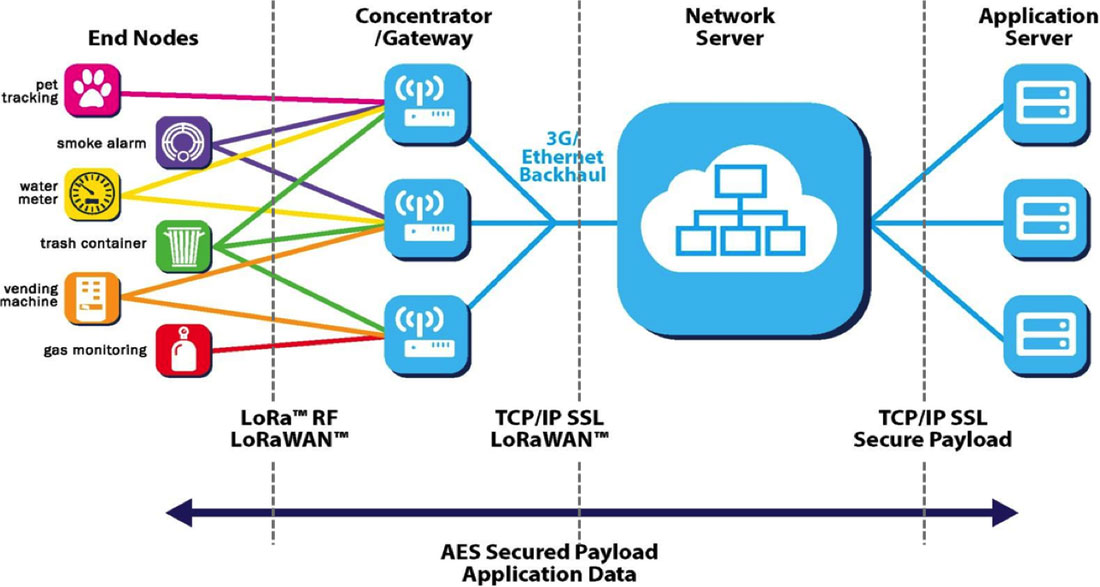
**Complete LoRa transmission chain: from the data to the browser**

Introduction

The following document will explain how the payload from an end-device (also called ‘node’) passes through different elements of the LoRa/LoRaWAN network and reaches the database of a server where the end-user can see the information with his/her we browser. This document will also explain each element of this transmission chain and how each of them behaves, what it does, what it adds to the transmission, …



**Fig.1: A simplified view of a LoRa transmission [1]**

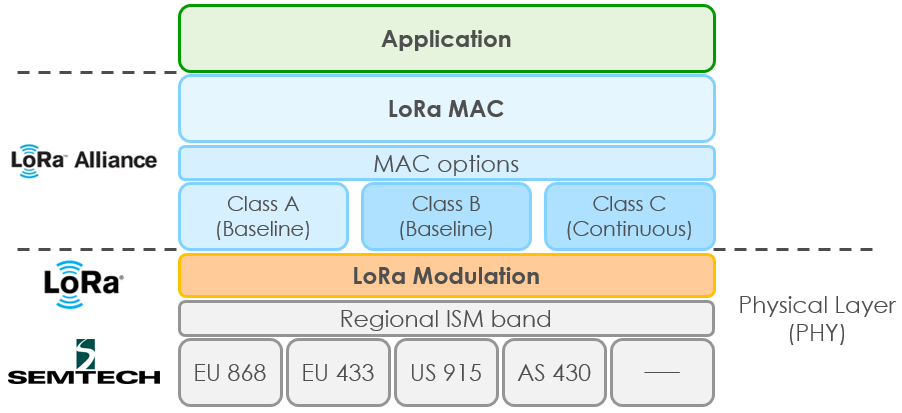
Difference between LoRa and LoRaWAN [2] [3]

Before getting into the subject, let’s briefly explain the difference between the two terms.

LoRa is the PHY (physical) layer of the OSI model that uses Chirp Spread Spectrum modulation in order to send data. This modulation is wireless and allows for low-power emitters (usually battery life >1 year) and high range (1km – 15km approx.).

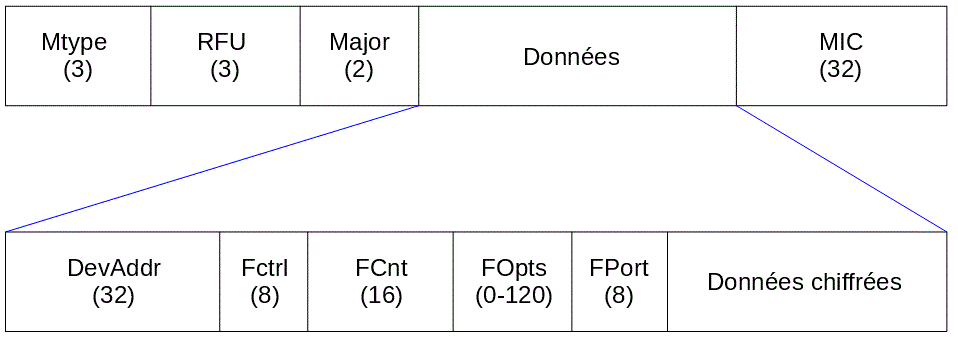
LoRaWAN is a network protocol that uses LoRa for communications. LoRaWAN also adds a MAC layer which includes network registration, addressing, encryption, decryption, message integrity, the possibility to use commands (called MAC commands) for various purposes (ex : join request, confirmed data up, join accept, …). In addition, LoRaWAN adds parameters to control the LoRa transmission (Spreading Factor, Coding Rate, Adaptive Data Rate, …). In the OSI model, LoRaWAN would cover the transport layer, IP layer and MAC layer.

It is possible to use only LoRa modulation without using LoRaWAN, but it wouldn’t be practical because of the (very) limited functionality of the LoRa modulation alone.



**Fig.2: OSI model of a LoRa/LoRaWAN transmission [4]**

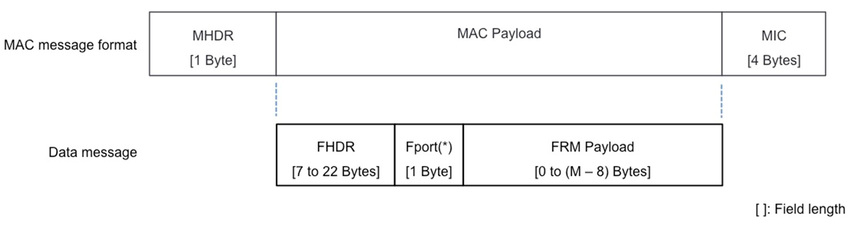
LoRaWAN data frame [5] [6]



**Fig.3: LoRaWAN data frame (field’s size is in bits) [5]**

The figure above shows the fields (the lengths are in bits) of the LoRaWAN data frame :

* Mtype (Message Type): indicates the kind of message (uplink or downlink)
* RFU: this field is reserved for future implementation
* Major: the version of the LoRaWAN protocol that is used
* MIC (Message Integrity Control) : used to control the integrity of the packet
* DevAddr (Device Address): the address of the end-device in the network
* Fctrl (Control Field): allows to adapt the data rate and is used for ACK messages
* FCnt (Counter Field): a counter that increments for every sent packet
* FOpts (Options Field): this field is used to pass the MAC commands
* FPort: (Port Field) application or service port where the packet belongs to



**Fig.4: LoRaWAN data frame (field’s size is in bytes) [6]**

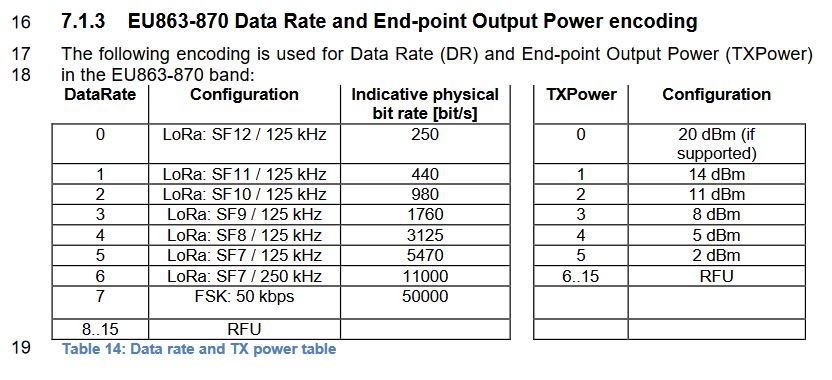
Usually, 13 bytes are added to the packet (if the user didn’t specify any options) [7] [8]:

* MHDR (Mac Header): 1 byte
* MIC (Message Integrity Code): 4 bytes
* DevAddr (Device Address): 4 bytes
* FCtrl (Control Field): 1 byte
* FCnt (Counter Field): 2 bytes
* FPort (Port Field): 1 byte

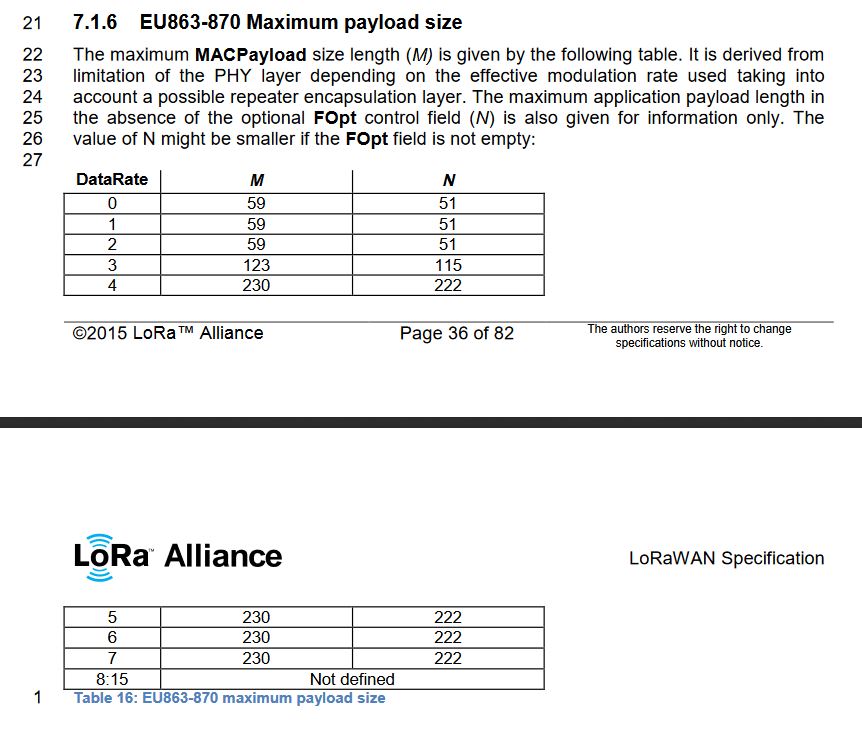
(Remark: FHDR stands for Frame Header, and it contains DevAddr, FCtrl, FCnt and FOpts.)

The encrypted payload’s size varies from 0 to (M-8) bytes, where M is the maximum size of the MAC payload that changes with the data rate (the faster it is, the higher the value M is).

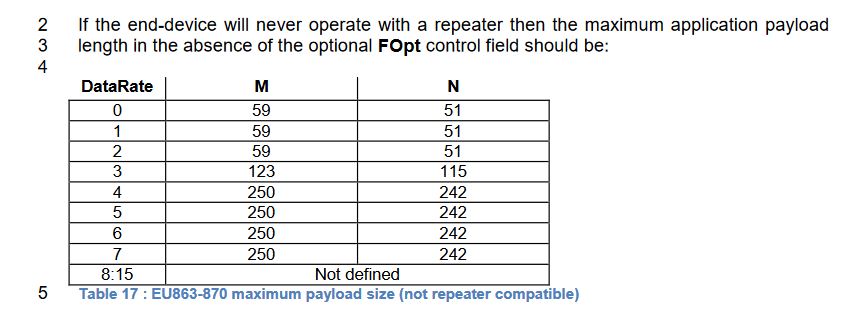
The figure below shows the correspondence between the DataRate and the spreading factor. It is good to remind that when the spreading factor increases, the range increases but the data rate is slower.



**Fig. 5 : Correspondence between Data Rate and physical configuration [9]**



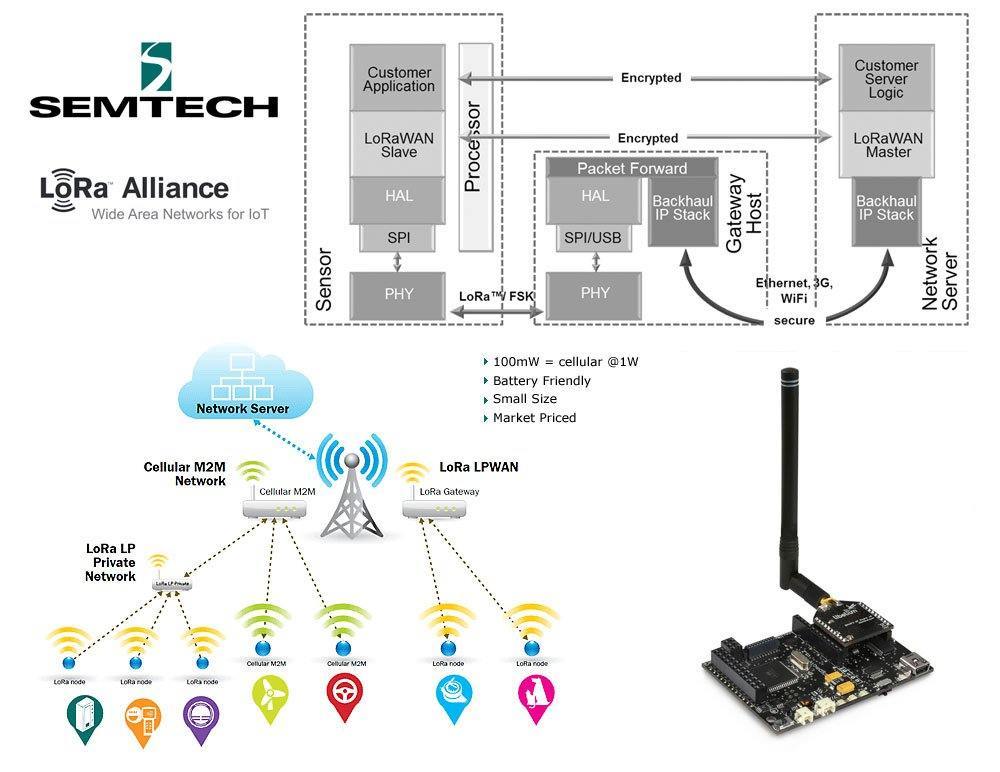
**Fig. 6 : Maximum MACPayload size length (M) for a given data rate [9]**



**Fig. 7 : Maximum MACPayload size without using repeaters [9]**

However, since we will send data through The Things Network, we will be limited by the “fair-access policy” value of 11 bytes for our non-encrypted payload. It is our limiting parameter.

Example of a LoRa transmission (from the node to the network server)



**Fig.8: Different layers of a node and a gateway in a LoRa transmission [10]**

Activation of an end-device (LoRaWAN 1.0.2 specification) [11]

The end nodes are the devices that will collect data (temperature, pressure, wind speed, activation of alarms, fill level of a tank, …) and will send it to application servers that will be accessed by the user. In some cases, the nodes are even simpler than what is described above and will not have any sensor on them (these nodes will just send some predetermined data at a certain period).

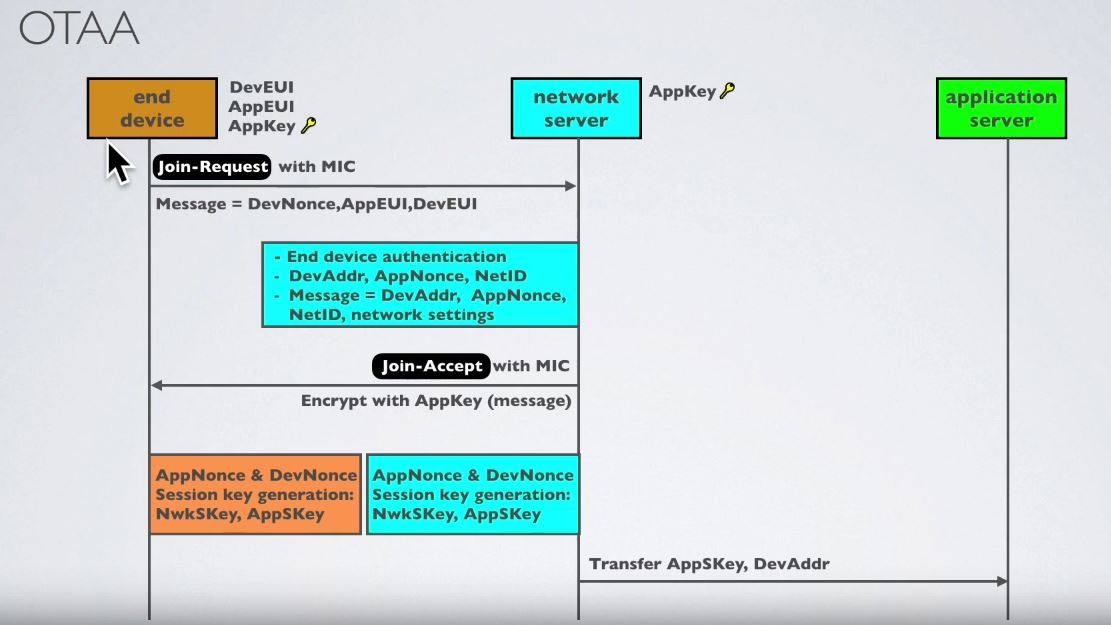
Before being able to communication with the IoT network (ex : TheThingsNetwork), the device must be activated. There are 2 methods: OTAA (Over The Air Activation) and ABP (Activation By Personalization). We will not discuss about the last one, as we will not use it for our project.

**The OTAA method** is a “join-request” and “join-accept” activation process, where the end-device sends the request to the network server, where it will respond by a “join-accept” if the device is properly authenticated. Before the activation, the node must store its DevEUI, AppEUI and AppKey. The network server stores the same AppKey as the node.

The DevEUI (64 bits – Extended Unique Identifier) uniquely identifies the end-device and is very similar to a MAC address. In most cases, end-devices come with a pre-loaded DevEUI.

The AppEUI uniquely identifies the application server we are going to send the data to. This parameter is like a port number.

The AppKey is an AES symmetric key (128 bits) and is used to generate a MIC (Message Integrity Code) to ensure the integrity of the exchanged messages.



**Fig.9: OTAA activation process (Copyright Robert Lie – mobilefish.com) [11]**

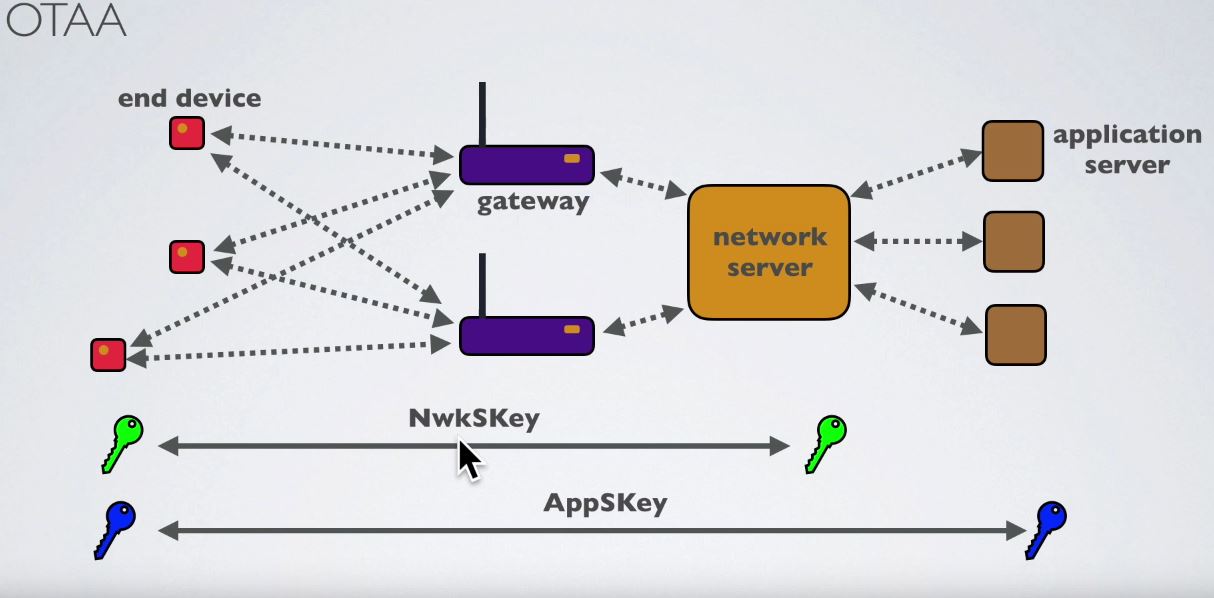
The first step of the activation process is the “join-request” message which contains the DevNonce (randomly generated number to ensure that no other device can repeat this request), AppEUI and DevEUI. The AppKey generates the MIC for the request message but the message itself is not encrypted with the AppKey. When the network server receives the request, it will check if the DevNonce has already been used or not (a message “failure” will be sent if it is the case). It will do the same for the AppEUI and DevEUI. Then, the network server will generate its own MIC with its own AppKey. If it is the same as the MIC sent by the device, the node is successfully authenticated.

The network server will then generate, for the end-device, a DevAddr. This is a shorter 32 bits address that is mapped to the unique DevEUI and that has a similar meaning to a client IP address. The purpose of this shorter address is to reduce the transmission overhead during the communications.

The AppNonce is a randomly generated number (like DevNonce) and the NetID (24 bits value to identify LoRaWAN networks) is the network identifier of the end-device.

The “join-accept” message sent to the node contains all of this information and some network settings (data rates to use for reception, channel frequency list, reception delay, …) and is encrypted with the AppKey. The “join-accept” message also contains its MIC.

At the end of this exchange, the AppNonce and the DevNonce (which are the same for both end-device and network server) will generate a network session key (NwkSKey) and an application session key (AppSKey). The network server will transfer the AppSKey and the DevAddr to the application server.



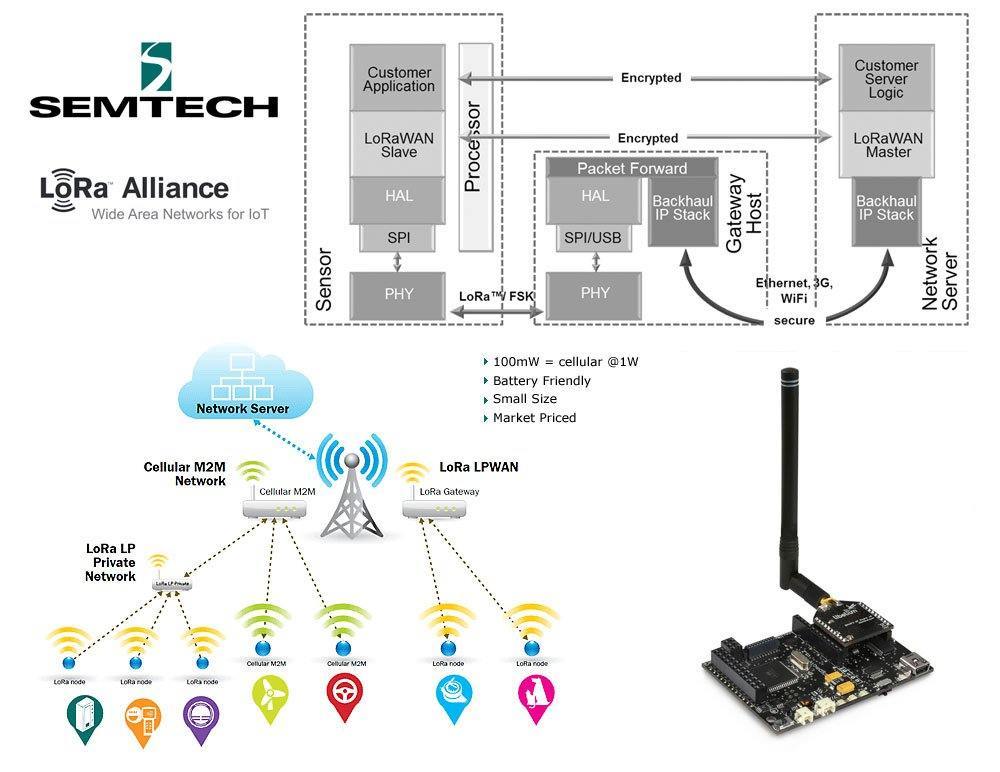
**Fig.10: OTAA message exchange (Copyright Robert Lie – mobilefish.com) [11]**

The NwkSKey is used between the node and the network server to ensure the MIC of all data messages (ensure data integrity). The AppSKey is used to encrypt/decrypt the payload between the end-device and the application server.

Sending data from the end-device to the end-user

After activation, the nodes are ready to send data the network. The first step before sending anything to the network is to encrypt the information with the AppSKey and ensure integrity of this message with the NwkSKey (generate a MIC). After the payload has been encrypted and the MIC added to the message, the node will broadcast its data to all available gateways in the vicinity. This broadcasting is done through LoRa RF and it makes sure that the data has arrived at least to one gateway. Remember that in LoRa, the power of a transmission is about a few dBm (which is relatively low compared to other technologies).

When the data has arrived at one (or more) gateway(s), it will be passed onto the network server. The gateway doesn’t know what is inside the data (the payload), because it is encrypted, and the gateways don’t have the keys to decrypt it. These devices only have a packet forwarder that, as the name implies, forward data to the next element of the chain.



**Fig.11: Close-up of the link between the node, the gateway and the network server [10]**

The only purpose of the gateway is to identify the network where the node belongs to (ex : TheThingsNetwork) by looking at the NetID [12]. The information is then sent from the gateway to the network server via WiFi, Ethernet, 4G, 3G, … One important thing to note is that the most recent versions of LoRaWAN allow roaming. This means that if my packet is sent to the wrong network (because the gateways that received my packet are registered to a different network than mine), the network server can redirect my packet to the right network.

As we have seen previously, several gateways can receive the same packet from my end-device. This is not a problem, as the network server will filter the received packets to only keep one.

When the network server finally receives the data, it will use the NwkSKey (which is the same as the node’s Session Key) to check the integrity of the received data. Then, the packet will be forwarded to the TTN (The Things Network) application server and decrypted with the AppSKey [13]. This means that TTN can (potentially) see the data that transits on the network … After that, by looking at the DevAddr (which is like the IP address of the node) in the LoRaMAC field, the TTN application server will send the packet to our Node-RED server (it is “our” application server) using the MQTT protocol (this protocol is explained in the next section). Finally, the user, with his/her browser, can access to the information contained on Node-RED and see the payload from the end-device (temperature, pressure, timestamp, alarm, fill-level, …)

MQTT broker [14] [15] [16]

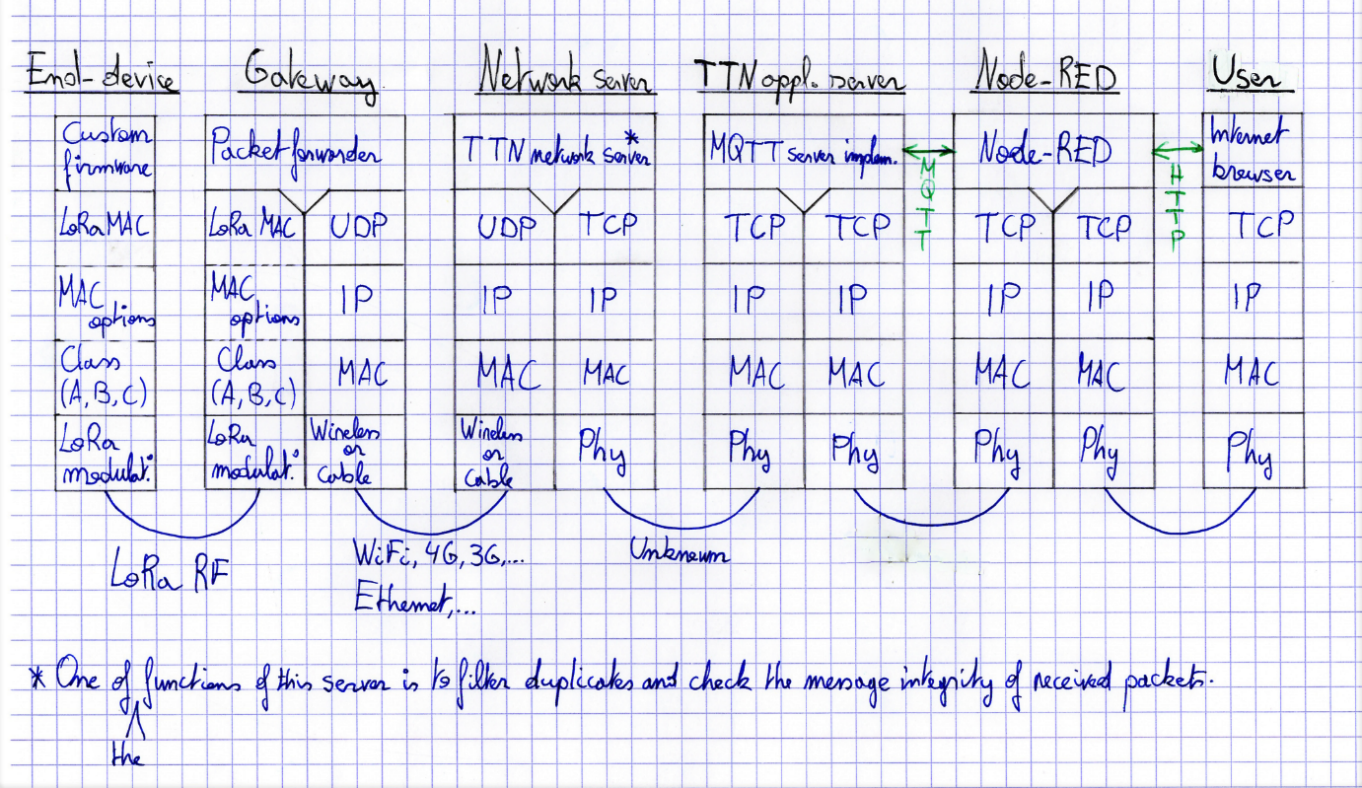
MQTT stands for “Message Queuing Telemetry Transport” and it is an application protocol. It is also known as a publish/subscribe protocol and it is designed to be lightweight and to minimise network bandwidth and resource requirements for the devices. It is well-suited for high-latency or unreliable networks (the last one is especially true for the LoRa link) and it attempts to ensure some reliability of packets delivery. MQTT is different from HTTP, where the last one is a request/response protocol used by the browser to navigate on the web pages and request content (text, image, video, …).

The MQTT broker is the centralised system that hands-over the messages to the subscribed clients (our application servers). This system is also able to store data temporarily in a buffer and push it to the subscribers. MQTT will only give the last received messages it has. It will not store them.

In an IoT architecture, the broker can be placed between the network server and the application server and simply relay the uplinks and downlinks between the nodes and the application servers. However, for this project, the MQTT broker is implemented inside the TTN (The Things Network) application server, but its function remains unchanged.

To receive messages on a certain topic, you need to subscribe to it (ex : IoT/home/node1 -> my application server will receive messages from node1 through the broker).

Summary of the transmission chain (LoRa/LoRaWAN side)



**Fig.12: Transmission chain between the node and the user (OSI model)**

This figure sums up the explanation that were given in the previous sections (except for the activation, where everything related to it was thoroughly explain there).

It is important to note that when the node sends a packet to the application server, it is called an uplink. When the application server sends a message to the node, it is called a downlink. Also, the MQTT broker is implemented in the TTN application server and communicates via MQTT protocol with Node-RED (“our” application server).

Bibliography

[1] <https://cdn.mikroe.com/blog/2016/02/lorawan.jpg>(viewed the 25.03.2019 at 17h00)

[2] Quora: What is the difference between Lora and LoraWan ? <https://www.quora.com/What-is-the-difference-between-Lora-and-LoraWan> (viewed the 25.03.2019 at 17h00)

[3] Proximus EnCo : What is the difference between LoRa and LoRaWAN ? <https://enablingsupport.zendesk.com/hc/en-us/articles/205089362-What-is-the-difference-between-LoRa-and-LoRaWAN-> (viewed the 25.03.2019 at 17h00)

[4] <https://zakelijkforum.kpn.com/lora-forum-16/what-is-lora-and-lorawan-8314> (viewed the 25.03.2019 at 17h00)

[5] Le protocole LoRaWAN (Wikipedia) : <https://fr.wikipedia.org/wiki/LoRaWAN#Le_protocole_LoRaWAN> (viewed the 25.03.2019 at 17h00)

[6] Modeling the Energy Performance of LoRaWAN (by Lluís Casals, Bernat Mir, Rafael Vidal and Carles Gomez) : <https://www.researchgate.net/publication/320435869_Modeling_the_energy_performance_of_LoRaWAN> (viewed the 25.03.2019 at 17h00)

[7] <https://docs.allthingstalk.com/tutorials/lora-payload/> (viewed the 25.03.2019 at 17h00)

[8] <https://www.thethingsnetwork.org/forum/t/spreadsheet-for-lora-airtime-calculation/1190/4> (Viewed the 25.03.2019 at 17h00)

[9] LoRaWAN specification V1.0 (by LoRa Alliance) (pages 35-36-37) : <https://www.rs-online.com/designspark/rel-assets/ds-assets/uploads/knowledge-items/application-notes-for-the-internet-of-things/LoRaWAN%20Specification%201R0.pdf> (Viewed the 23.03.2019 at 20h45)

[10] <https://www.postscapes.com/wp-content/uploads/2018/03/lora.jpg> (viewed the 27.03.2019 at 16h30)

[11] LoRa/LoRaWAN tutorial 21: OTAA, ABP and LoRaWAN Security : <https://www.youtube.com/watch?v=KrNDOBzhxeM> (viewed the 27.03.2019 at 16h50)

[12] NetID FAQ : <https://lora-alliance.org/sites/default/files/2018-11/20181114_NetID_FAQ.pdf> (viewed the 27.03.2019 at 16h50)

[13] LoRaWAN security: <https://www.thethingsnetwork.org/docs/lorawan/security.html> (viewed the 21.03.2019)

[14] MQTT FAQ : <http://mqtt.org/faq> (viewed the 27.03.2019 at 16h50)

[15] M2M, IoT, LoRa, MQTT, … : comment profiter simplement des objets connectés ? : <https://www.atys-concept.com/blog-de-la-performance/articles-cybersecurite-et-reseaux/m2m-iot-lora-mqtt-comment-profiter-simplement-des-objets-connectes/> (viewed the 27.03.2019 at 16h50)

[16] MQTT Publish and Subscribe Beginners Guide: <http://www.steves-internet-guide.com/mqtt-publish-subscribe/> (viewed the 27.03.2019 at 16h50)