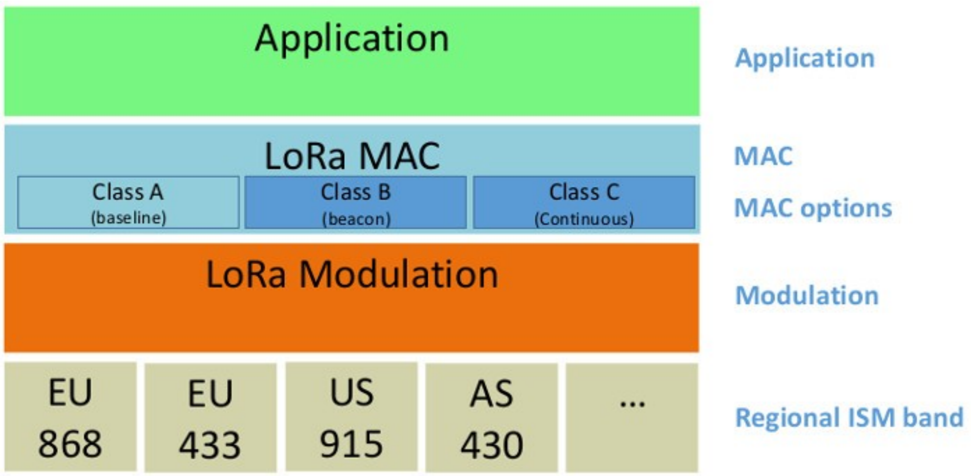




- Raising the spreading factor:
  - increases the range (until several kilometers)
  - decreases the bandwidth
  - increases the time on air



The LoRaWAN protocol



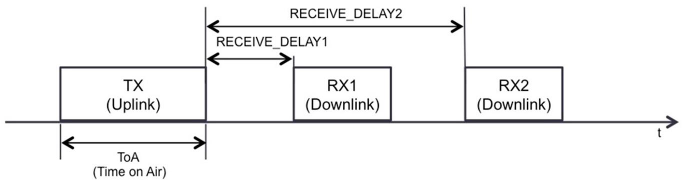
- Different frequency bands depending on the geographical regions
- Use LoRa modulation
- 3 device classes ⇒ A, B & C
- The application layer is directly on top of the MAC layer

Access to the physical layer

Band	Edge Frequencies		Field Power	Spectrum Access	Band Width
g (Note1,2)	863 MHz	870 MHz	+14 dBm	0.1% or LBT+AFA	7 MHz
g (Note2)	863 MHz	870 MHz	-4.5 dBm / 100 kHz	0.1% or LBT+AFA	7 MHz
g (Note2)	865 MHz	870 MHz	-0.8 dBm / 100 kHz	0.1% or LBT+AFA	5 MHz
	865 MHz	868 MHz	+6.2 dBm / 100 kHz	1% or LBT+AFA	3 MHz
g1	868.0 MHz	868.6 MHz	+14 dBm	1% or LBT+AFA	600 kHz
g2	868.7 MHz	869.2 MHz	+14 dBm	0.1% or LBT+AFA	500 kHz
g3	869.4 MHz	869.65 MHz	+27 dBm	10% or LBT+AFA	250 kHz
g4	869.7 MHz	870 MHz	+14 dBm	1% or LBT+AFA	300 kHz
g4	869.7 MHz	870 MHz	+7 dBm	No requirement	300 kHz
Note1: Modulation bandwidth ≤ 300 kHz is allowed. Preferred channel spacing is ≤ 100 kHz. Note2: Sub-bands for alarms are excluded (see ERC/REC 70-03 Annex 7).					

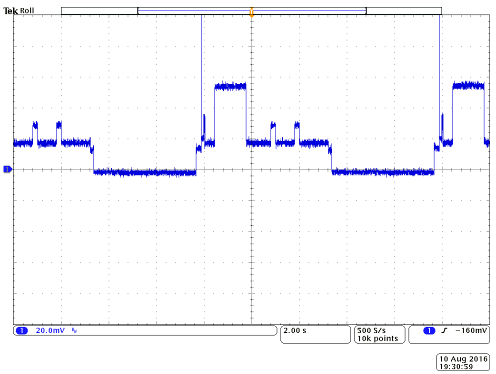
- Public and free **ISM bands** used: EU868 (ETSI), US915, etc
- Bands are divided into **channels** of 3 different widths: 125kHz, 250kHz ou 500kHz
- Time constrained access to the physical layer ⇒ **Duty Cycle** (1% / channel)
- Example: at least 16 channels can be used in EU868 band

Class A & C devices



Class A device

- Can only receive after a send
- Smallest power consumption
- Can be used on battery



Power consumption of a class A device

Class C device

- Always listening: low latency
- More power consumption
- Cannot be used on battery

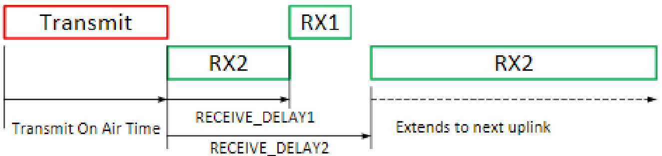
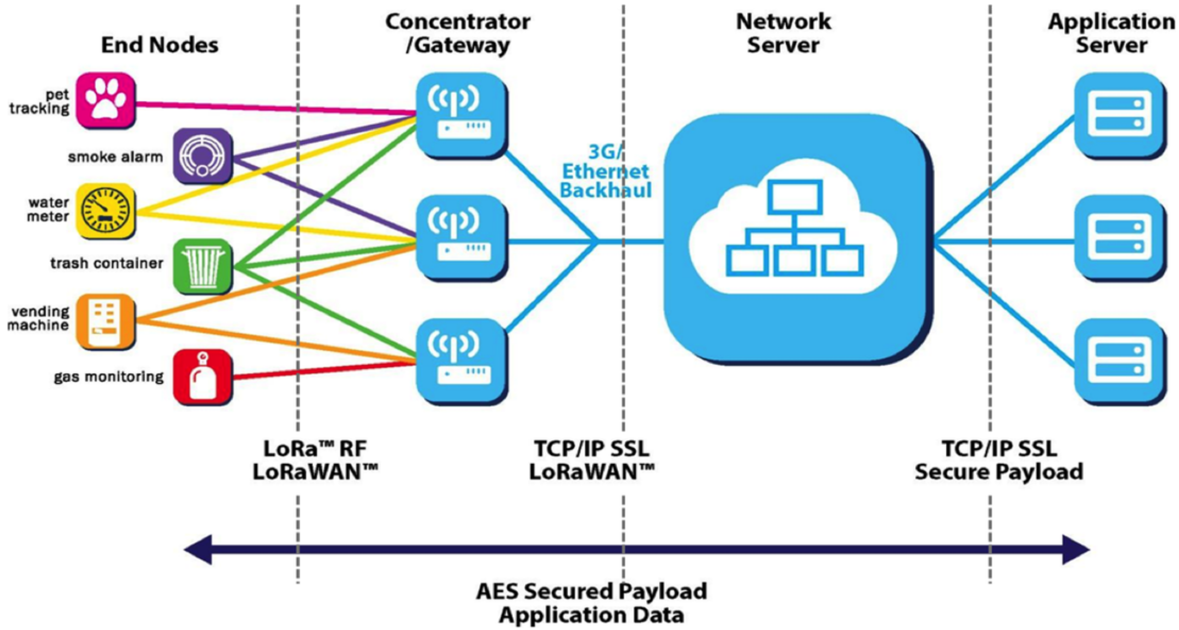


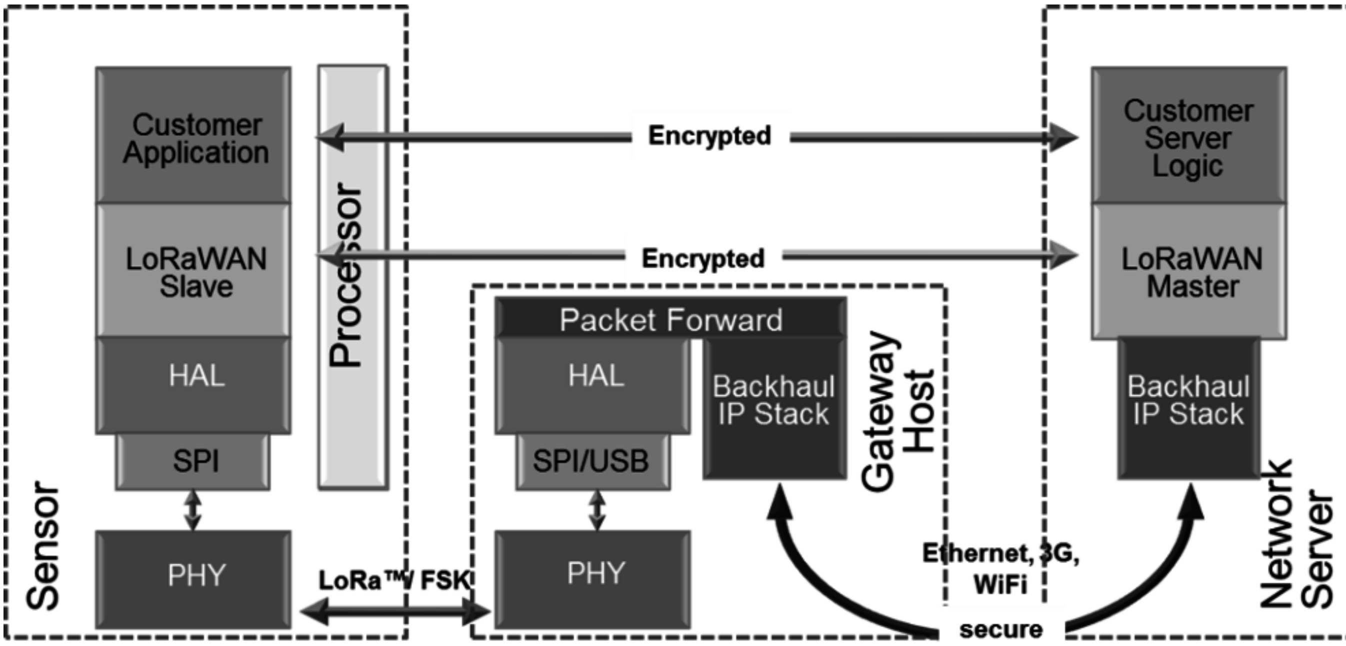
Figure 14: Class C end-device receive slot timing.

LoRaWAN network architecture



- **Devices and gateways** exchange messages using LoRa communications
- **Gateway** are connected to the network server via regular Internet protocols
- Users access their data via an application connected to the network server
- Security of the data is garantueed by **AES** encryption (symmetric keys)

Structural overview of the network parts



Gateway manufacturers

- IMST Lite Gateway https://shop.imst.de
- Kerlink https://www.kerlink.fr/
- Multitech: https://www.multitech.com/

Network servers implementation

- https://www.loraserver.io/ (Opensource)
- https://www.resiot.io/en/

How to program the end-device

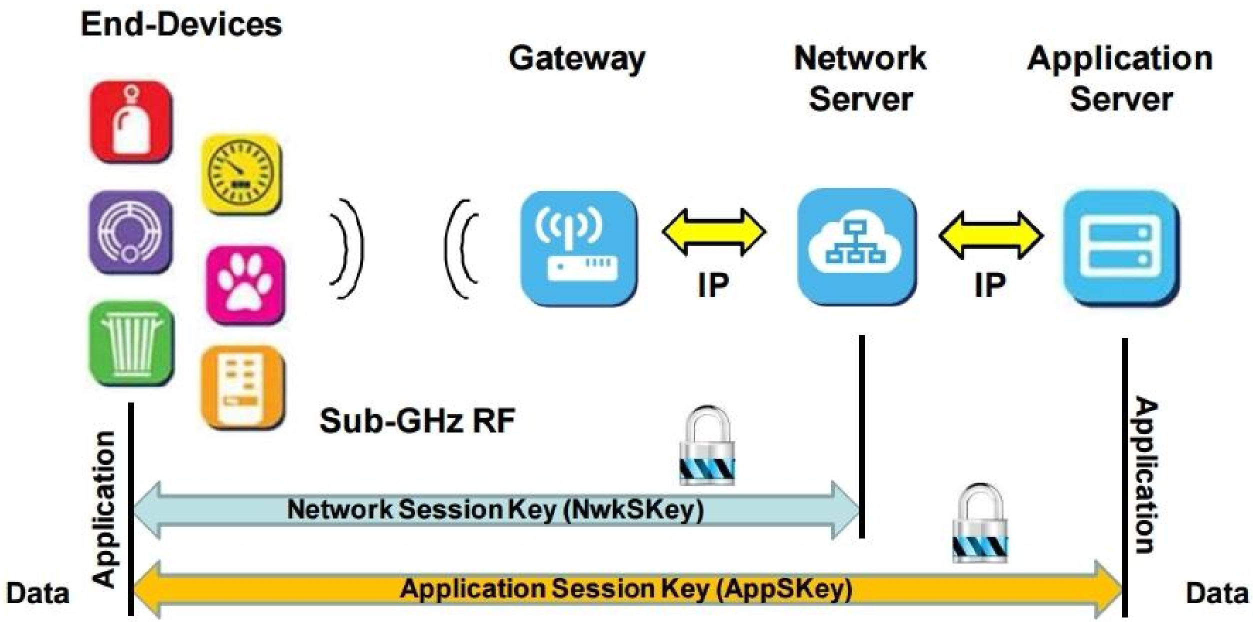
Existing open-source implementations:

- Arduino LMIC https://github.com/matthijskooijman/arduino-lmic ⇒ nearly unmaintained
- Arduino LoRa https://github.com/sandeepmistry/arduino-LoRa ⇒ active
- Loramac-node https://github.com/Lora-net/LoRaMac-node ⇒ reference implementation, used for certification from LoRa Alliance

End-device high-level support (generally based on Loramac-node):

- ARM mbedOS: https://www.mbed.com/en/platform/mbed-os/
- Mynewt: https://mynewt.apache.org/
- Micropython: https://pycom.io/
- RIOT: https://riot-os.org/

Device communication on the network



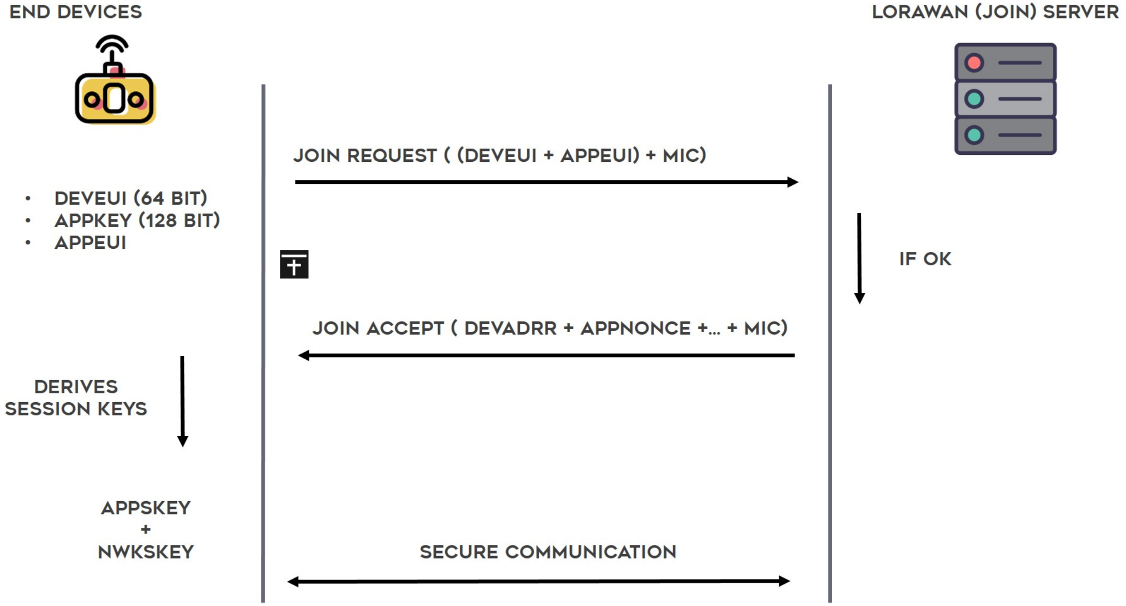
- Every device is identified by a 4 bytes address
- "Network session key" ⇒ used to encrypt the network related data (MAC)
- "Application session key" ⇒ used to encrypt the application related data

Activation procedures

To exchange data, all devices must be activated by the network

⇒ 2 type of activation procedures:

- Over-The-Air Activation(OTAA)
- Activation By Personnalization (ABP)



Activation procedures

- in **OTAA**:
  - Requires Device EUI, Application EUI and Application Key information
  - The device initiates a handshake with the server to get its address and a "nonce" ⇒ the device address changes after each activation
  - The 2 session keys are derived from the application key and the nonce
- in **ABP**
  - Requires Application session key, Network session key and device address
  - No handshake required

Network operators

Lots of public network operators:

- Actility https://www.actility.com/
- Loriot https://www.loriot.io/
- Objenious (Bouygues Telecom) http://objenious.com/
- Orbiwise https://www.orbiwise.com/
- TheThingsNetwork https://www.thethingsnetwork.org/

TheThingsNetwork (TTN)

- The network deployment is **community based**
- Software stack is open-source

- Unlimited access to the backend
  - no device limit
  - no message limit (with respect to the duty-cycle)
  - friendly API (MQTT)

First steps with TTN

1. Create an account
   https://account.thethingsnetwork.org/register

   Manage your gateways and application from your web console:
   https://console.thethingsnetwork.org/

2. Managing your gateways (optional)
   https://www.thethingsnetwork.org/docs/gateways/registration.html

3. Creating an application
   https://www.thethingsnetwork.org/docs/applications/add.html

4. Register your device(s)
   https://www.thethingsnetwork.org/docs/devices/registration.html

Example: using RIOT

- Loramac port documentation
  http://doc.riot-os.org/group__pkg__semtech-loramac.html

- Build and run the test/demo application provided by RIOT

[redacted]

- Configure the device using the shell of RIOT

[redacted]

- Join the network using OTAA activation procedure

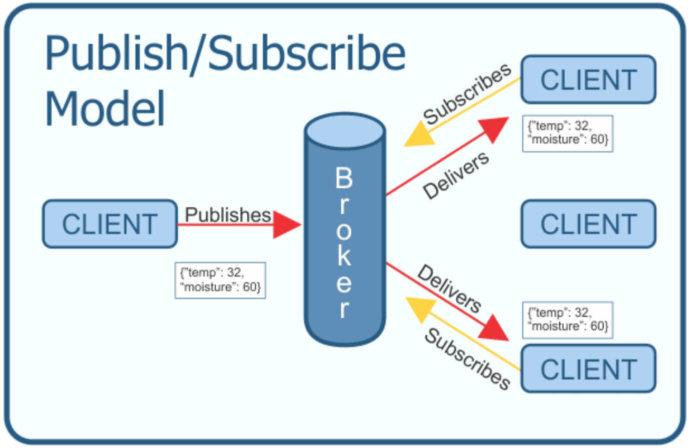
[redacted]

- Send (and eventually receive) messages to the network

[redacted]

The TTN MQTT API

- MQTT protocol uses a publish/subscribe approach

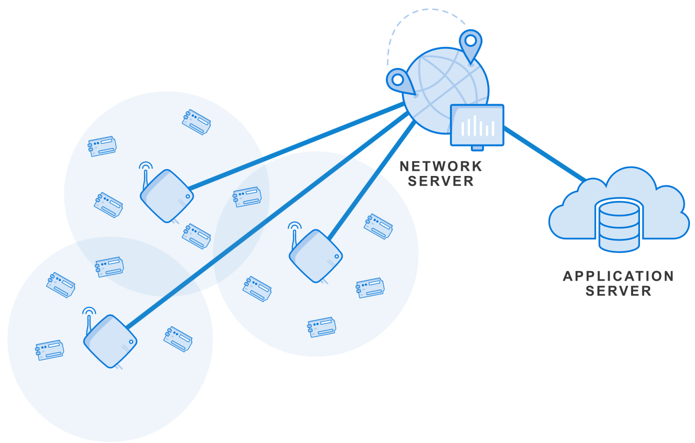


- TTN MQTT API documentation
  https://www.thethingsnetwork.org/docs/applications/mqtt/

- Reference implementation provided by the Eclipse Mosquitto project
  https://mosquitto.org/

- Eclipse also provides a python library: *paho*
  https://www.eclipse.org/paho/

Using the MQTT API



- Listening to upling messages (device to network):

[redacted]

- Sending a downling message (network to device):

[redacted]

Integration with external services

- Use of TTN http and/or MQTT API to retrieve the IoT data
- Super simple to integrate
- Available services:
    - Customizable dashboards with Cayenne
      https://mydevices.com/
    - Location service with Collos
      http://preview.collos.org/
    - Gather and analyze workspace use and sensors with OpenSensors
      https://opensensors.com/
    - Just store your IoT data with TheThingsIndustries
      https://www.thethingsindustries.com/

An example: Cayenne

https://mydevices.com/cayenne/docs/lora/#lora-the-things-network

- Create only dashboards in a few clicks from your LoRaWAN data
- Access your sensor data from anywhere
- Payload format requirement: Low Power Payload (LPP)
  - Library available for python/micropython: https://github.com/jojo-/py-cayenne-lpp
  - Library available for Arduino (C++): https://github.com/sabas1080/CayenneLPP
  - Generic library in C https://github.com/aabadie/cayenne-lpp

Demo