LoRaWAN stands for Long Range Wide Area Network. It’s a standard for wireless communication that allows IoT devices to communicate over large distance with minimal battery usage.

LoRa or LoRaWAN

The term LoRa and LoRaWAN are often used in a mixed fashion but by definition there is a difference. LoRa defines the standard for the physical (layer 1) standard, LoRaWAN defines all that plus the MAC layer and application standards.

LoRaWAN is a wireless communication standard. You could put it in the same category of Bluetooth, GSM, 3G, LTE,… but it’s still different. It has the range of your mobile phone with the flexibility of Bluetooth or WiFi and the battery life of your watch for the cost of a beer.

The main characteristics of LoRaWAN are:

- Long range (>5 km urban, >10 km suburban, >80 km VLOS)
- Long battery life (>10 years)
- Low cost (<$5/module)
- Low data rate (0.3 bps – 50 kbps
  , typically ~10 kB/day)
- Secure
- Operates in unlicensed spectrum
- Localisation support
- Bidirectional

As you can see in the list of characteristics, everything sounds outstanding except when we’re looking at the data rate. Unfortunately, physically limited, wireless communication is always a trade off between distance, speed and power (energy). LoRa was designed with use cases in mind where this data rate should be enough and the other characteristics are way more important.

LoRaWAN was also designed with large service providers in mind. Just as with the cellphone network: a few operators that maintain and control the network and millions of users that exploit the network and do not need to care about the infrastructure. Nevertheless, since LoRa is operating in unlicensed spectrum, it’s perfectly possible to set up your own gateway(s), have a coverage of a few kilometers and run your own network.

Use cases

As mentioned before, LoRa allows for low power, long range but low data rates. Typical use cases can be found in the IoT (Internet of Things), IoE (Internet of Everything) or M2M (Machine to Machine) area. We’re typically looking at (small) sensors/devices/things that are battery operated and communicate limited information on a limited interval.

Some examples of use cases:

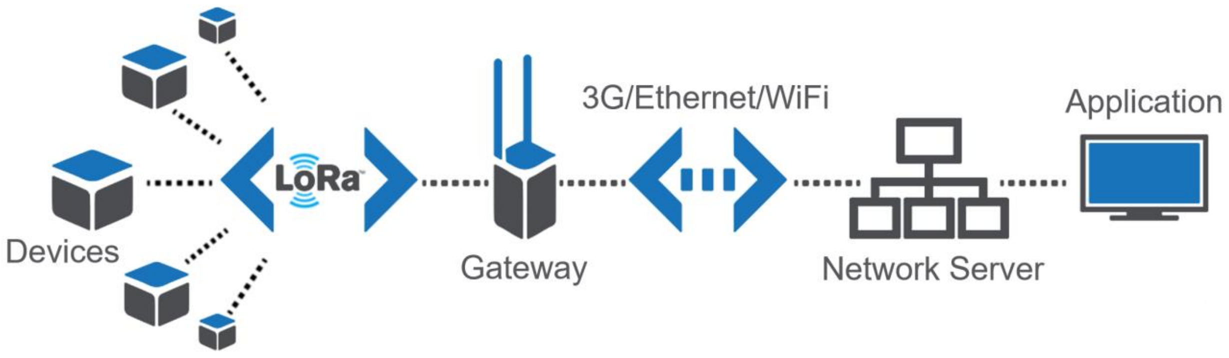
- Metering:
  - Sends 1 to 5 messages a day about the current usage
  - For example: gas/water/electricity

    - >10 years on battery
- Smart parking:
    - Sends a message when a vehicle arrives or leaves a parking spot
    - Can be combined with an app to find a free parking spot
    - No cabling required, low maintenance
- Smart bin:
    - Sends a message when a trashcan is full
    - Optimize route for garbage collection, save labour and fuel
- Smart lighting:
    - Control streetlight status and operation
- Environment monitoring
    - For example: sound, temperature, pollution, radiation, humidity, …
    - Create valuable insights in combination with geolocation
- Asset management
    - Check status and location of various assets
    - Control relays, locks, lights,…
- Healthcare
    - For example: activity/fall detection, personal alarm, surveillance
    - No need for charging, great network coverage
- Tracking
    - Track goods, vehicles, animals

Architecture

From a high level, the architecture of a typical LoRa network looks like the following:

Devices -> LoRa radio -> Gateway -> 3G/Ethernet -> Network Server -> Application



For upstream messages, for example a sensor that sends information to an application, the flow is from left to right. The sensor value (payload) gets encrypted and gets transmitted over LoRa radio. One or more gateways receive the message and forward it over another network (typically 3G or Ethernet) to a Network Server. The Network Server routes the message to the correct end application.

For downstream messages, for example a signal to turn on a light, the flow is from right to left. Upstream messages are initiated by the device itself and downstream by the end application. Since LoRa is designed with as low energy usage as possible, not all devices are always listening for incoming messages. This depends on the device classes.

Devices

LoRaWAN devices can be anything that sends or receives information, there is no real definition for them but usually we're speaking about sensors, detectors, actuators.

Some examples of devices:

- Adeunis LorRaWAN demonstrator (http://www.adeunis-rf.com/en/products/lorawan-products/lorawan-demonstrator-by-adeunis-rf)
    - This is a device to demonstrate LoRa
    - GPS / Temperature / Accelerometer sensor
    - Self-powered and rechargeable
- NKE Watteco smartplug
    - Ability to turn connected device on/off
    - Measures power, voltage and frequency
- Abeeway
- Raspberry Pi or Arduino
    - Combined with sensors: temperature, pressure, humidity, light, sound, GPS Location, accelerometer, motion detector, air quality, magnetic switch, …

Device classes

- A: Can only receive a message at the time the device is sending a message
- B: Same as A but listens for incoming messages on regular intervals
- C: Continuously listening for incoming messages

Logically, class A device can last the longest on battery and class C devices are typically not battery powered.

The LoRaWAN standard describes two different types of messages: MAC messages, to control the radio and network and data messages, the actual payload that is application/device specific. Since we're limited in the time and number of messages we can send over the air, MAC messages can be piggybacked (send together) with data messages and multiple MAC messages can be sent in one time.

Device addressing

As with most networking standards, devices need some kind of address and identification to be able to contact them and differentiate them from each other. LoRa uses the following addressing.

- DevEUI: Device unique hardware ID: 64 bits address. Comparable with a MAC-addresss for a TCP/IP device.
- DevAddr: Device address**:** 32 bits address assigned or chosen specific on the network. Comparable with an IP address for a TCP/IP device.
- AppEUI: Application ID**:** EUI64 address format. Uniquely identifies the application provider of the device. Then AppEUI is stored in the end-device before the activation procedure is executed.
- Fport: identifies end application/service. Port 0 is reserved for MAC messages. Comparable with a TCP/UDP port number for a TCP/IP device.

LoRa radio

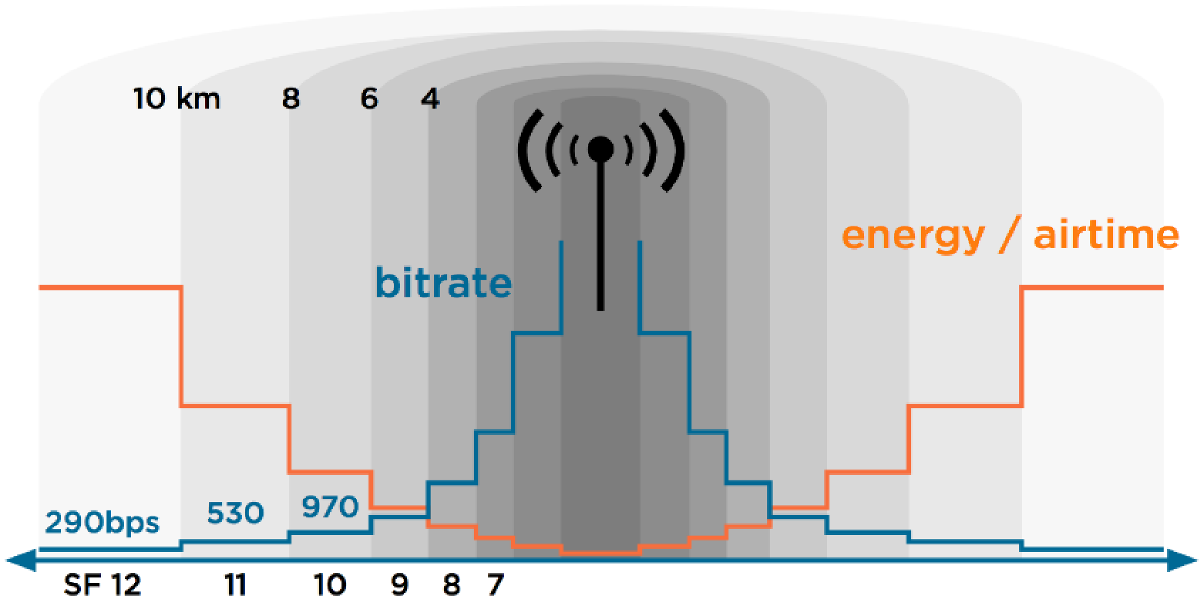
LoRa is using RF signals to communicate. It's operating in unlicensed spectrum. This means that everybody can freely use this band without paying or getting a license as longs as you follow some regulations. Specifically, LoRa is operating in the ISM (Industrial, Science, Medical) band. Unfortunately, the ranges in the ISM band are geographically different.

To accomplish the goals of LoRa in terms of range and data rate, the following frequencies are used: Europ: 868Mhz and 433 Mhz, US: 915Mhz and AS: 430 Mhz. Obviously this makes gateways and devices for Europe incompatible with the ones manufactured for the US, etc. A big disadvantage for LoRa .

LoRa is using different techniques to improve data reliability. I will try to mention the most important ones:

- Spread Spectrum: This a technique originally designed for military applications where the actual information (for example one bit) is spread over a larger frequency. By doing this, the signal to noise ratio is very small and the signal (the bit sent) is much more resistant to noise, interference or jamming signals.
- ADR: Adaptive Data Rate: Dynamically change the data rate and transmit power in function of the signal quality and distance to the gateway. Slower transmission (higher spreading

factor) allows for a longer and more reliable range:



- FEC: Forward Error Correction: Add redundant information (recovery/parity bits) to each message to be able to correct small errors.

Gateway

The gateway, also called modem or access point, is the device that is receiving all LoRa radio sent by the devices in its range. By design, there isn't really an association between a device and a specific LoRa gateway. Every gateway in the range of the RF signal sent out by a device will pick up the signal and process it, even if this LoRa gateway is part of a network that doesn't know the device that sent the message.

Some examples of LoRa gateways currently available:

- Kerlink LoRa IoT station:
- Multitech Multiconnect Conduit with LoRa module
- Link labs LL-BST-8 LoRa
- LoRa mCard in combination with a Raspberry Pi or similar
- IMST iC880A in combination with a Raspberry Pi or similar
- Cisco IR910
- Cisco IXM

Network Server

After a device sent a message that was received by a LoRa gateway, it will most likely be forwarded to a Network Server. This component is the most intelligent part in the LoRaWAN network. The Network Server is responsible for the following tasks:

- Aggregate the incoming data from all LoRaWAN gateways in its network and all sensors in the range of them.
- Route/forward incoming messages to the correct end application
- Control LoRa radio configuration to the gateways
- Select the best gateway for downlink message in case multiple gateways are in the range of one device
- Downlink buffering: store messages until a LoRa class A or B device is ready to receive a message
- Remove duplicates: since there is no association between device and gateway there is a good chance that multiple gateways pick up the same message so duplicates need to be filtered
- Monitor the devices and gateways

Some examples of Network Server software:

- Actility Thingpark: https://www.actility.com/products/
- Loraserver: https://github.com/brocaar/loraserver
- Semtech: https://semtech.force.com/lora

In most cases, the network server will route/forward the message from a certain id and fport combination to a predefined application. Usually this is done by either forwarding it to a HTTP(S) web service or putting it into an MQTT queue.

Application

Last item in the flow or architecture is the actual application. This is the application that the manufacturer or developer writes to parse the messages sent by the devices relevant for this application. This application uses the actual data (payload) sent by the device and interprets/uses it.

Security

LoRaWAN is designed with security in mind.

The basic idea is that communication should be secure on multiple levels. A Network Server does not need to be able to read the actual contents of the message if it's not relevant for the network or infrastructure. Therefore, there are two different keys in the picture during normal message exchange:

- The network session key (NwkSKey) is used to encrypt the whole frame (headers + payload) in case a MAC-command is sent. When data is sent, this key is used to sign the message which allows the network server to verify the identity of the sender

- The application session key (AppSKey) is used to encrypt the payload in the frame. This key doesn't need to be known by the Network Server to be able to know where to forward the message to. The application server then decrypts the information using the same key.

Security: joining the network

Until here, things are still quite understandable but the way that a device can join the network makes it more complicated. Since you wouldn't want that for every application/use case, that the user is responsible for configuring network access/subscriptions/service provider choice but you also wouldn't want any type of device to be tied to a specific network there are basically two ways of activating a device on a LoRa network.

The goal of this activation process is to get a NwkSKey, AppSKey and NwkAddr. Once these are received or configured, the rest of the communication is exactly the same for both of the activation methods.

<u>Activation By Personalization (ABP)</u>

ABP is the most simple way to activate a LoRa device on a network. With ABP, the supplier of the device agrees with the network provider and buys/reserves a range of DevAddr from a network provider and pre-provisions his devices with this. Once the device is turned on, it's immediately ready to send data. There is no over the air join process.

Steps in the progress:

1. Manufacturer/supplier of LoRa device buys connectivity from the network operator. AppSKey is generated in advance.
2. Service provider is delivering a NwkSKey and DevAddr for each DevEU.
3. Before shipping devices, they are pre-provisioned with: NwkSKey, AppSKey and DevAddr
4. On first use, there is no need to join the network and the device can communicate immediately as they already know the necessary information

<u>Over The Air Activation (OTAA)</u>

OTAA is a little more complicated but allows for devices to join any network in the range and doesn't require a special agreement between network operator and the device supplier. As with ABP, the goal of the process is to become the NwkSKey, DevAddr and AppSKey for each device prior to sending actual data.

To retrieve this information in a secure way, there is an extra key involved which is pre-provisioned in the device: the Appkey. Personally I find the name very confusing as it's pretty similar to AppSKey. The Appkey is a different 128 bit key.

Steps in the OTAA join process:

1. LoRa Device sends JOIN_REQUEST (signed with AppKey). The join request contains the following information: AppEUI, DevEUI, DevNonce.
   DevNonce is a randomly generated number.
2. The Network Server receives the JOIN_REQUEST and calculates AppSKey and NwkSKey based on: AppKey, AppNonce, NetID and DevNonce.
   As with DevNonce, the AppNonce is another randomly generated number.

3. Network Server generates JOIN_ACCEPT and includes AppNonce
4. Device receives JOIN_ACCEPT (encrypted with AppKey). The JOIN_ACCEPT contains the following information: AppNonce, NetID, DevAddr, RFU, RxDely, CFList
   Since at this point both the Network Server and LoRa device have the same information, the LoRa device can equally calculate the NwkSkey and AppSKey