Laboratory of Electronics
Antennas and Telecommunications

LoRaWAN Mac Layer

Ver. 1.1 - 2019

Fabien Ferrero
Université Côte d'Azur

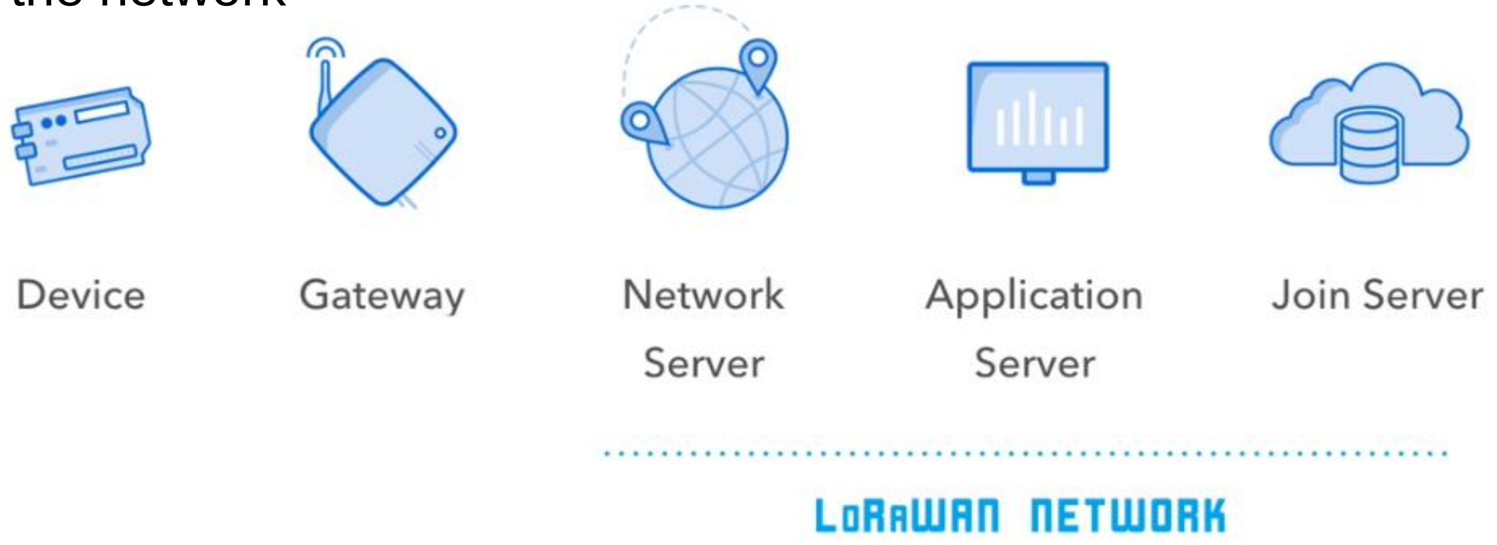
Outline

LoRaWan

- **LoRaWan** is a software protocol using lora physical layer

- **LoRaWan** include :
  - Session Layer
  - Network Layer
  - Data Link Layer

- It defines the packet format, and the way the packet are processed by the network

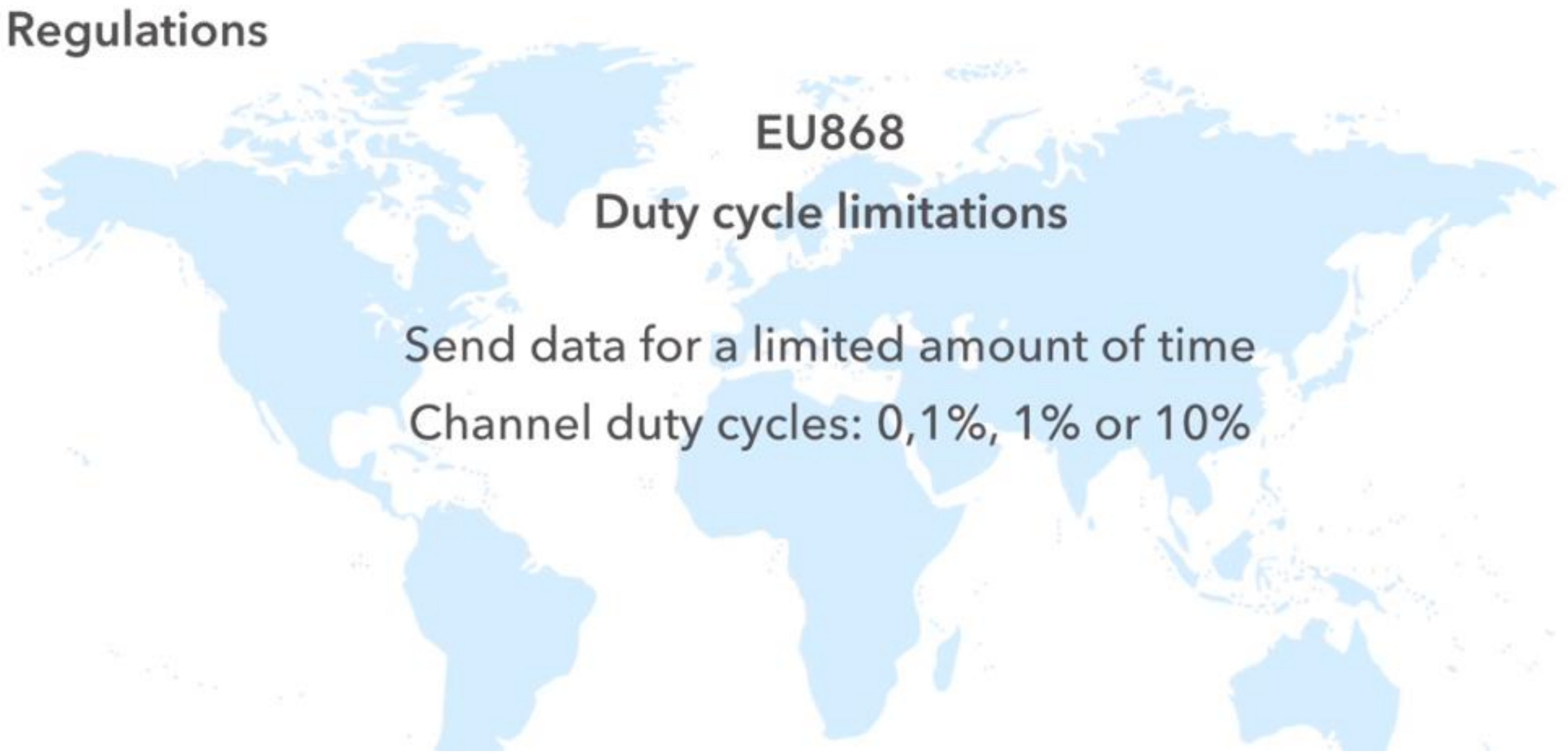


LoRaWan

- **LoRaWan is defined by the Lora alliance**

www.lora-alliance.org

LoRaWan bands



Regulations

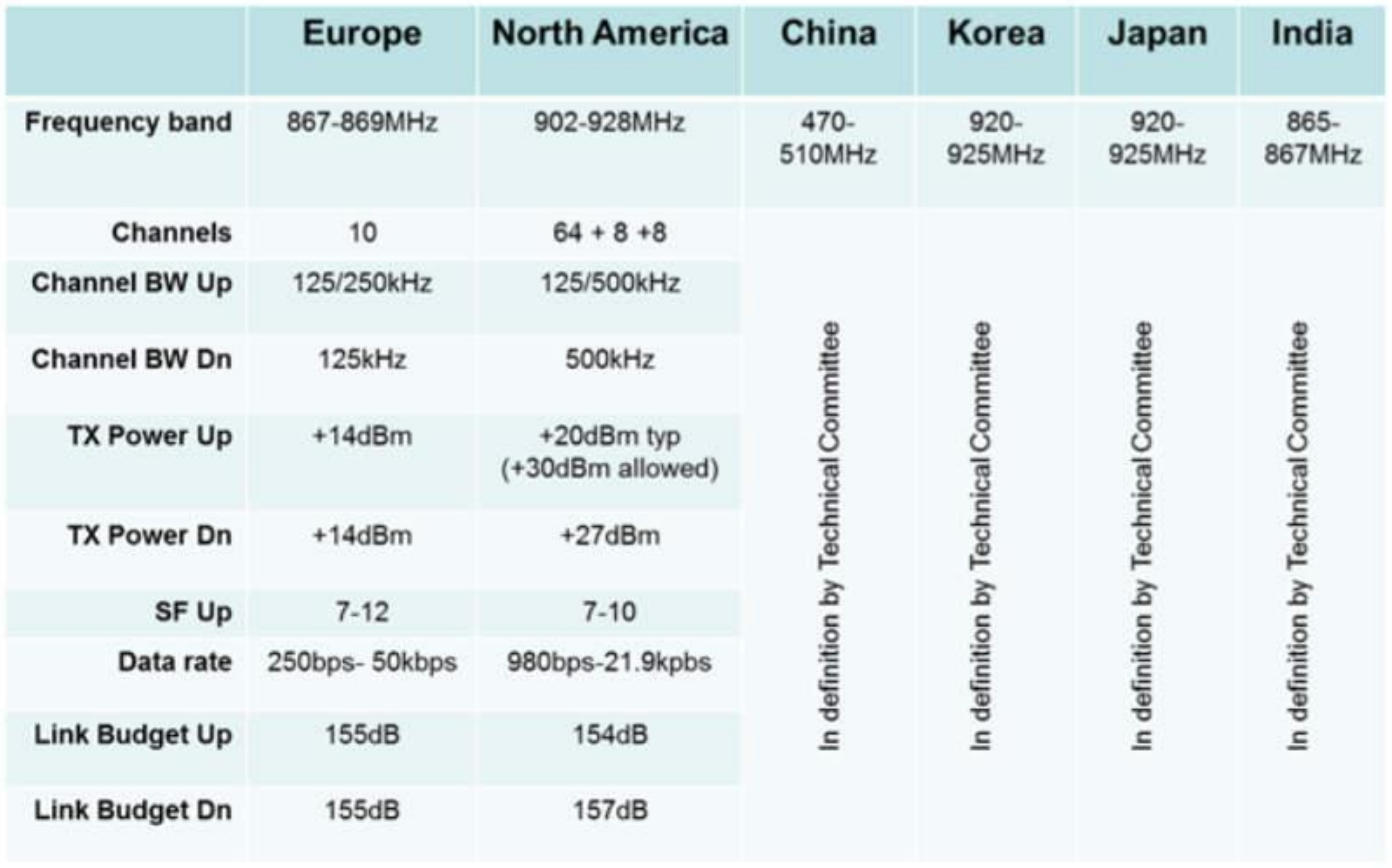
EU868

Duty cycle limitations

Send data for a limited amount of time

Channel duty cycles: 0,1%, 1% or 10%

LoRaWan bands

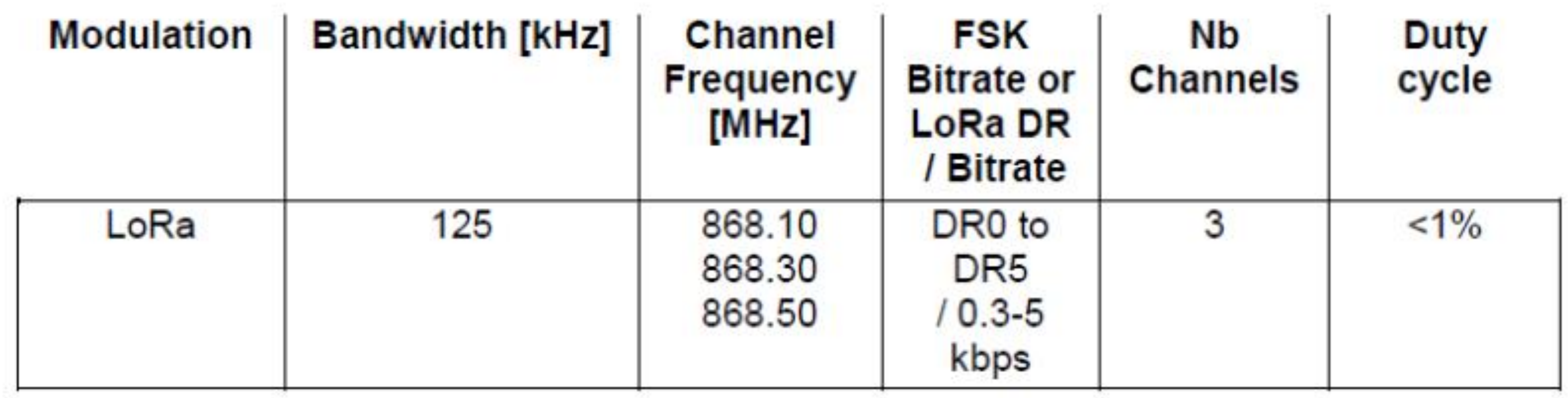


	Europe	North America	China	Korea	Japan	India
Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
Channels	10	64 + 8 +8	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee
Channel BW Up	125/250kHz	125/500kHz				
Channel BW Dn	125kHz	500kHz				
TX Power Up	+14dBm	+20dBm typ (+30dBm allowed)				
TX Power Dn	+14dBm	+27dBm				
SF Up	7-12	7-10				
Data rate	250bps- 50kbps	980bps-21.9kpbs				
Link Budget Up	155dB	154dB				
Link Budget Dn	155dB	157dB				

LoRaWan bands

License free Sub-GHz Frequencies :

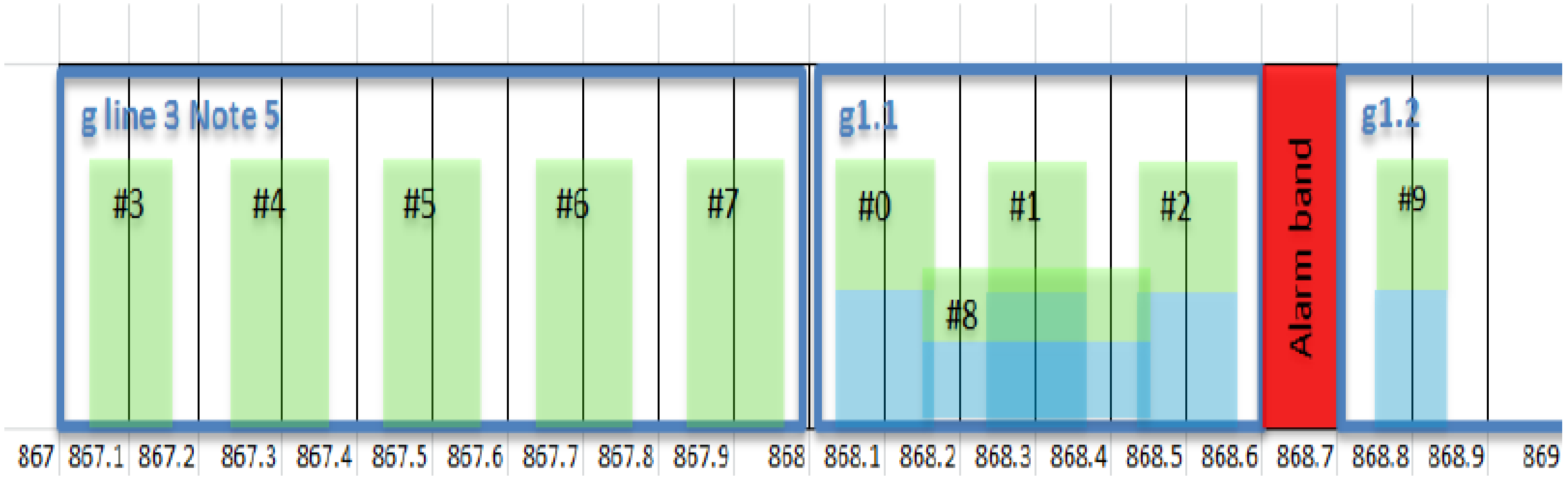
- Europe 868 MHz Band (863 ~ 870 MHz)
- Network channels can be freely attributed by the network operator
- Three mandatory channels that all gateways should constantly receive, and used by the end-points during the Join procedure :



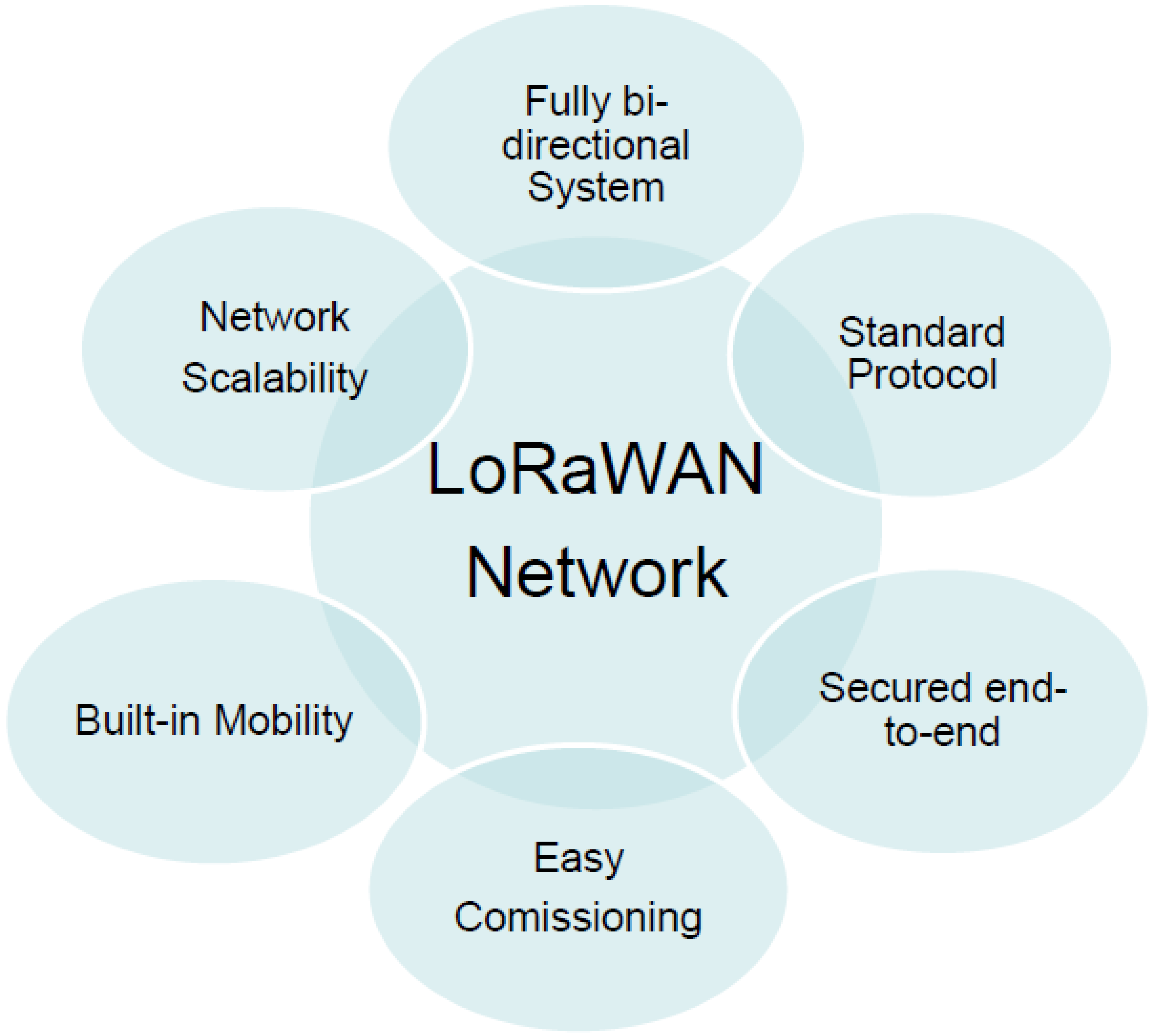
Modulation	Bandwidth [kHz]	Channel Frequency [MHz]	FSK Bitrate or LoRa DR / Bitrate	Nb Channels	Duty cycle
LoRa	125	868.10 868.30 868.50	DR0 to DR5 / 0.3-5 kbps	3	<1%

LoRaWan bands

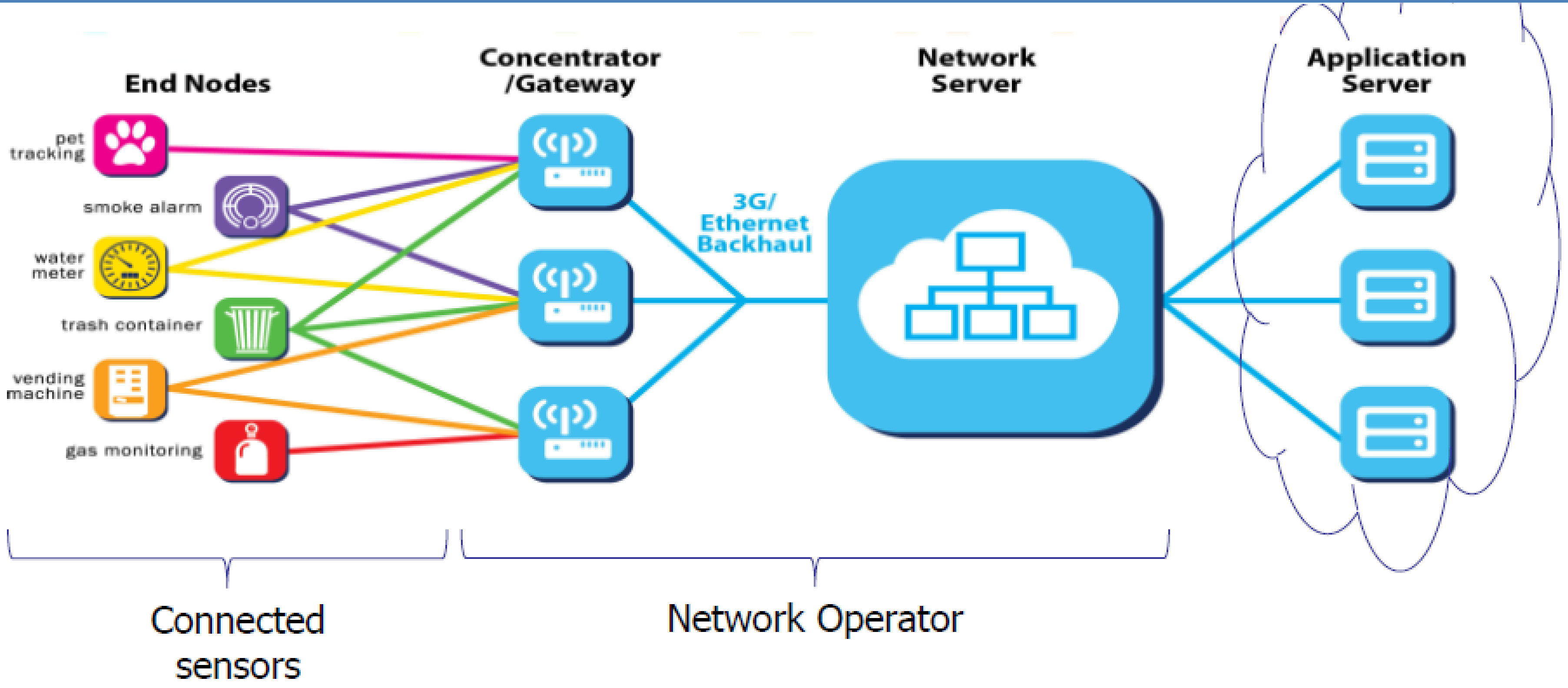
- 8 channels with multi-datarate 240 bps to 5.5 kbps
- 1 high-speed LoRa channel 11 kbps
- 1 high speed GFSK channel 50 kbps



LoRaWan


Fully bi-directional System
Network Scalability
Standard Protocol
LoRaWAN Network
Built-in Mobility
Secured end-to-end
Easy Comissioning

LoRaWan

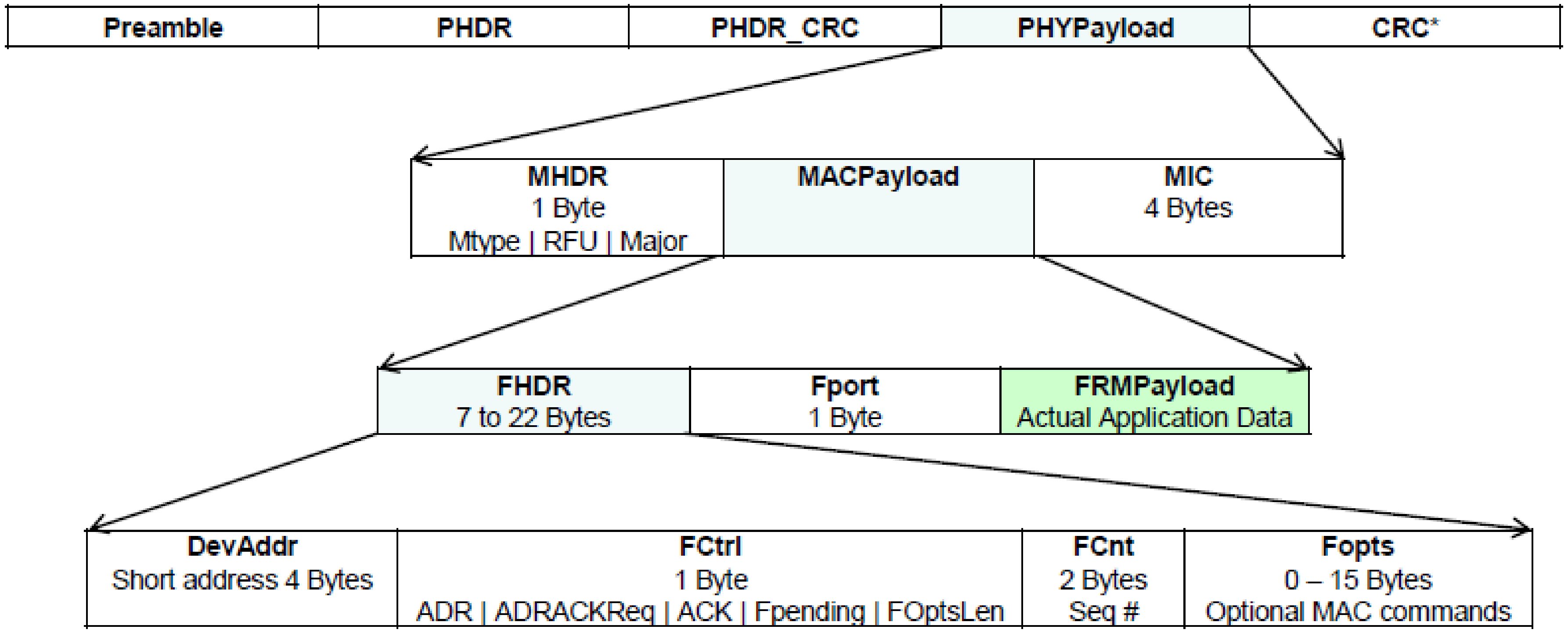


Centralized intelligence, cloud based

- No notion of connection or pairing from GW to device
- Mostly uplink, Aloha type system

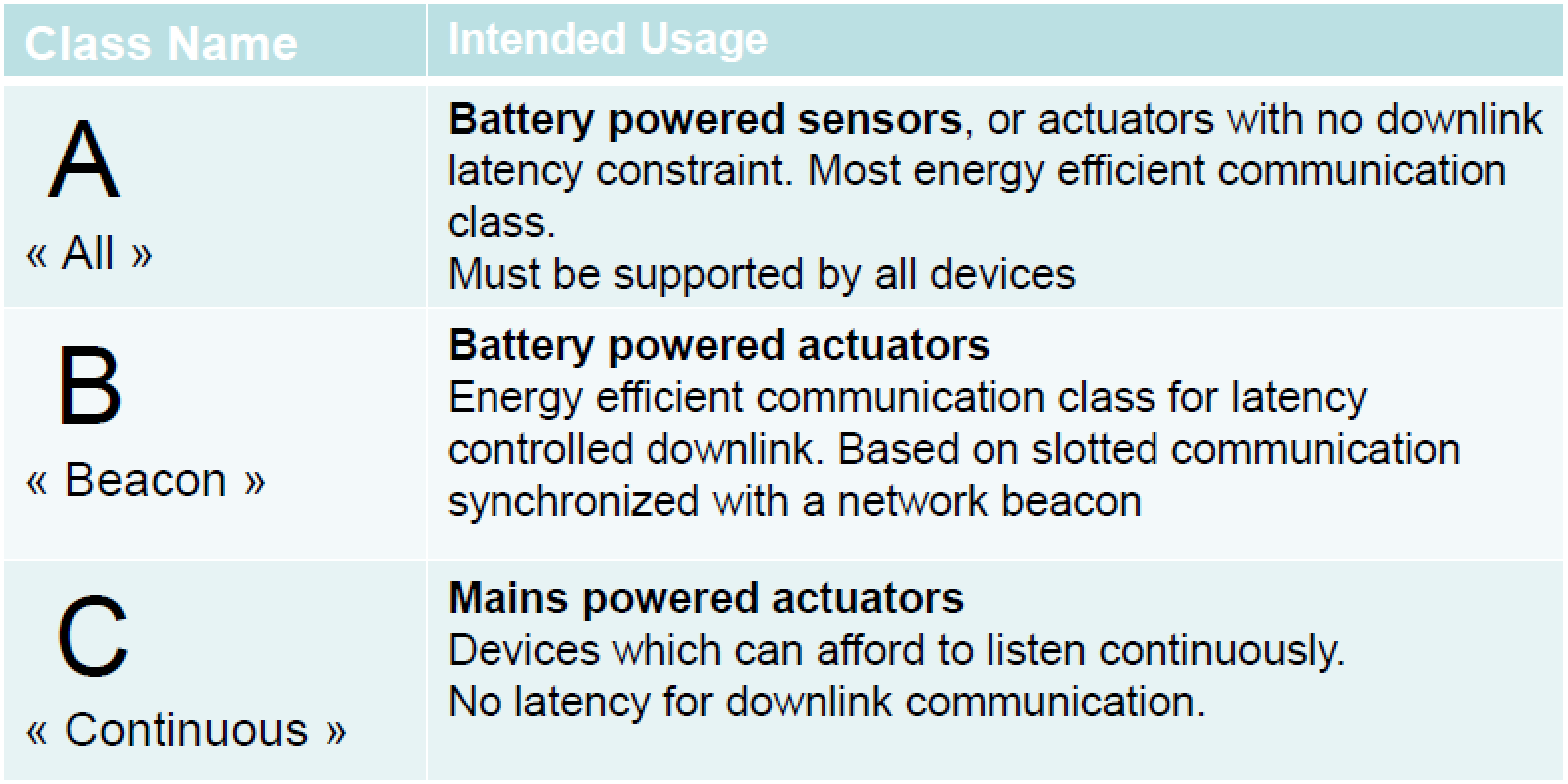
– IOT cannot support connection-based systems, from the energy standpoint
– Scheduling requires negociation and therefore energy
– Scheduling in a license-free band wouldn't be efficient

LoRaWan



Typ. 12 Bytes of Overhead: Lightweight protocol

LoRaWan



Class Name	Intended Usage
A « All »	**Battery powered sensors**, or actuators with no downlink latency constraint. Most energy efficient communication class. Must be supported by all devices
B « Beacon »	**Battery powered actuators** Energy efficient communication class for latency controlled downlink. Based on slotted communication synchronized with a network beacon
C « Continuous »	**Mains powered actuators** Devices which can afford to listen continuously. No latency for downlink communication.

LoRaWan

Class B

Report moisture, t° a few times per day
Turn valves on or off with a few minutes latency
Very low-energy, which depends on latency

Class A

Report status a few times per day
No planned actuation required
Extremely low energy

Class C

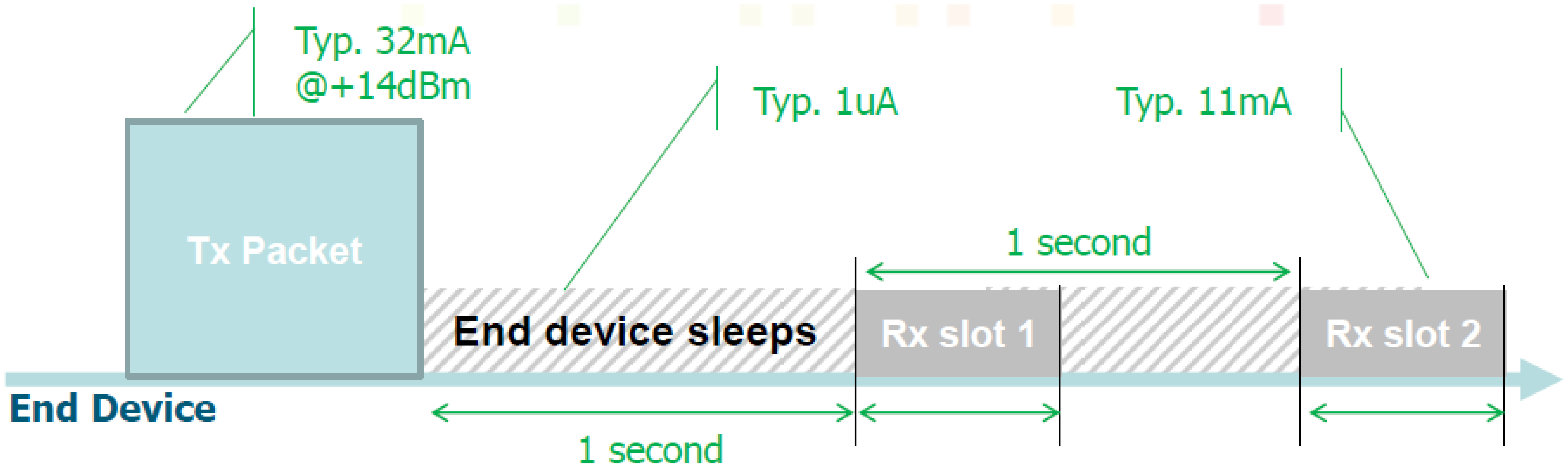
Maintenance and index info a few times / day
Constantly listens for network «ping»
For low-latency actuation

LoRaWan – Class A

Bidirectional communication

- Unicast messages
- End-devices initiates communication (uplink)
- Server communicates with end-device (downlink) during predetermined response windows
- Pros : Lowest power consumption = longest battery life
- Cons : downlink latency driven by the uplink pace

LoRaWan – Class A



Minimum Rx wake-up time = 5 Symbols :

5.1 ms @ SF7

10.2 ms @ SF8

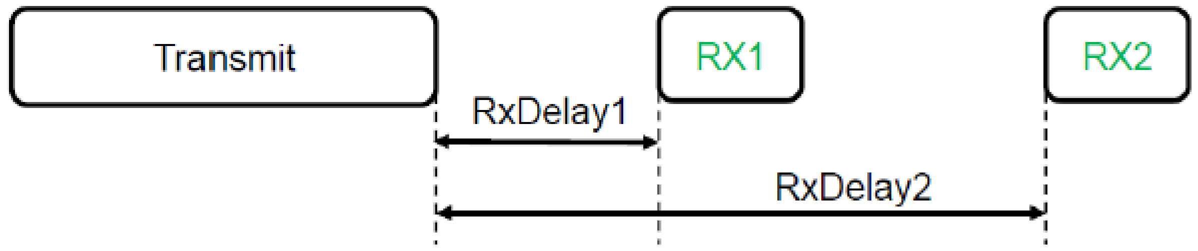
...

164 ms @ SF12

No (wake-up time only) Rx Slot OR Rx Slot 1 OR Rx Slot 2 is used

❑The energy drain when no downlink is << the Tx energy

LoRaWan – Class A



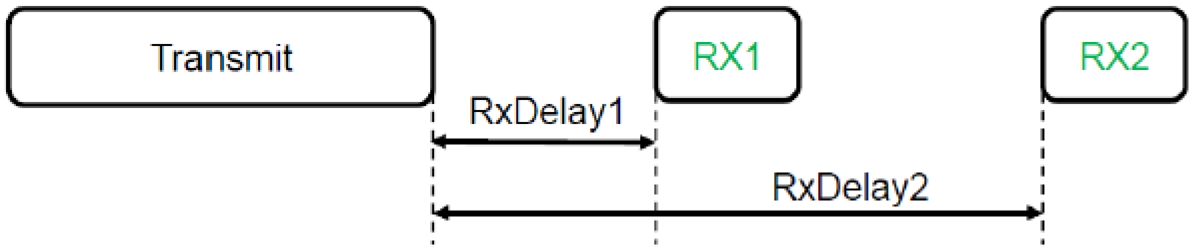
First receive window «RX1» :

✓RxDelay1 is a fixed configurable delay in seconds. Default is 1 second.
✓RX1 frequency uses the same frequency channel as the uplink.
✓RX1 Data Rate is programmable, can equal or lower than the uplink data rate. By default the first receive window data rate is identical to the data rate of the last uplink.

Second receive window «RX2» :

✓RxDelay2 is a fixed configurable delay in seconds. Must be RxDelay1 + 1 second. Default is 2 seconds.
✓RX2 frequency is a fixed configurable frequency.
✓RX2 Data Rate is a fixed configurable data rate.

LoRaWan – Class A



Receive window duration :
The length of a receive wondow must be at least the time required by the end-device’s radio transciever to effectively detect a downlink preamble

Receiver activity during the receive windows :
If a preamble is detected during one the receive windows, the radio receiver stays active until the dowlink frame is demodulated.

Is Second receive window «RX2» mandatory ?
If the end-device did not receive the downlink frame during the first receive window “RX1”, it must open a second receive window “RX2”.
The end-device does not open the second receive window if a frame intended for this end-device has correctly checked the address and MIC (message integrity code) during the first receive window

LoRaWan – Class A

FRMPayload size (Bytes)	240 bps SF12/125k	1 kbps SF10/125k	5.5 kbps SF7/125k
4	~5 uA	~2.2 uA	~1.2 uA
16	~7 uA	~2.5 uA	~1.3 uA
30	~9 uA	~3 uA	~1.4 uA

Assumptions: Pout = +14 dBm, Average Current

- 10 packets / day
- Sleep current ~1uA (includes the MCU)
- MCU is mostly Off during Tx
- No ACK received
- The energy usage of the 2 unused Rx windows is low (<10%)
- Pout = +14 dBm, IDDTX = 32 mA

LoRaWan – Class B

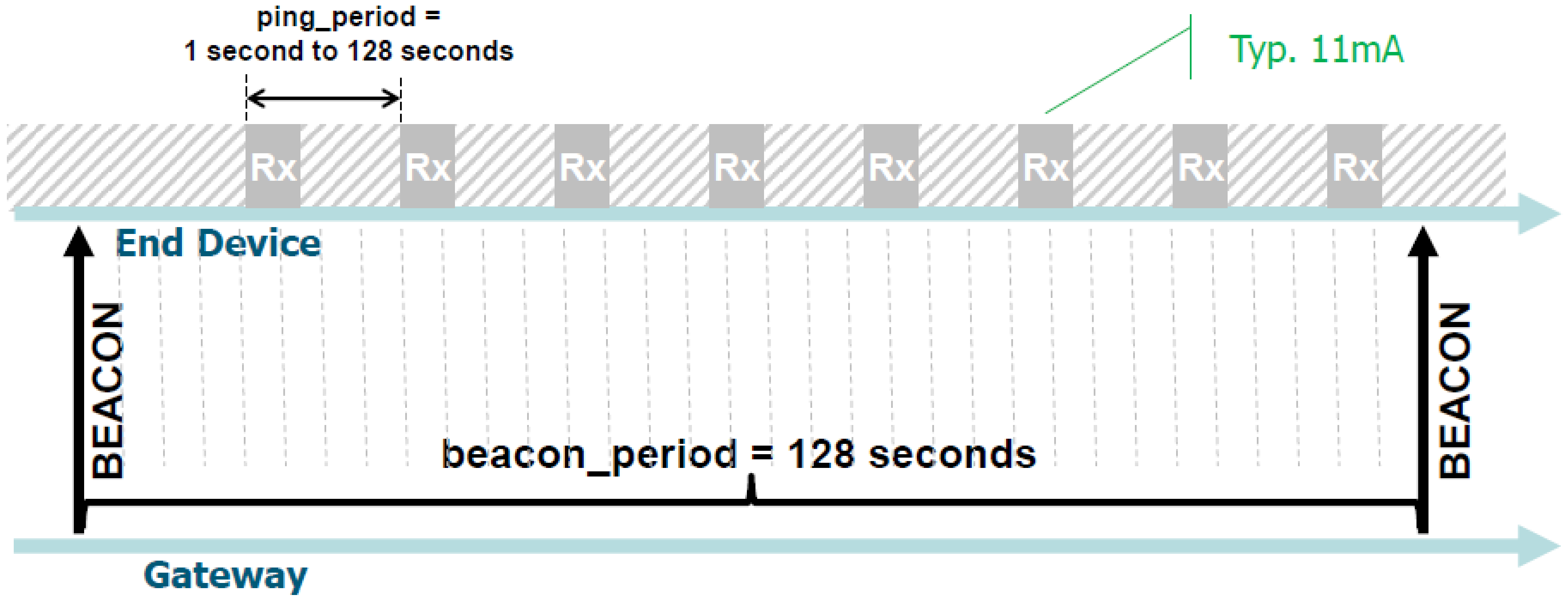
Low downlink latency end-device

Bidirectional communication with scheduled receive slots

- Unicast and Multicast messages
- Periodic beacon from gateway
- Extra receive windows
- Server can initiate transmission at fixed intervals
- Pros : deterministic downlink latency
- Cons : higher power consumption

LoRaWan – Class B

Coordinated Sampled Listening : Network may send downlink packet to node at any Rx slot

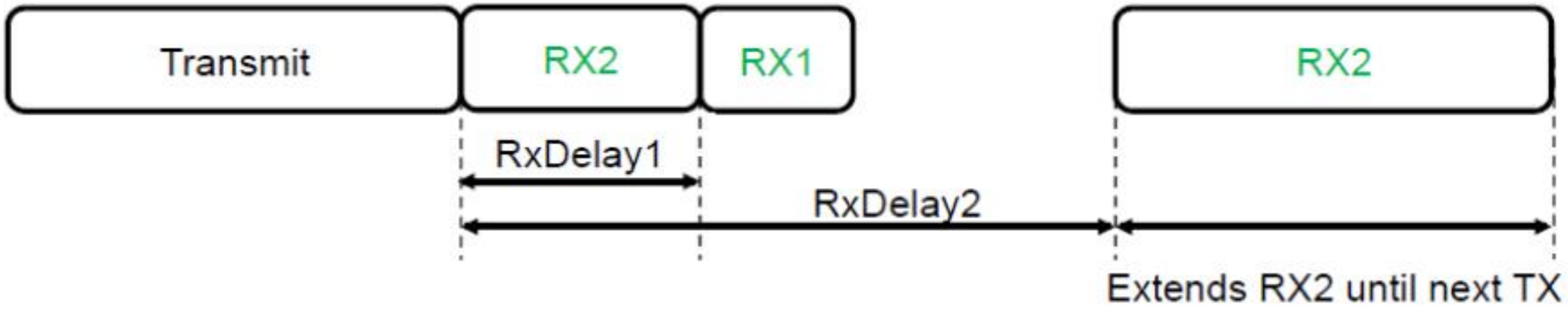


LoRaWan – Class C

No downlink latency end-device

Bidirectional communication

- Unicast and Multicast messages
- Server can initiate transmission at any time
- End-device is constantly receiving
- Pros : Lowest downlink latency
- Cons : highest power consumption (expect end-device to be mains powered)



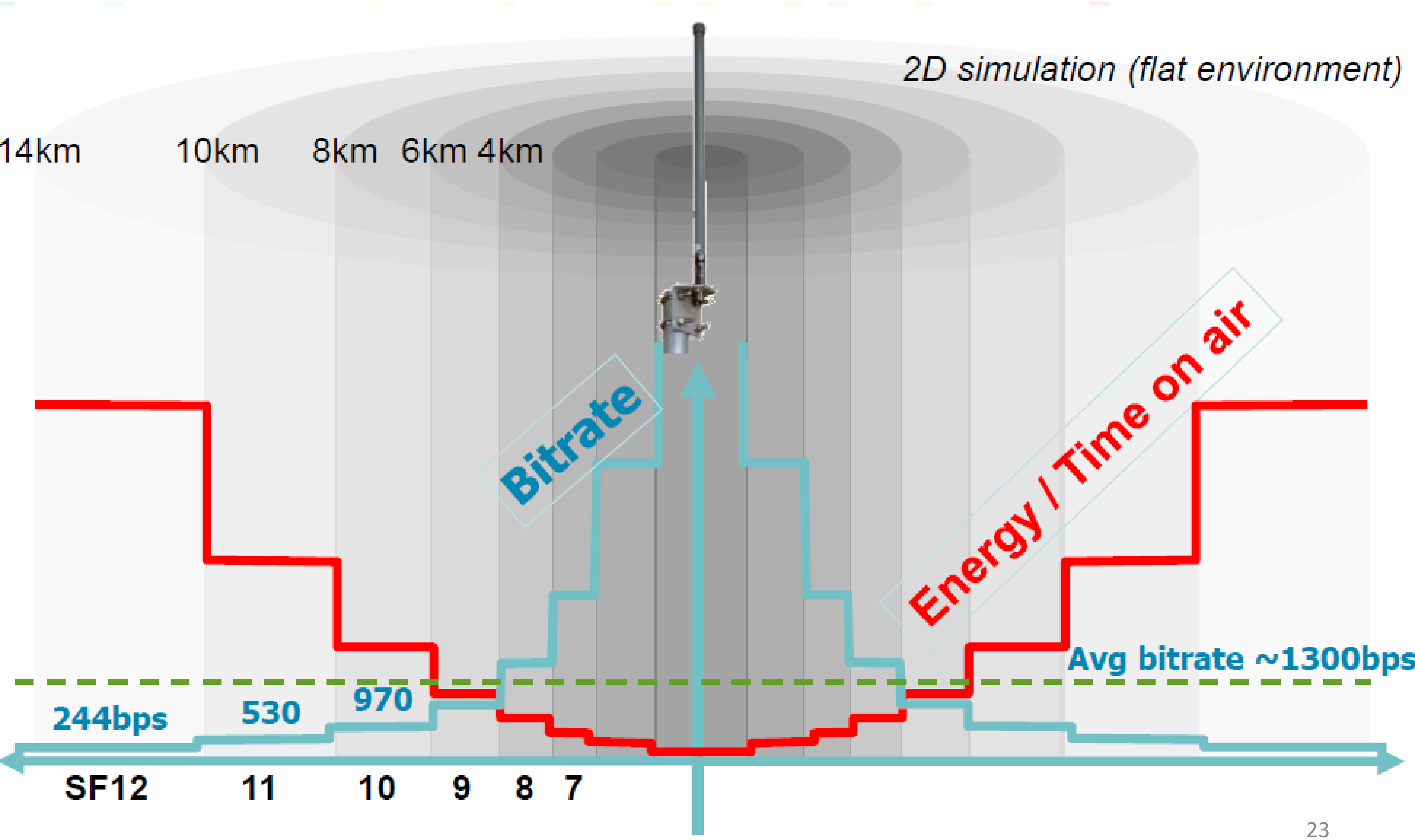
Adaptative datarate

ADR stands for Adaptive Data Rate.
It is the process of adapting the end-device's transmitter output power, data rate and transmit diversity based on the average radio channel attenuation.

- If the average radio link is good the data rate can be increased
- If the demodulation is marginal, the data rate must be lowered

Consequence:
Identical end-devices in different locations might use different data rates / output power

Adaptative datarate



Modulation settings

Longest distance on LoRa modulation :

Data Rate (DR) = 0
•LoRa modulation
•Spreading Factor (SF) = SF12
•Bandwidth (BW) = 125 kHz
•Coding Rate (CR) = 4/5
•Low Datarate Optimize = Enabled

Bit Rate = 244 bps

Receive Sensitivity = -137 dBm

Max Application Payload Size = 51 bytes
•Preambule (programmed) = 8 symbols
•Time On Air = 2466 ms

Modulation settings

Highest Bit Rate on LoRa modulation :

Data Rate (DR) = 6
•LoRa modulation
•Spreading Factor (SF) = SF7
•Bandwidth (BW) = 250 kHz
•Coding Rate (CR) = 4/5
•Low Datarate Optimize = Disabled

Bit Rate = 10938 bps

Receive Sensitivity = -124 dBm

Max Application Payload Size = 222 bytes
•Preambule (programmed) = 8 symbols
•Time On Air = 174 ms

Modulation settings

Highest Bit Rate on GFSK modulation :

Data Rate (DR) = 7
- GFSK modulation

Bit Rate = 50 kbps

Receive Sensitivity = -108 dBm

Max Application Payload Size = 222 bytes
- Time On Air = 39 ms

ADR

ADR must be handled differently for static & mobile end-devices.

- **Static end-devices experience pretty stable radio propagation to the surrounding gateways**
- The ADR is performed by the network server based on the history of the uplink packets received = **Network managed ADR or Static ADR**

This is a slow and gradual process

- **Mobile end-devices experience rapidly variable channel attenuations as they moved.**
- The network based ADR strategy cannot work because of the totally unpredictable channel attenuation as the device moves
- ADR is performed “blindly” on the end-device side = **blind ADR**

Security

- Encryption : One key field in security is encryption. It protects your data from being read everyone. As IoT may handle sensitive data, it's important for you to understand this concept as you may need to protect the information your device sends.

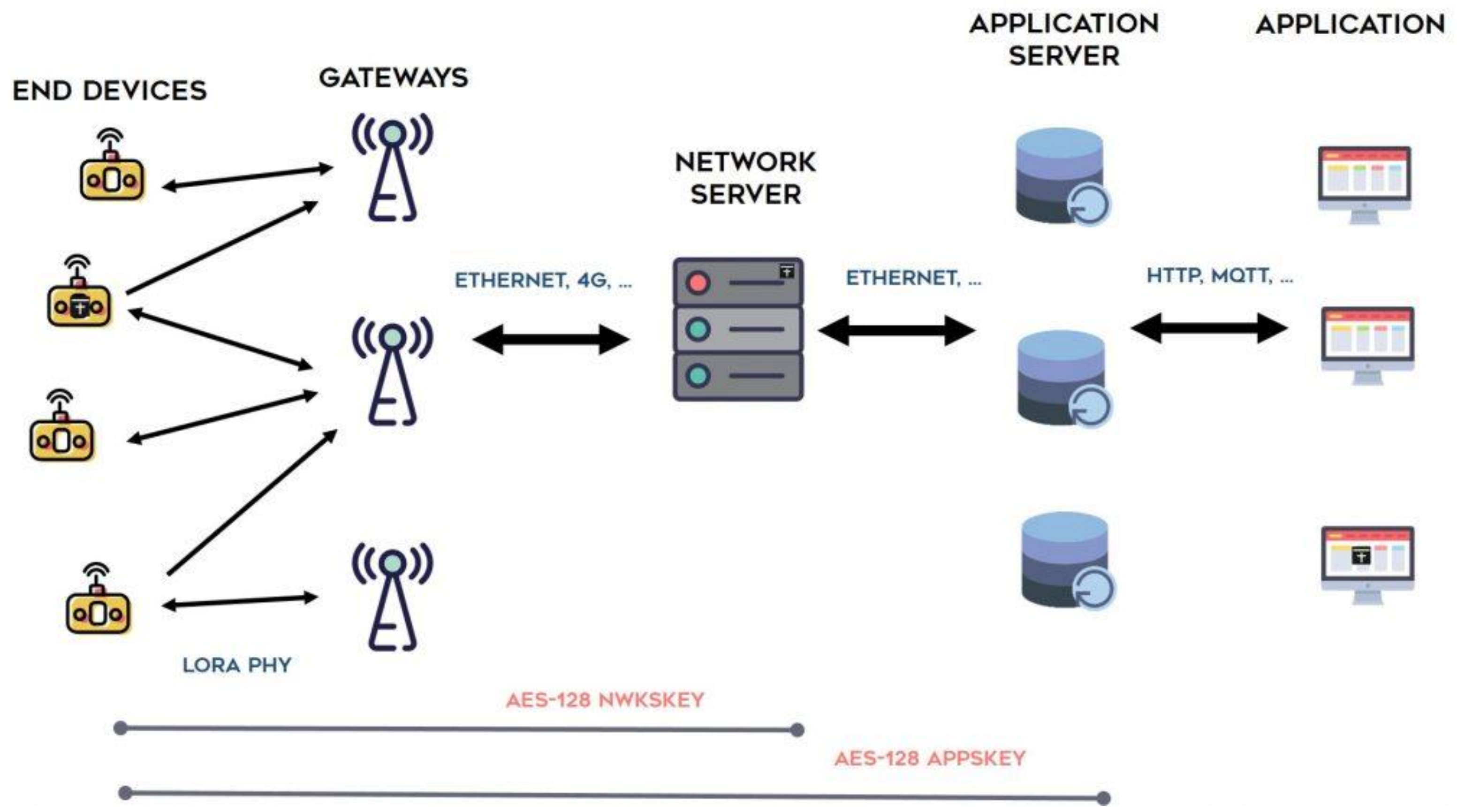
- https://www.makeuseof.com/tag/encryption-care/

- **What is Encryption?**

Encryption is a modern form of cryptography that allows a user to **hide information** from others. Encryption uses a complex algorithm called a cipher in order to turn normalized data (plaintext) into a series of seemingly random characters (ciphertext) that is unreadable by those without a special key in which to decrypt it. Those that possess the key can decrypt the data in order to view the plaintext again rather than the random character string of ciphertext.

Security

Step1: Encryption: only FRMPayload is encrypted (1 to 1) with the 128-bit AppSKey
Step 2: Authentication: MIC is generated using the 128-bit NwkSKey, and appended to the frame for packet authentication



Security

1. DevAddr: Device Address
   - 32-bit identifier stored in the end-device
   - Uniquely identifies the device on the network
   - Transported in each frame to and from the end-device
   - Obtained during the Activation process

2. NwkSKey: Network Session Key, 128-bit AES encryption key
   - Specific to the end-device
   - Used to encrypt/decrypt payload of MAC-only messages (port 0 between end-device MAC and Network controller)
   - Used to calculate the MIC to ensure message integrity
   - Obtained during the Activation process

3. AppSKey: Application Session Key, 128-bit AES encryption key
   - Specific to the end-device
   - Used to encrypt/decrypt payload of application messages
   - Obtained during the Activation process

Security

DevAddr : Device 32 bits device address on a network. This address is dynamically set by the network operator during the on-boarding process .The same DevAddr can be reused in a network or across different networks. In case of address collision, the cryptographic signature allows to disambiguate

DevAddr:	**NID** **7bits**	**end-device number** **25bits**

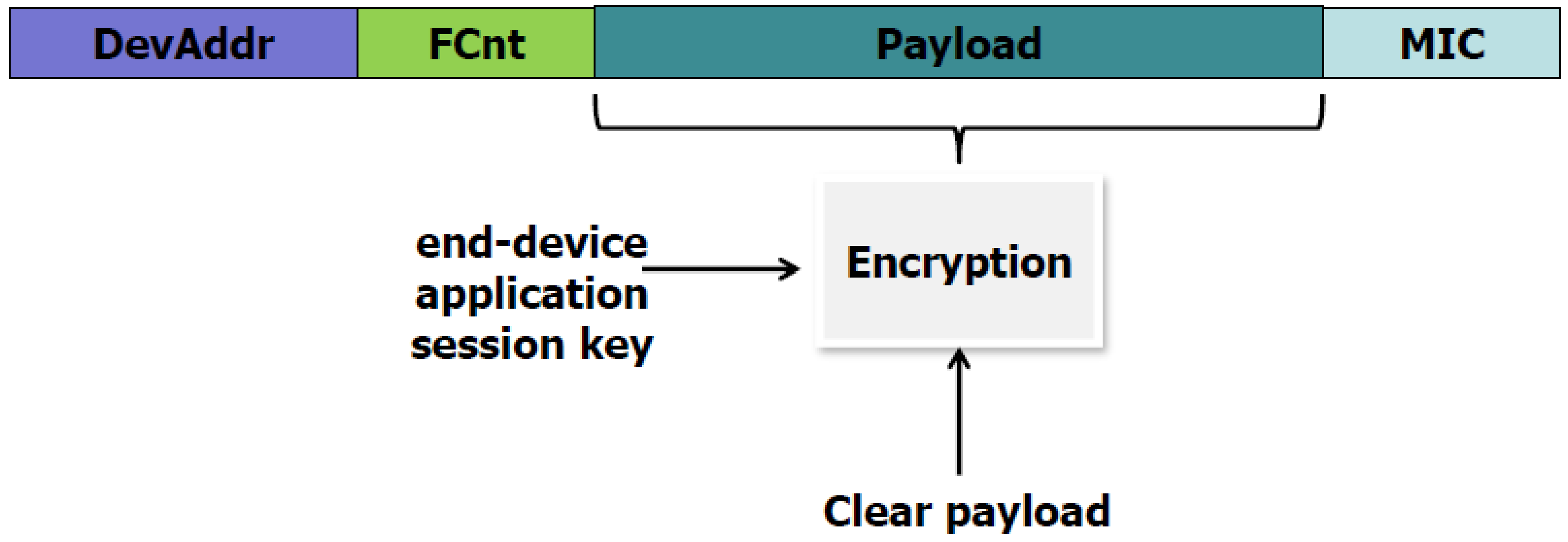
NID : The NID (Network Id) consists of the 7 LSB of the operator network ID (NetID) which is a 3 bytes unique network identifier allocated by the LoRa alliance . Two different networks must have different NetID but may end up with the same NID. The NID field eases the roaming process by reducing the amount of signaling required between Network Servers. Experimental/Private network must use 0000000 or 0000001 for NID.

Security

The **NwkSKey** is a **network session key** specific for the end-devices. It is used by both the network server and the end-device to calculated and verify the **MIC** (message integrity code) of all data messages to ensure data integrity. It is further used to encrypt and decrypt the payload field of a MAC-only data messages.

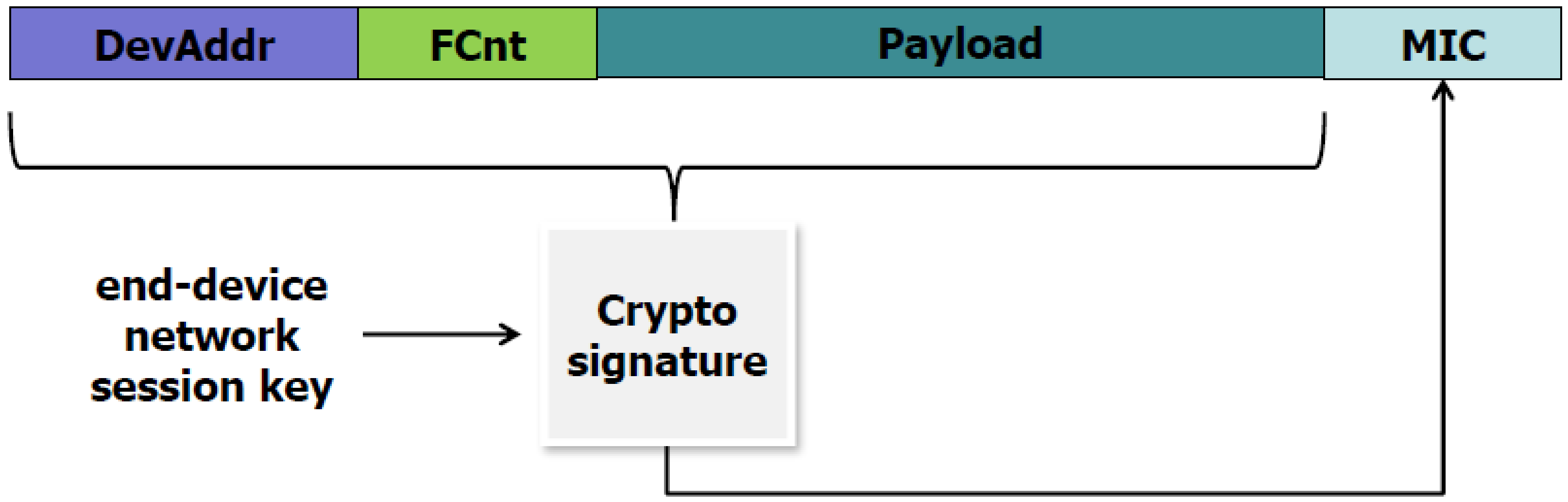
The **AppSKey** is an application session key specific for the end-devices. It is used by both the application server and the end-device to encrypt and decrypt the payload field of application-specific data messages. It is also used to calculated and verify an application level MIC that may be included in the payload of application –specific data messages.

Step 1 : Payload Encryption



All payloads are encrypted using an AES algorithm with a 128bits secret key, the Application Session Key
Each end-device has its own unique Application Session Key only known by the end-device and the application server and only used for encryption

Step 2 : Packet authentification



All frames contain a 32bits cryptographic MIC signature computed using an AES algorithm with a 128bits secret key, the Network Session Key
The Network Session Key is different from the Application key Session
Each end-devices has its own Network Session Key only known by the end-device and the network server and only used for signature

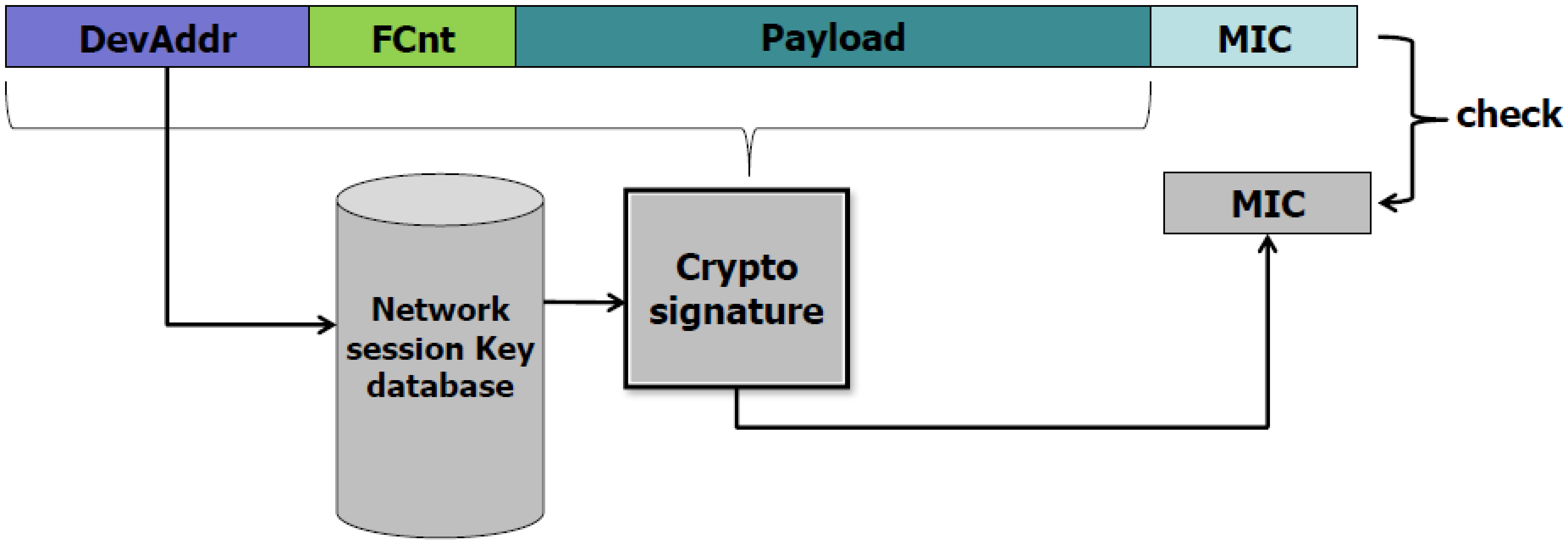
Frame signature

The frame signature (Message Integrity Code “MIC”) is computed over the entire frame

- Therefore the frame cannot be modified in any way between the end-device and the network server without compromising the signature.
- The only components involved are:
- The end-device
- The network server
- The gateway, the gateway backhaul link, etc, … are totally transparent from a security perspective

Authentification of the uplink

Upon reception of a frame, the network server checks that the frame received MIC signature matches the one computed using the end-device's network session key contained in its key database



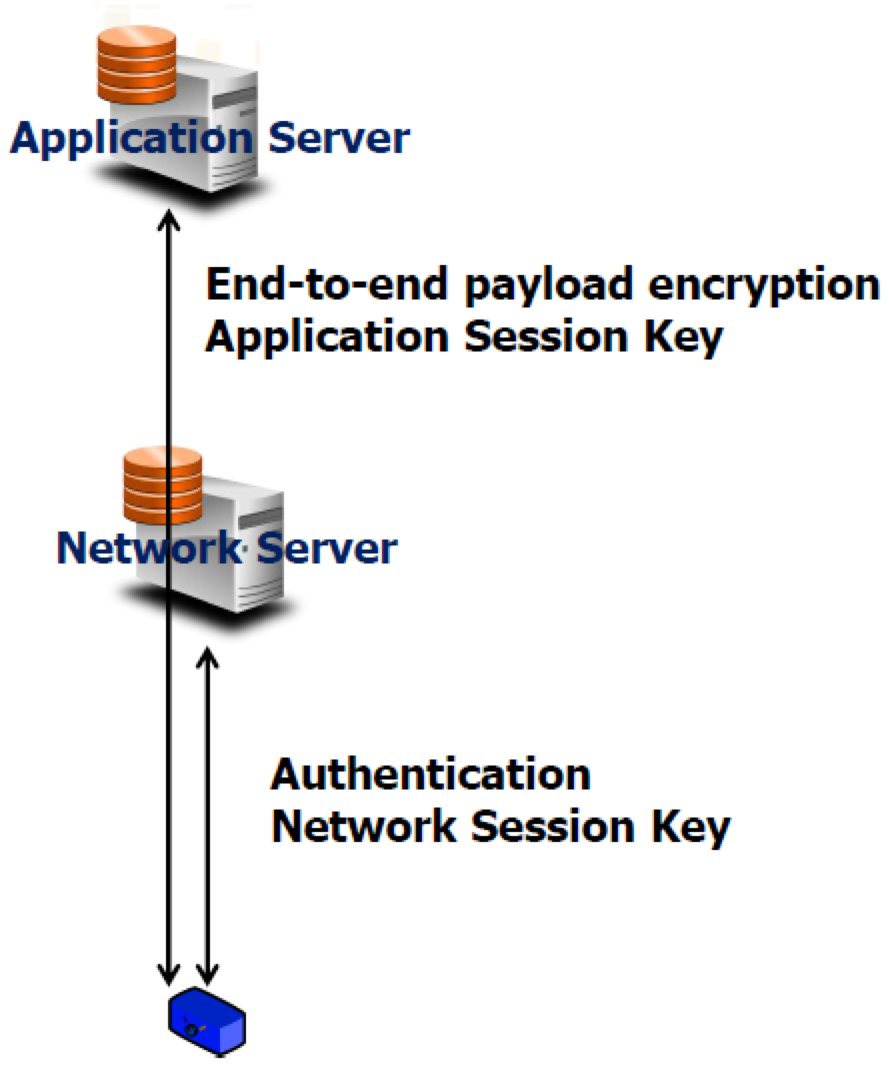
If the 2 MICs match then the frame is really coming from the legitimate end-device and its content hasn't been modified in any way

Authentification of the downlink

- **The same process happens on the down-link.**
- **The network server signs each frame with a MIC computed over the entire transmitted packet using the destination end-device’s Network Session Key.**

This way, the server can differentiate a legitimate end-device from one trying to steal the end-device’s DevAddr (cloning attack).
The end-device can check that the commands coming from the network are legitimate.
Additionally each frame contains a frame counter forbidding “replay” attacks.

End to End security



End-device personalization and Activation

To participate in a LoRaWAN network, each end-device has to be personalized and activated :

✓Personalization : End-device is configured at production time.

✓Activation : Before an end-device can communicate on a LoRaWan network, it must be actived.

End-device Activation : 2 methods

By
Personalization
«ABP»

Over the Air
«OTAA»

Activation By Personalization

Network authority and Application provider negociate

- A set of DevAddr (for instance 1000 Addresses, unique to the network)
- A set of AppSKey
- A set of NwkSKey

Advantage: devices are already pre-commissionned
Drawback: roaming from network to network is impossible

Activation By Personalization

This is the simplest scenario , targeting mainly B2B use cases.

The device is registered by the service provider with a given network operator before being personalized.

The operator directly provides the correct DevAddr and network/app session keys The device is directly personalized at fabrication with the correct DevAddr and network+application session keys.

The device is fully functional on the network as soon as powered-up.

The device is therefore tied to a given network and service provider

Over the Air Activation “OTAA”

The join procedure requires the end-device to be personnalized with the following information before it starts the join procedure : a globally unique end-device identifier (DevEUI), the application identifier (AppEUI) and a an AES-128 key (AppKey)

- **DevEUI:** unique source address identifying the node of the application provider
- **AppEUI:** shared secret, destination address, globally unique identifier owned by the application provider. AppEUI routes the Join Req to the right Join Server
- **AppKey:** used forJoinReq and JoinAccept authentication

Warning ! AppKey is different from AppSKey

Over the Air Activation “OTAA”

DevEUI : Device 64 bits unique device identifier. Equivalent to the MAC address of a network adapter card. This address is written in the device by the manufacturer at the same time than the firmware. It consists of a 24bits Organizationally Unique Identifier (*) and a 40 bit serial number. The OUI code is unique for any manufacturer and provided by the IEEE and the serial code is set freely by the manufacturer.

DevEUI:	**O.U.I 24bits**	**Serial number 40bits**

Over the Air Activation “OTAA”

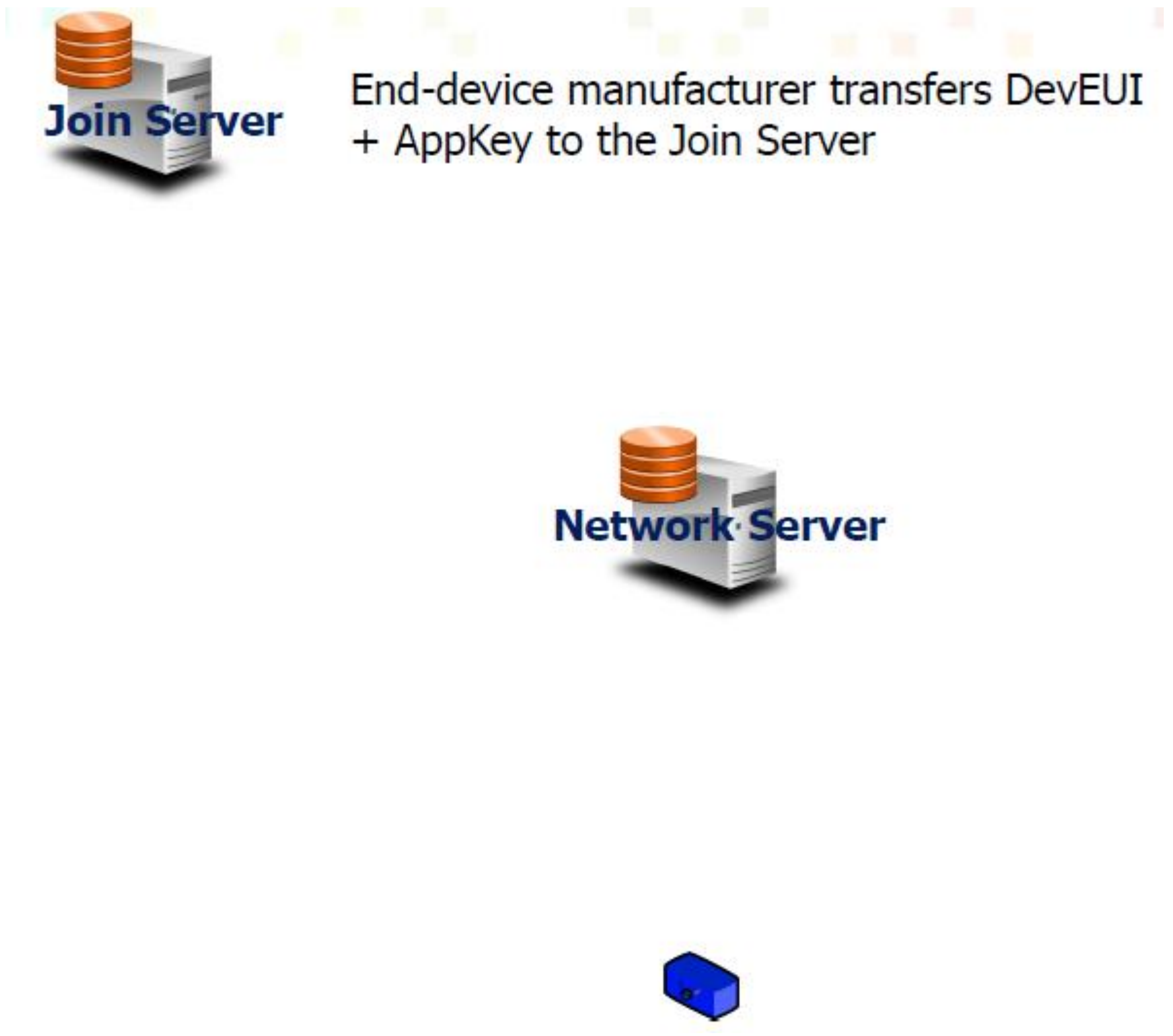
AppEUI : Join Server 64 bits unique server identifier. This number uniquely identifies a Join Server It consists of a 24bits Organizationally Unique Identifier (*) and a 40 bit serial number. The OUI code is the manufacturer’s code of the entity operating the Join Server. The serial number can be used if several Join Server are operated by the same entity. For example Semtech might operate a Join Server with AppEUI : 00 16 C0 00 00 00 00 01 “00 16 C0” is Semtech’s OUI

AppEUI:	**O.U.I** **24bits**	**Serial number** **40bits**

Over the Air Activation “OTAA”

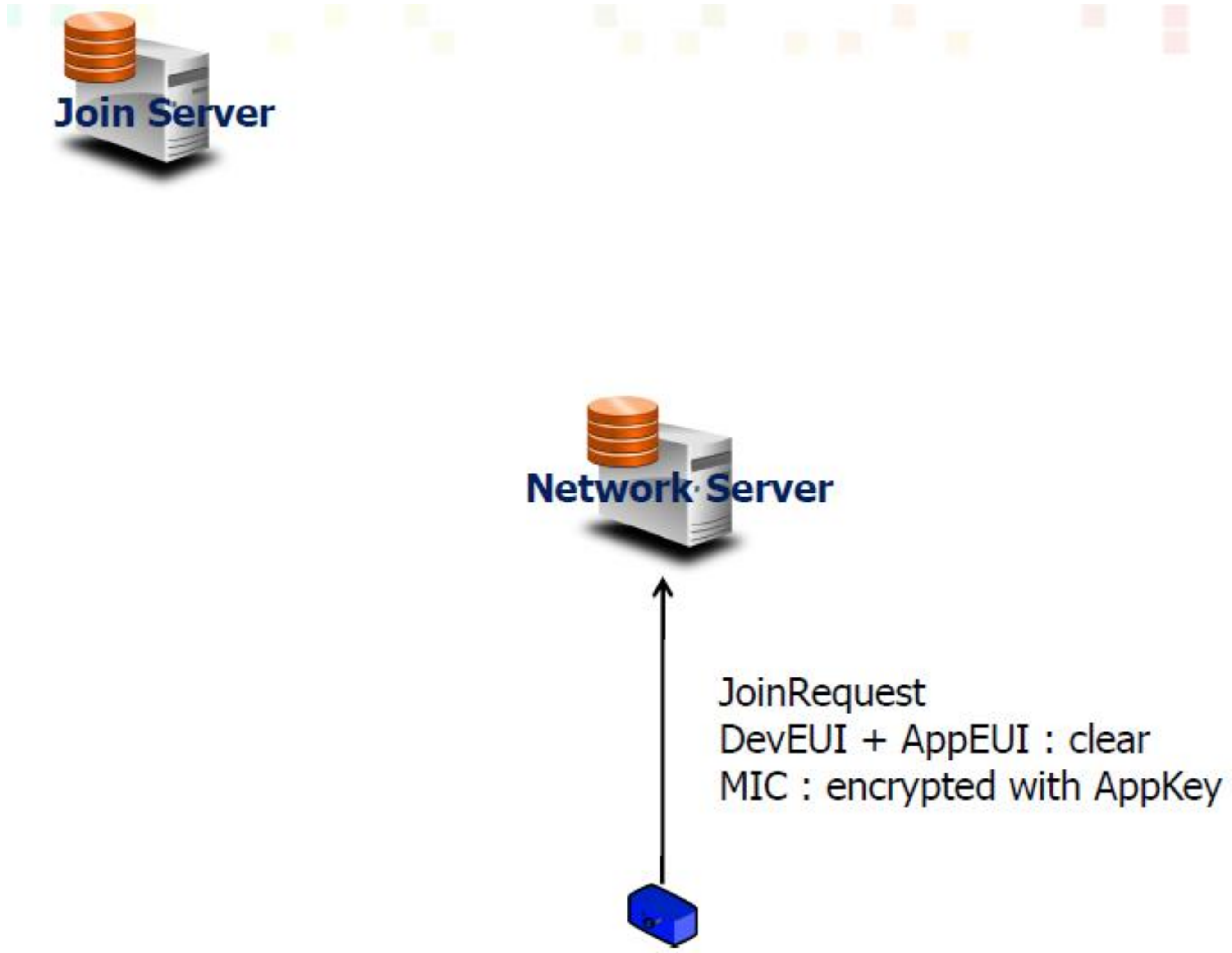
The **AppKey** is an AES-128 application key specific for the end-device that is assigned by the application owner to the end-device and most likely derived from an application specific root key exclusively known to and under control of the application provider.
Whever an end-devices joins a network via OTAA, the AppKey is used to derive the session keys NwkSKey and AppSKey specific for the end-device to encrypt and verify network communication and application data.

Over the Air Activation “OTAA”

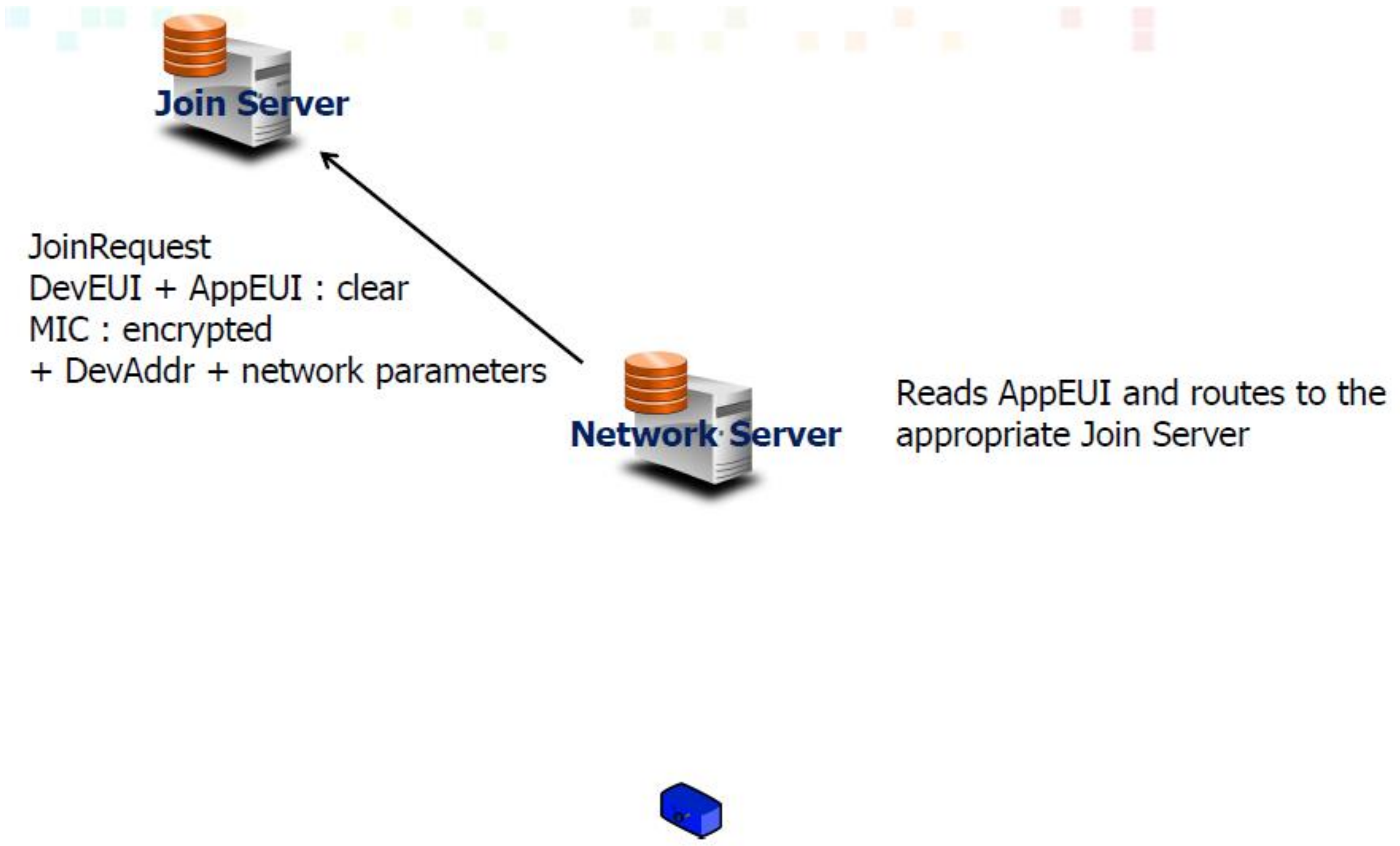


End-device manufacturer transfers DevEUI + AppKey to the Join Server

Over the Air Activation “OTAA”



Over the Air Activation “OTAA”

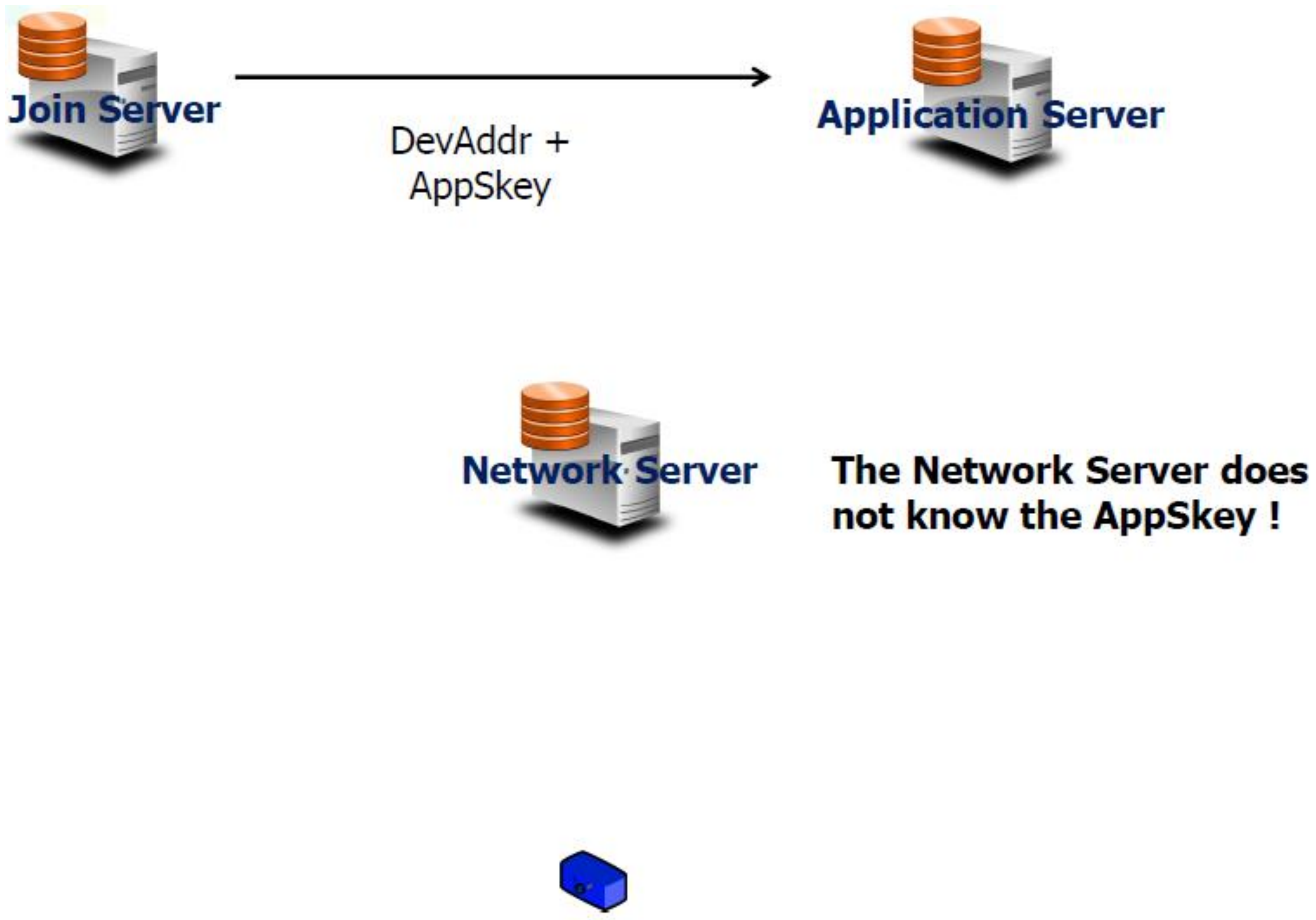

Join Server
JoinRequest
DevEUI + AppEUI : clear
MIC : encrypted
+ DevAddr + network parameters
Network Server
Reads AppEUI and routes to the appropriate Join Server

Over the Air Activation “OTAA”

Join Server
Network Server

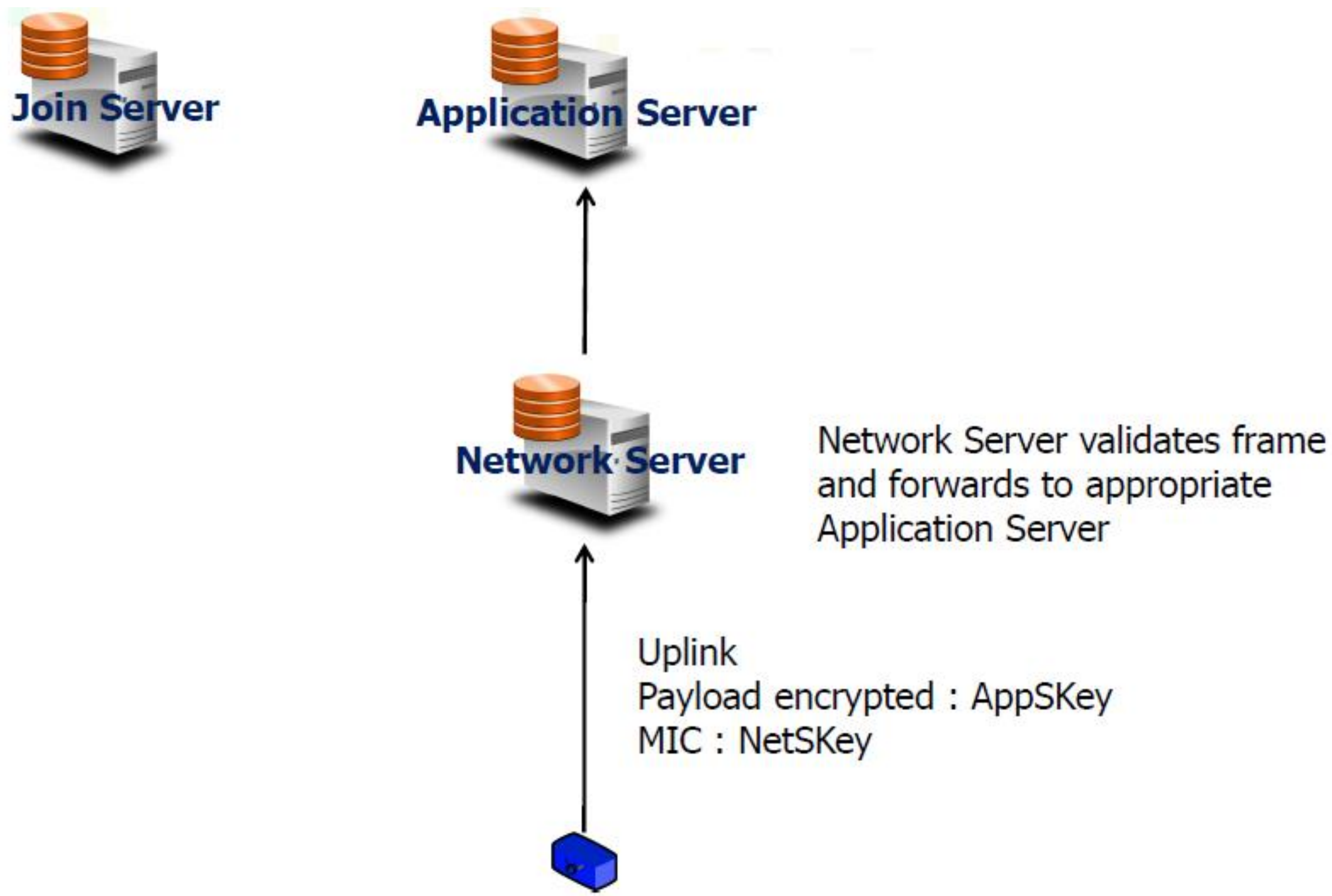
Device derives the same AppSkey and NetSKey from content of JoinResponse + AppKey

Over the Air Activation "OTAA"



The Network Server does not know the AppSkey !

Over the Air Activation "OTAA"


Join Server
Application Server
Network Server
Network Server validates frame and forwards to appropriate Application Server
Uplink
Payload encrypted : AppSKey
MIC : NetSKey

Over the Air Activation “OTAA”

In this scenario the device is specific of a given service provider but is activated over-the-air on any network.
The device’s devEUI and AppKey (the device root key) are allocated autonomously by the device manufacturer.
The device’s AppEUI is set to point to the wanted Join Server (selected or operated by the service provider).
The device is service provider specific. The device is purchased by the service provider along with its devEUI and AppKey.
The device is activated over the air with the chosen operator. The same device can operate on any network and in different countries.

The selection of the operator is performed during the JoinReq – JoinResponse message exchange.