IoT Networking, part 2:

LoRa, LoRaWan, TheThingsNetwork & MQTT

Agenda

- LoRa, LoRaWan, TheThingsNetwork
- MQTT

- [Networking exercises with

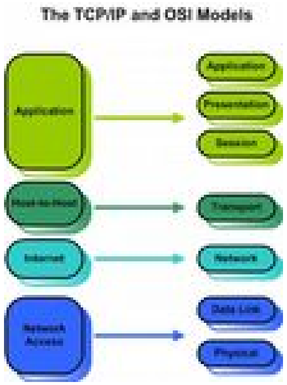
  LoRa, Bluetooth, WiFi

  MQTT, LoRaWan]

LoRa / 1

LoRa is a **proprietary Layer 1 standard** owned by Semtech

Chirp Spread Spectrum (CSS) with forward error coding and interleaving).



Bandwidth 125/250/500 kHz

Frequency in Europe: **ISM 868 MHz**

Data rate 11 kbps

Focus is on **long range, power efficiency, robustness.**

Modulation details:

http://www.semtech.com/images/datasheet/an1200.22.pdf

https://www.lora-alliance.org/portals/0/documents/whitepapers/LoRaWAN101.pdf

https://revspace.nl/DecodingLora#Modulation_basics

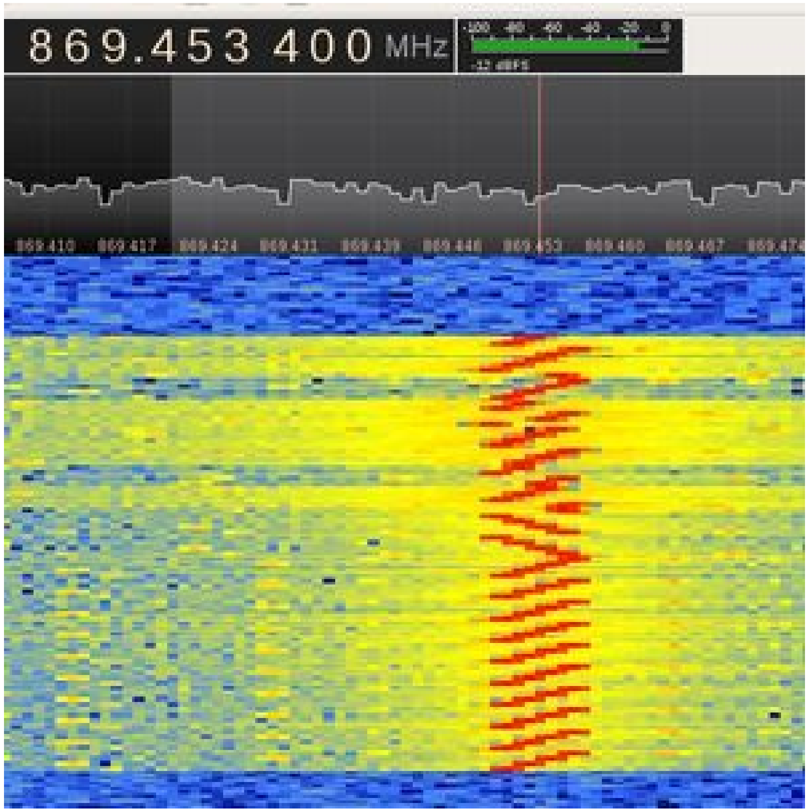
https://myriadrf.org/blog/lora-modem-limesdr/

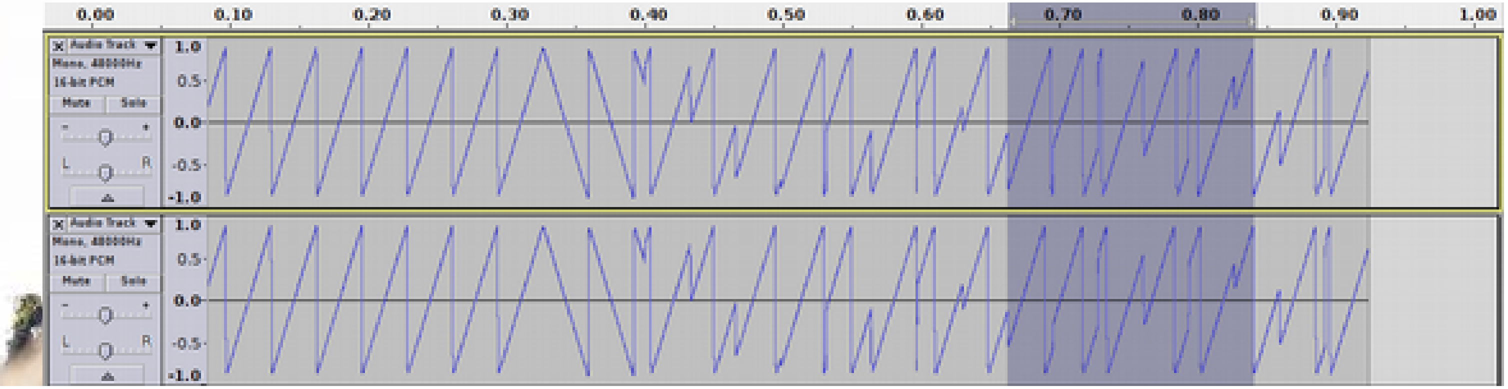
https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf

LoRa / 2

What is a chirp?

Source: https://revspace.nl/DecodingLora





LoRa / 3 / terms

Spreading Factor SF

$$\mathrm{SF} = \frac{\text{chip rate}}{\text{symbol rate}}$$

(think of it as “one bit is spread out over so and so many pulses”)

Control rate CR, determines depth of forward error coding

(Think of it as saying CCCAAAFFFEEE or CAFECAFECAFE instead of CAFE)

LoRa / 4 / terms

Interleaving

“Mixing up the letters to gain robustness against *burst errors*”

```
Transmitted sentence:             ThisIsAnExampleOfInterleaving...
Error-free transmission:          TIEpfeaghsxlIrv.iAaenli.snmOten.
Received sentence, burst error:   TIEpfe______Irv.iAaenli.snmOten.
after deinterleaving:             T_isI_AnE_amp_eOfInterle_vin_...
```

LoRa / 5

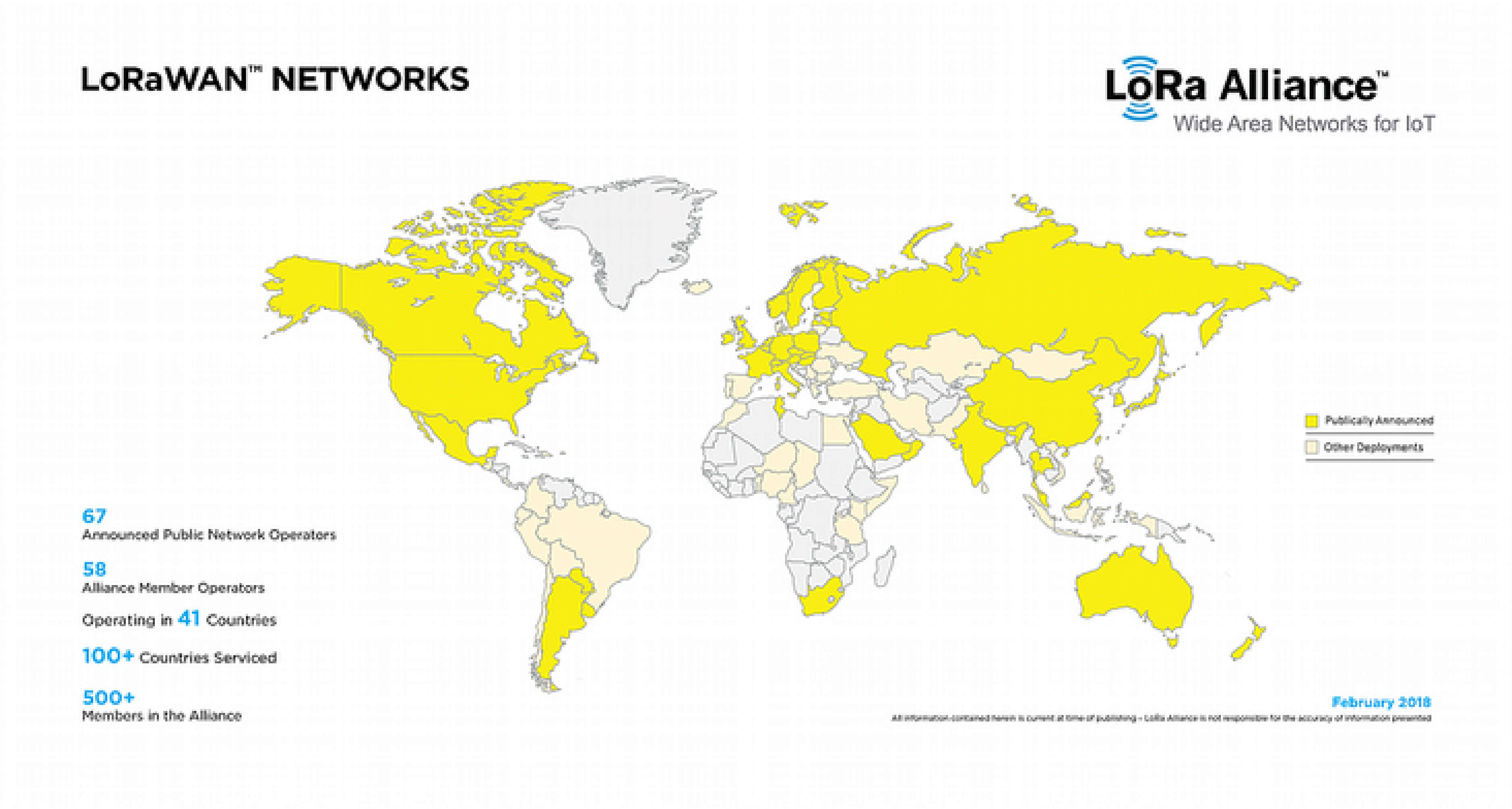
Business aspects:
LoRa / LoRaWan - one of many contenders for the IOT throne.
Others include NB-IoT, Sigfox, ...

SIGFOX
One network A billion dreams
nwave
LoRa™
LTE-M
NB-LTE
INGENU
WEIGHTLESS
IEEE 802.11ah
DASH7™ ALLIANCE
EC-GSM
ZigBee 3.0
Bluetooth® 4.0

LoRaWan / 1

- LoRaWan is an open standard building on top of LoRa
- https://www.lora-alliance.org/

LoRaWan / 2



LoRaWAN™ NETWORKS

LoRa Alliance™
Wide Area Networks for IoT

🟨 Publically Announced

□ Other Deployments

67
Announced Public Network Operators

58
Alliance Member Operators

Operating in **41** Countries

100+ Countries Serviced

500+
Members in the Alliance

February 2018

All information contained herein is current at time of publishing – LoRa Alliance is not responsible for the accuracy of information presented

LoRaWan / 3

LoRaWAN™ is a Low Power Wide Area Network (LPWAN) specification intended for wireless battery operated Things in a regional, national or global network. LoRaWAN targets key requirements of Internet of Things such as secure bi-directional communication, mobility and localization services. The LoRaWAN specification provides seamless interoperability among smart Things without the need of complex local installations and gives back the freedom to the user, developer, businesses enabling the roll out of Internet of Things.

Details:

https://www.lora-alliance.org/What-Is-LoRa/Technology

LoRaWan / 4

- **Star-of-stars topology**
- **Gateways** are transparent bridges relaying messages between **end-devices** and a central **network server** in the backend.
- **Gateways are connected** to the network server via **standard IP connections** while end-devices use single-hop wireless communication to one or many gateways.
- All end-point communication generally **bi-directional**, supports **multicast** enabling **software upgrade over the air** or other mass distribution messages

Details:

https://www.lora-alliance.org/What-Is-LoRa/Technology

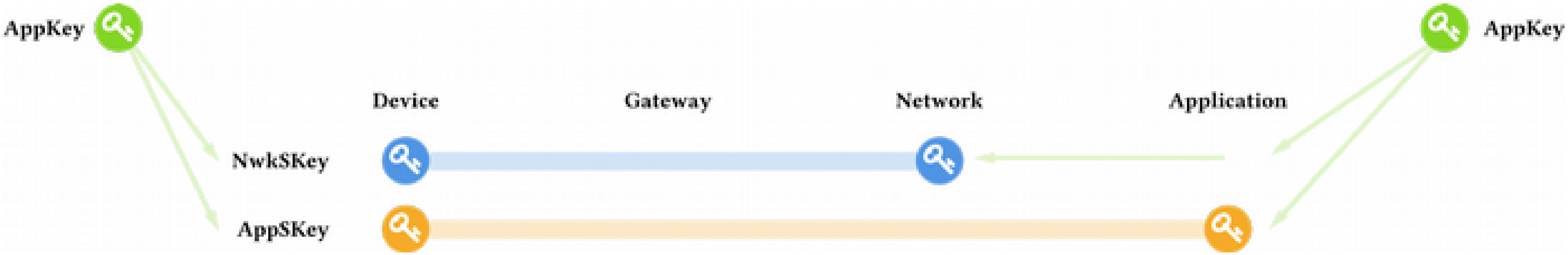
LoRaWan / 5

Security measures:

Unique Network key (EUI64), network level

Unique Application key (EUI64), end to end security on application level

Device specific key (EUI128),
Device 32 bit id, unique in network, ideally globally unique



Source, Details:

https://www.lora-alliance.org/What-Is-LoRa/Technology

LoRaWan / 6

Device classes

A Battery powered, small loads, long breaks, long latency, unicast

B low latency, scheduled receive slots, periodic beacon from gateway, uni/multicast, higher power, 14-30 mA

C no latency, uni/multi, constantly receiving, power hungry

Classes can be dynamically assigned / changed

Source, Details:

https://www.lora-alliance.org/What-Is-LoRa/Technology

The Things Network / 1

The Things Network / 2 / Manifesto

Everything that carries power will be connected to Internet eventually.

Controlling the network that makes this possible means controlling the world. We believe that this power should not be restricted to a few people, companies or nations. Instead this should be distributed over as many people as possible without the possibility to be taken away by anyone. We therefore founded "The Things Network".

The Things Network is an open source, free initiative with the following properties:

It connects sensors and actuators, called "Things", with transceivers called "Things Gateways" to servers called "Things Access".
The first connection is "Over The Air", the second is "Over The Net". The distributed implementation of these concepts is called "The Things Network".
Anyone shall be free to set up "Things" and connect to "Things Gateways" that may or may not be their own.
Anyone shall be free to set up "Things Gateways" and connect to "Things Access" that may or may not be their own. Their "Things Gateways" will give access to all "Things" in a net neutral manner, limited by the maximum available capacity alone.
Anyone shall be free to set up "Things Access" and allow anonymous connections from the Internet. Their "Things Access" will give access to all "Things Gateways" in a net neutral manner, limited by the maximum available capacity alone. Furthermore their "Things Access" will allow connection of other "Things Access" servers for the distribution of data.
The "Over The Air" and "Over The Net" networks shall be protocol agnostic, as long as these protocols are not proprietary, open source and free of rights.
Anyone who perpetrates a "Things Access" or a "Things Gateway" will do so free of charge for all connecting devices and servers.
Anyone making use of the network is allowed to do so for any reason or cause, possibly limited by local law, fully at own risk and realizing that services are provided "as is" and may be terminated for any reason at any moment. The use may be open for anybody, limited to customers, commercial, not-for-profit, or in any other fashion. "The Things Network" providers will not pose restrictions upon its users.

We invite you to sign this Manifesto, and uphold its principles to the best of your abilities.

Source, Details:

https://github.com/TheThingsNetwork/Manifest

The Things Network / 3 / Essentials

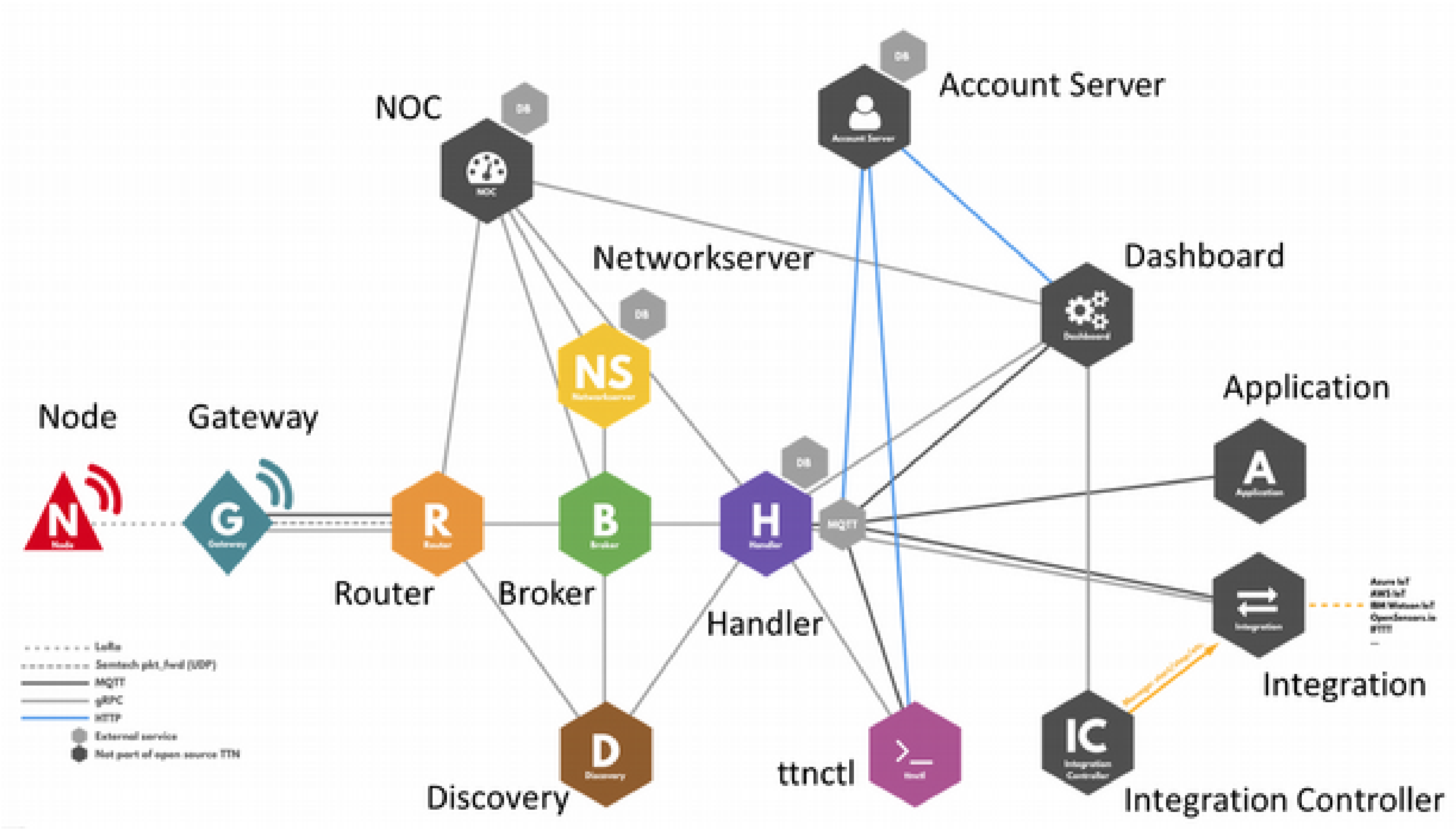
- Open source
- Free ... to set up and run their own, in particular:

  Anyone who perpetrates a "Things Access" or a "Things Gateway" will do so **free of charge for all connecting devices and servers.**

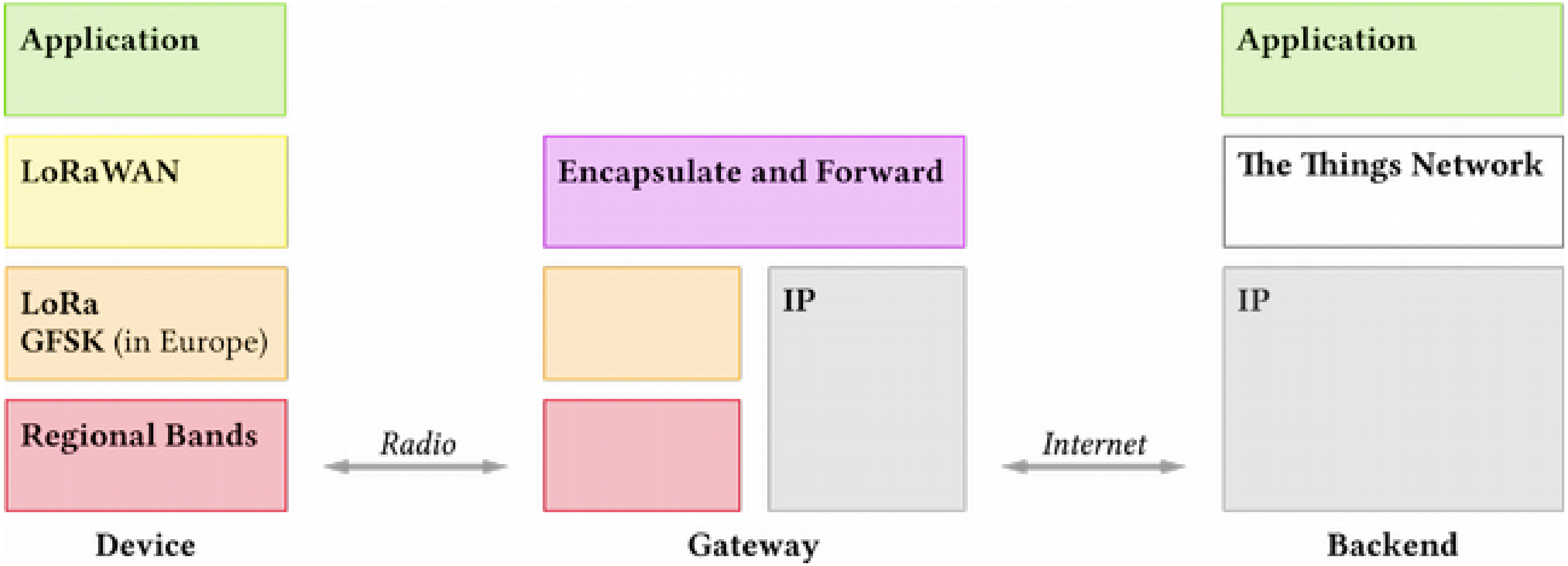
  This to some degree explains our current interest in TTN, in an educational context.

Source, Details:
https://github.com/TheThingsNetwork/Manifest

The Things Network / 4 / Architecture



The Things Network / 5 / Architecture



GFSK = Gaussian Frequency Shift Keying - https://en.wikipedia.org/wiki/Frequency-shift_keying

The Things Network / 6 / Security

= LoRaWan defined

NwkSKey, AppSKey and AppKey. All keys have a length of 128 bits.

NwkSKey for interaction between Node and the Network. Also used to map a non-unique device address (DevAddr) to a unique DevEUI and AppEUI.

AppSKey is used for encryption and decryption of the payload.

NwkSKey and AppSKey are unique per device, per session. If you dynamically activate your device (OTAA), these keys are re-generated on every activation.

The Things Network / 7 / Security, cntd

Dynamically activated devices (OTAA) use the application key (AppKey) to derive the two session keys during the activation procedure. In The Things Network you can have a default AppKey which will be used to activate all devices, or customize the AppKey per device.

What you will use, in your code:

DevEUI, AppEUI, AppKey

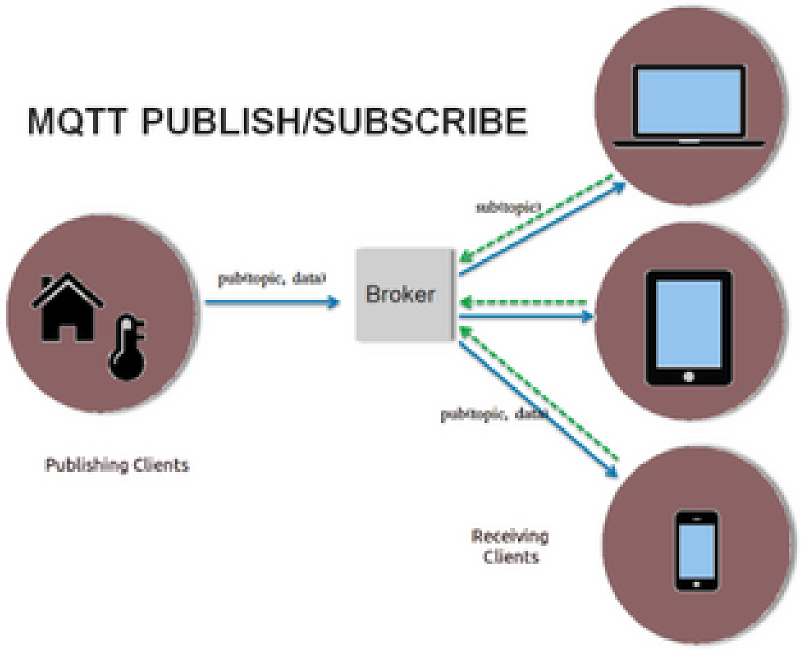
Keys will be generated on TTN server, on registration.

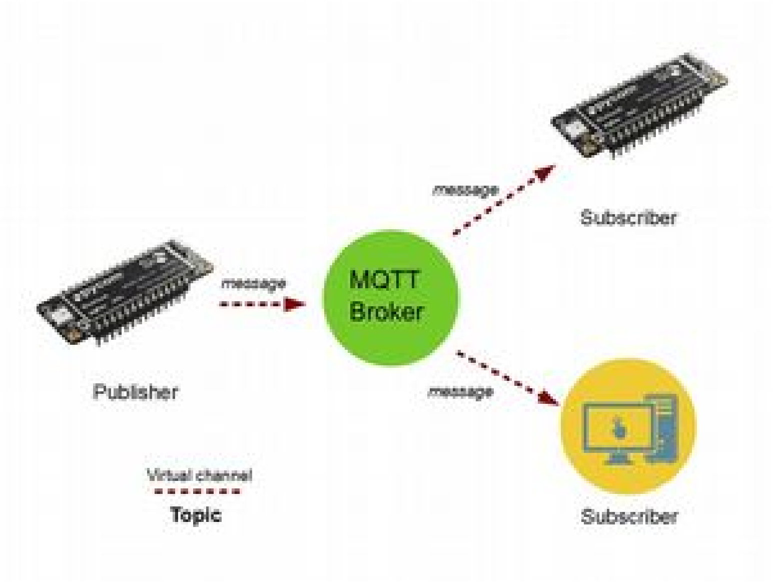
Source, Details:

https://www.thethingsnetwork.org/wiki/LoRaWAN/Security

MQTT / 1

MQTT (Message Queuing Telemetry Transport) is a **publish-subscribe**-based messaging protocol.
Works on top of **TCP/IP**.
A Message broker or server receives and redistributes messages.





Source: https://wso2.com/library/articles/2016/06/article-the-basics-of-mqtt-and-how-wso2-products-support-mqtt-protocol/
https://www.hackster.io/bucknalla/mqtt-micropython-044e77

MQTT / 2 / topics

MQTT publishing and subscription is organized by topics:

e.g.
/house/light
Or
/greenhouse/temperature

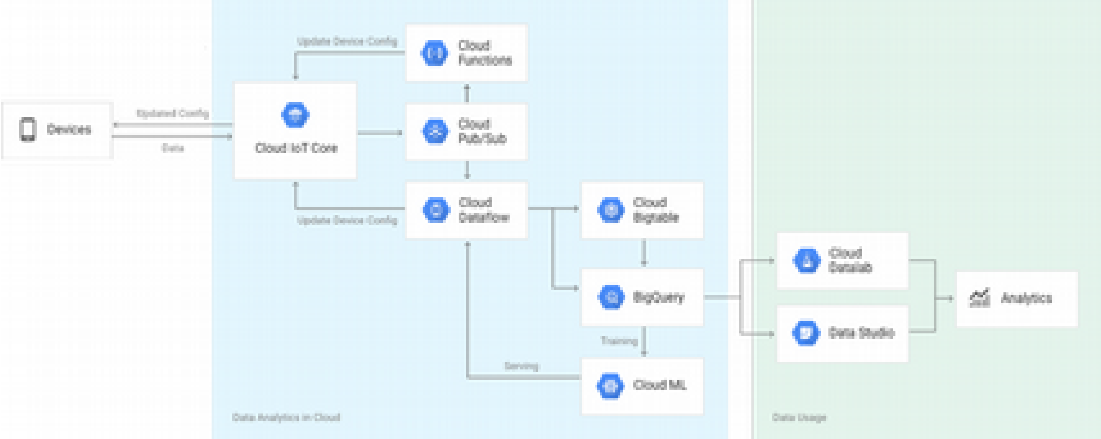
(case sensitive!)

Messages are in **free format,** however, often you will see e.g. json or xml messages.
Some service might restrict message formats.

MQTT / 3 / adoption

MQTT is a widely accepted and deployed standard in services/platforms such as

- Amazon AWS IoT
- Microsoft Azure
- Facebook Messenger (to an unknown degree)
- Google Cloud IoT Core is able to support data “from millions of globally dispersed devices.” Like similar services, Cloud IoT Core supports the standard MQTT and HTTP protocols for talking to devices.



Source: https://techcrunch.com/2018/02/21/googles-cloud-iot-core-is-now-generally-available

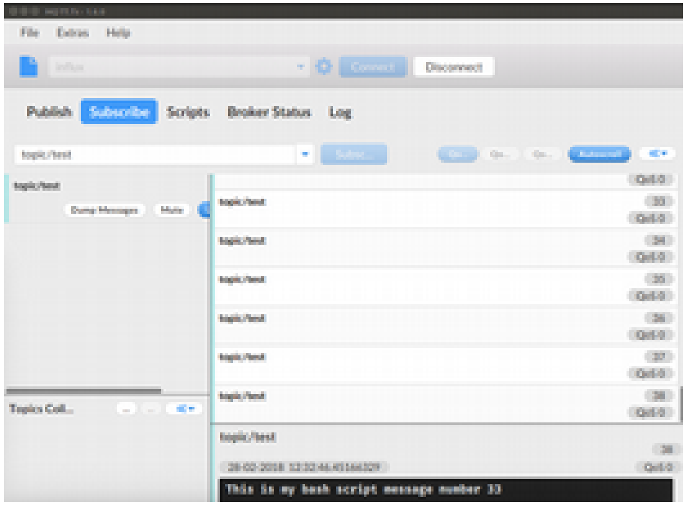
MQTT / 4 / usage

MQTT clients exist for a wide variety of platforms and languages:
https://github.com/mqtt/mqtt.github.io/wiki/software?id=software
https://www.hivemq.com/blog/seven-best-mqtt-client-tools

e.g
MQTT.fx
(available for Win/MacOSX/Linux, http://www.jensd.de/apps/mqttfx/, free)
mqtt-spy
(based on Java 8, http://kamilfb.github.io/mqtt-spy/ , OpenSource)
mosquitto_tools
(commandline, for Win/MacOSX/Linux, https://mosquitto.org/download/ , OpenSource)

And of course for

Arduino (C)
Raspberry (== Debian Linux)
Pycom (micropython)



```
topic/test                                   QoS 0
topic/test                                      33
                                             QoS 0
topic/test                                      34
                                             QoS 0
topic/test                                      35
                                             QoS 0
topic/test                                      36
                                             QoS 0
topic/test                                      37
                                             QoS 0
topic/test                                      38
                                             QoS 0
topic/test
                                                38
28-02-2018 12:32:46.45166329                 QoS 0
This is my bash script message number 33
```

MQTT / 5 / servers (aka brokers)

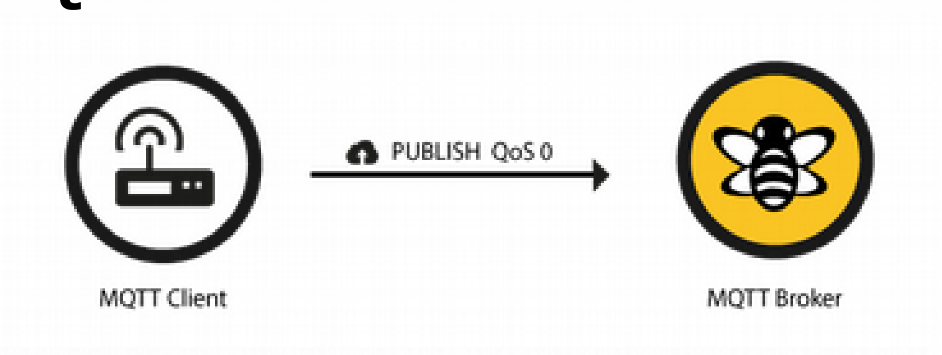
A wide choice of servers is available, many of them open source:
https://github.com/mqtt/mqtt.github.io/wiki/servers

influx.itu.dk runs mosquitto server.

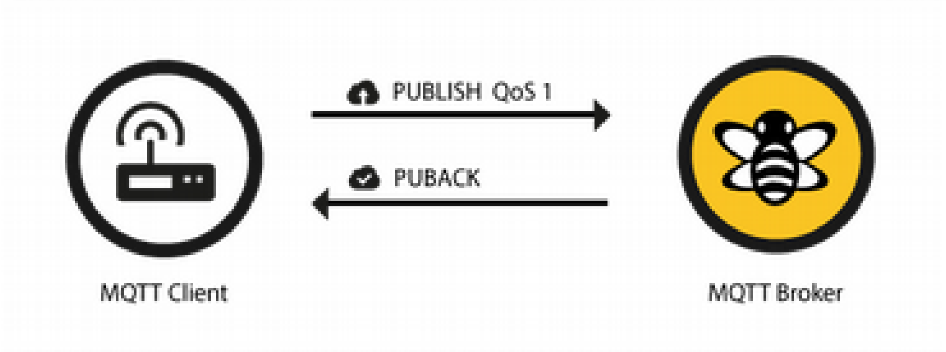
MQTT / 6 / QoS

MQTT implements 3 levels of QoS:

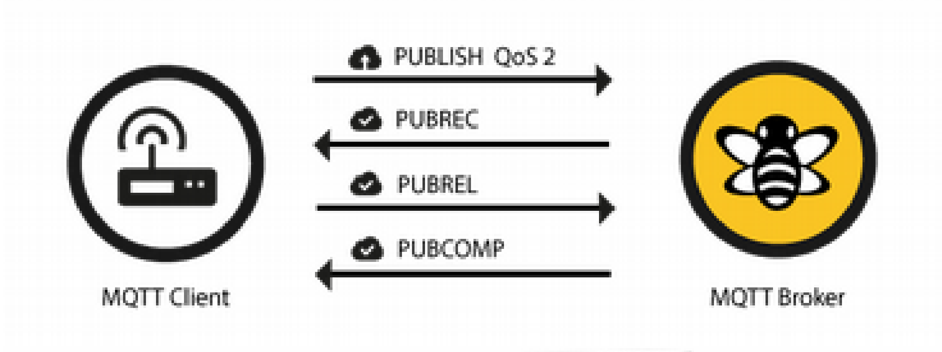
At most once (0)
Just send it



At least once (1)
Send and confirm



Exactly once (2)
Send, confirm, reference, stop.



MQTT / 7 / Last Will

MQTT Last Will and Testament

The Last Will and Testament (LWT) feature is used in MQTT to notify other clients about an ungracefully disconnected client. Each client can specify its last will message (a normal MQTT message with topic, retained flag, QoS and payload) when connecting to a broker. The broker will store the message until it detects that the client has disconnected ungracefully, and in tat case, send it out to subscribers.

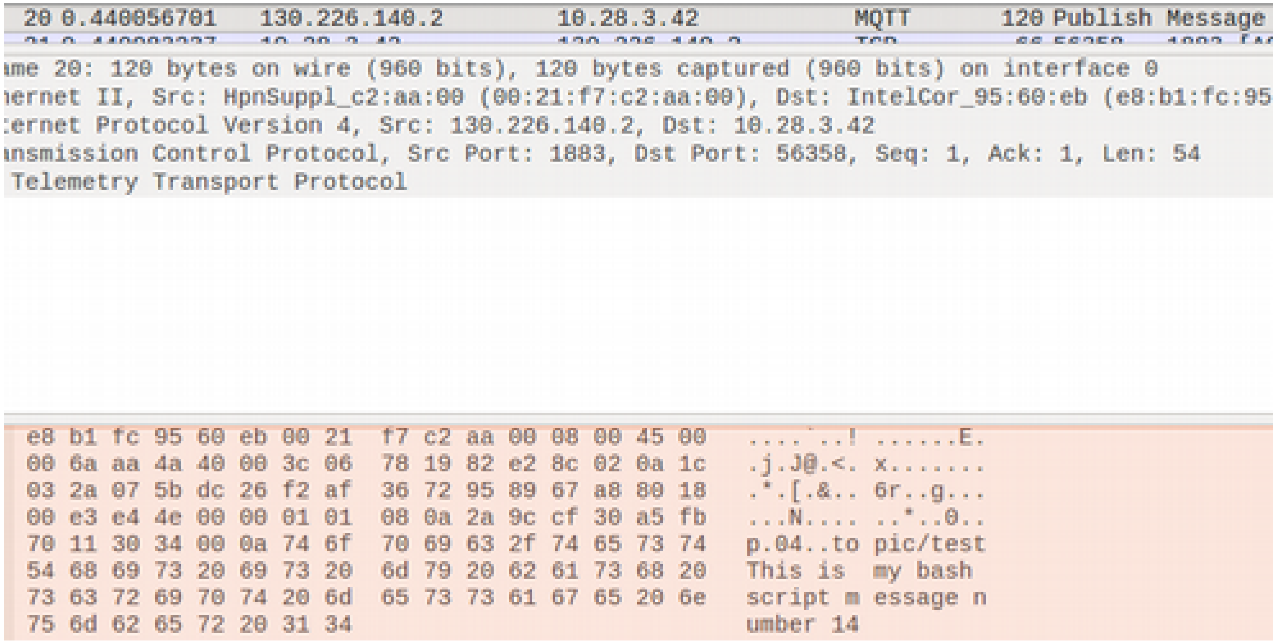
Why is this important?

In a lean communications protocol, dependent devices and services need to know about and have some chance to react to the disappearance of devices.

MQTT / 8 / security

MQTT may be used encrypted or unencrypted - here is mosquittos standard ports:

1883 : MQTT, unencrypted
8883 : MQTT, encrypted
8884 : MQTT, encrypted, client certificate required
8080 : MQTT over WebSockets, unencrypted
8081 : MQTT over WebSockets, encrypted



```
 20 0.440056701   130.226.140.2          10.28.3.42           MQTT        120 Publish Message
ame 20: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface 0
hernet II, Src: HpnSuppl_c2:aa:00 (00:21:f7:c2:aa:00), Dst: IntelCor_95:60:eb (e8:b1:fc:95
ternet Protocol Version 4, Src: 130.226.140.2, Dst: 10.28.3.42
ansmission Control Protocol, Src Port: 1883, Dst Port: 56358, Seq: 1, Ack: 1, Len: 54
 Telemetry Transport Protocol
```

```
e8 b1 fc 95 60 eb 00 21  f7 c2 aa 00 08 00 45 00   ....`..! ......E.
00 6a aa 4a 40 00 3c 06  78 19 82 e2 8c 02 0a 1c   .j.J@.<. x.......
03 2a 07 5b dc 26 f2 af  36 72 95 89 67 a8 80 18   .*.[.&.. 6r..g...
00 e3 e4 4e 00 00 01 01  08 0a 2a 9c cf 30 a5 fb   ...N.... ..*..0..
70 11 30 34 00 0a 74 6f  70 69 63 2f 74 65 73 74   p.04..to pic/test
54 68 69 73 20 69 73 20  6d 79 20 62 61 73 68 20   This is  my bash
73 63 72 69 70 74 20 6d  65 73 73 61 67 65 20 6e   script m essage n
75 6d 62 65 72 20 31 34                            umber 14
```

MQTT / 9 / security

MQTT with TLS/SSL works very much the same as https (which we are familiar with from web usage).
Example with letsencrypt certificates:

```
/etc/mosquitto/conf.d/ssl.conf
```

```
listener 8883
certfile /etc/letsencrypt/live/influx.itu.dk/cert.pem
cafile /etc/letsencrypt/live/influx.itu.dk/chain.pem
keyfile /etc/letsencrypt/live/influx.itu.dk/privkey.pem
```

A mosquitto_pub client would publish like this:

```
mosquitto_pub -h influx.itu.dk -p 8883 --capath
/etc/ssl/certs/ -t topic/test -m "encrypted msg"
```

MQTT / 10 / security

MQTT with TLS/SSL works very much the same as https.

In addition to “web style” TLS/SSL, specific client certificates can be demanded.

Username / password protection is also available.

MQTT / 11 / security note

Note:

If devices publish encrypted data,
but the broker allows subscribers to listen unencrypted,
data will be readable on the network!

Take-Aways, Networking 2

- Be able to name protocols and standards on

PHY, MAC and higher layers -
Know which protocol belongs where

- Be able to describe the main features of

LoRa
LoRaWan

MQTT

- Be able to explain security measures for these

Exercises

1/ **MQTT**

Find a MQTT client for your laptop OS,
install and configure to send messages to

influx.itu.dk:1883 topic/test

subscribe to topics, create new topics and share

2/ Wireshark monitoring

download and install wireshark.
use wireshark to monitor your communication.
Compare the encrypted and unencrypted communications.

3/ LoPy & MQTT

https://docs.pycom.io/chapter/tutorials/all/mqtt.html

(This guide is for adafruit broker -
how would you send to influx.itu.dk instead?)

4/ LoRaWan and TheThingsNetwork

See
https://github.com/ITU-PITLab/public/blob/master/Exercises_Networking2.2-LoPy-TTN.txt

For detailed instructions