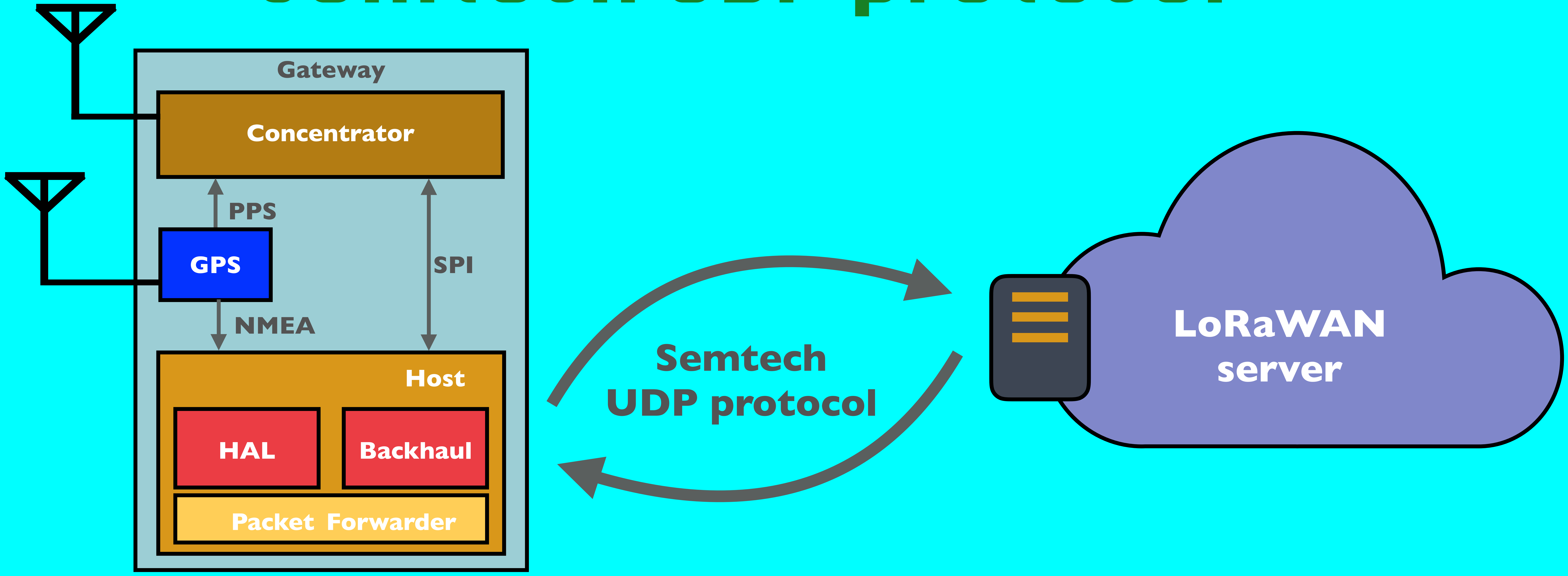
mobilefish.com

LORA / LORAWAN TUTORIAL 29

Semtech UDP Packet Forwarder & Semtech UDP protocol



v1.1.0

INTRO

- In this tutorial I will briefly explain what the Semtech UDP Packet Forwarder is.
  I have already explained this in tutorial 28.

- But the main focus is explaining what the Semtech UDP protocol is.

SEMTECH UDP PACKET FORWARDER

- A packet forwarder is a program running on the host of a LoRa gateway and interfaces with the LoRa concentrator to pull and push packets, while interacting at the same time with the network server.

- The Semtech Corporation created the first packet forwarder, which is a reference design and is called the "Semtech UDP Packet Forwarder".

- How the packet forwarder and LoRaWAN network server communicates with each other are defined by a set of rules also known as communication protocol.

- When a LoRa gateway uses the Semtech UDP Packet Forwarder, it connects to a LoRaWAN network server through the Semtech UDP protocol.
  See: https://github.com/Lora-net/packet_forwarder

SEMTECH UDP PACKET FORWARDER

- However the Semtech UDP Packet Forwarder has several flaws, for example UDP is not secure, UDP is not reliable and the forwarder is hard to configure.
  More information:
  https://www.thethingsnetwork.org/docs/gateways/start/connection.html

SEMTECH UDP PROTOCOL VERSION 2

- More information about the Semtech UDP protocol:
  - The Gateway to Server Interface Definition [6]
  - https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT
  - https://github.com/Lora-net/packet_forwarder/blob/master/lora_pkt_fwd/src/lora_pkt_fwd.c
- In this tutorial the focus will be on the Semtech UDP protocol version 2.
  This is the protocol version used since Semtech UDP packet forwarder version 3.0.0.
- In the next slides you will find tables with the description of the JSON object keys.
  This is based on the above mentioned 3 sources.

LEGACY PACKET FORWARDER

- Several developers forked the Semtech UDP packet forwarder and implemented new functionalities.

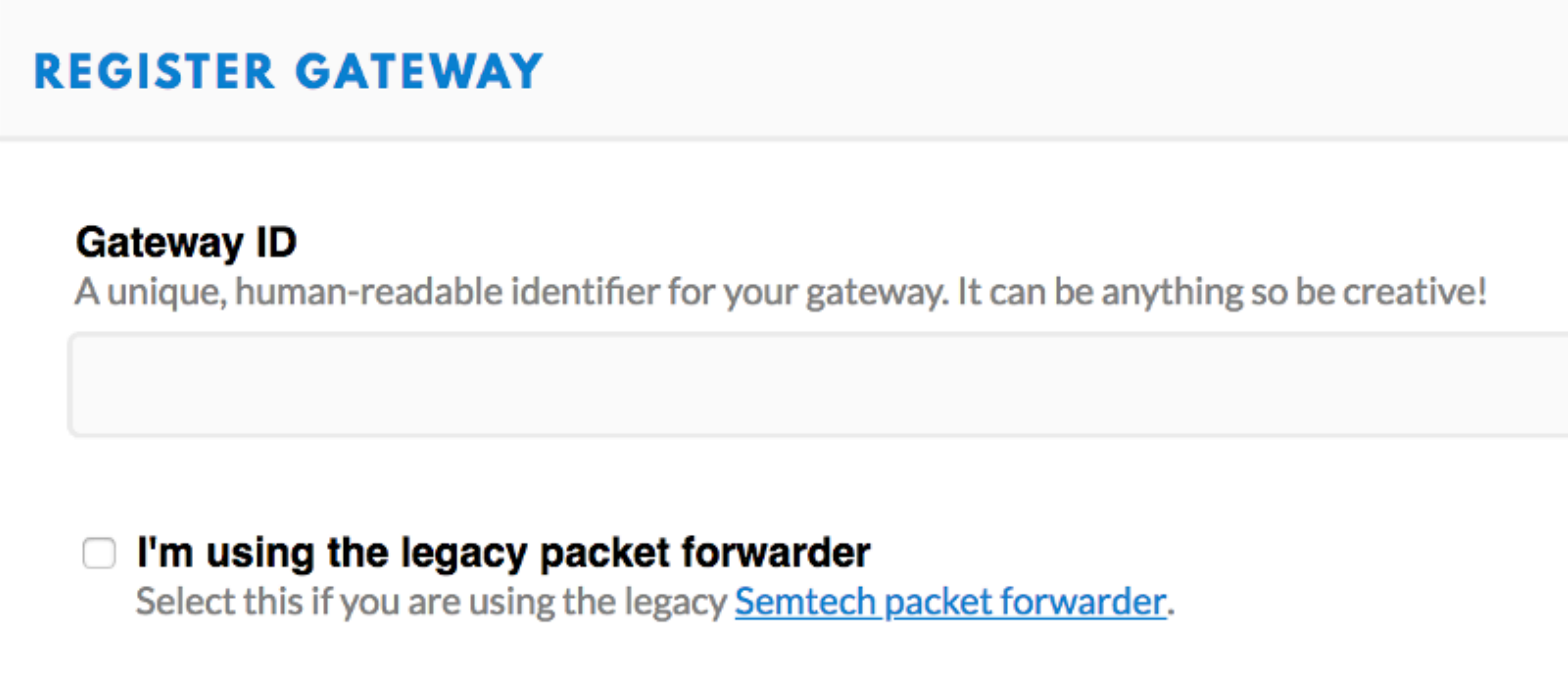
- All these forked packet forwarders are using the Semtech LoRa Gateway library (= libloragw.a). See: https://github.com/Lora-net/lora_gateway

- A packet forwarder which uses the Semtech UDP protocol is called "legacy packet forwarder".

- The Things Network has developed another protocol called "Gateway Connector Protocol" to avoid the UDP disadvantages. Packet forwarders using this protocol are NOT legacy packet forwarders. More information: https://www.thethingsnetwork.org/docs/gateways/start/connection.html

LEGACY PACKET FORWARDER

- When registering a gateway in TTN console and the gateway uses the legacy packet forwarder, meaning it uses the Semtech UDP protocol, than check the box ”I'm using the legacy packet forwarder”.

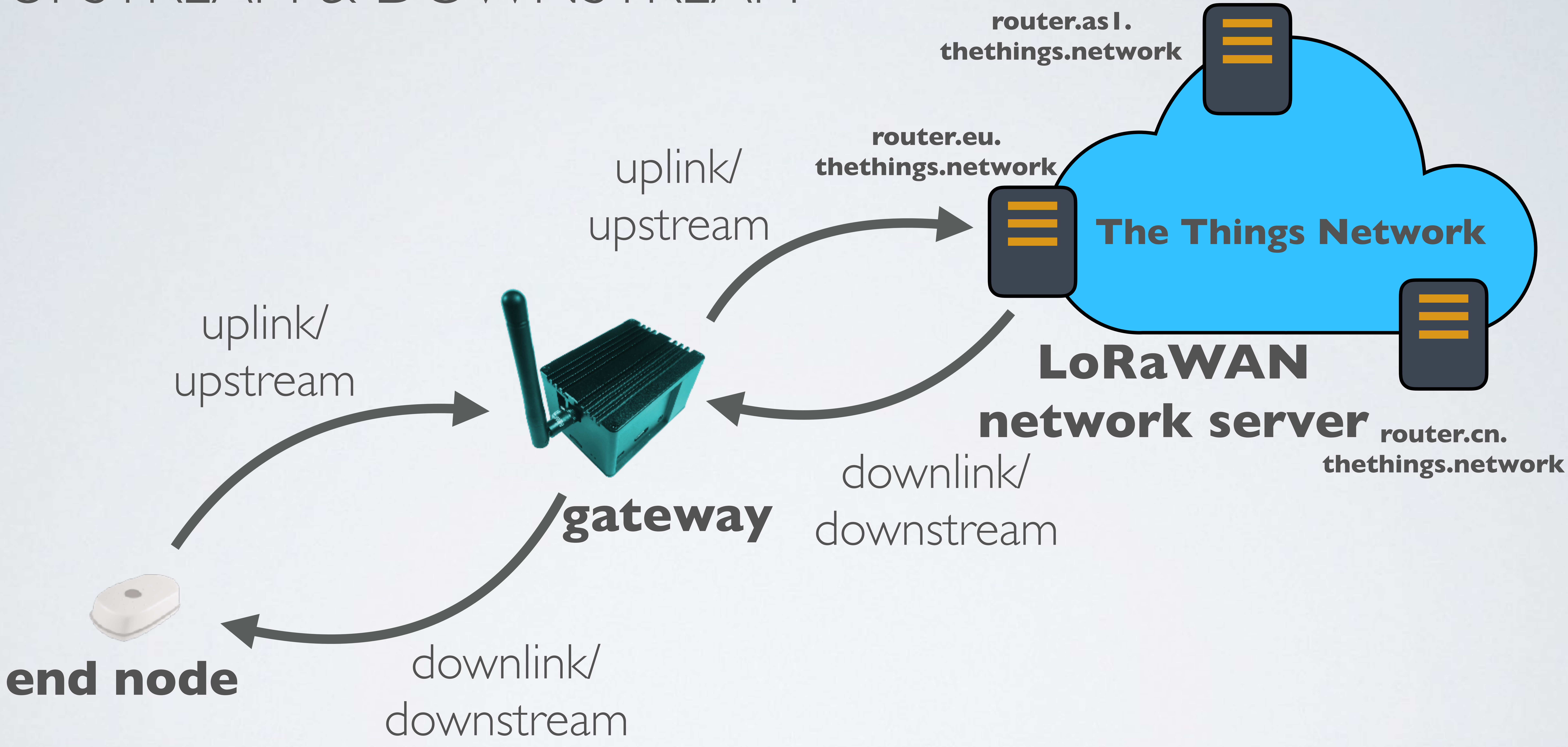


REGISTER GATEWAY

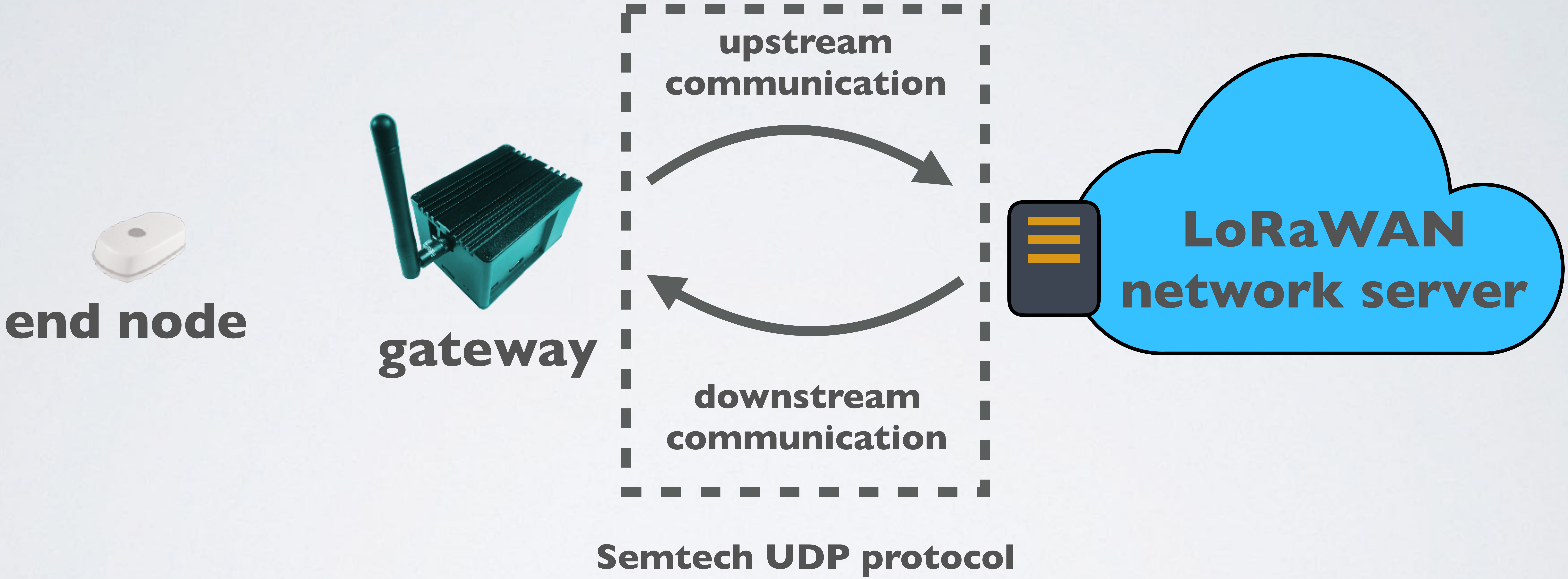
Gateway ID
A unique, human-readable identifier for your gateway. It can be anything so be creative!

□ **I'm using the legacy packet forwarder**
Select this if you are using the legacy Semtech packet forwarder.

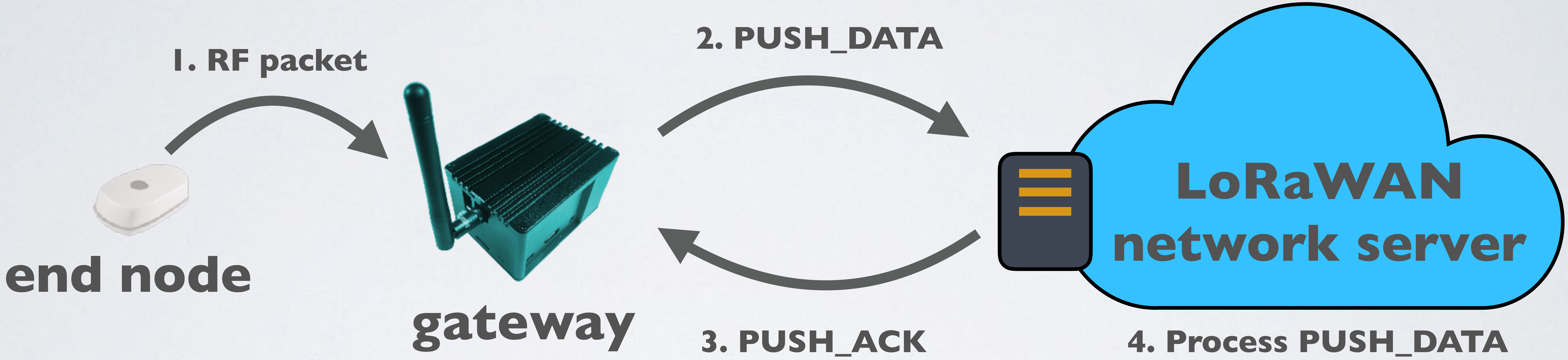
UPSTREAM & DOWNSTREAM



SEMTECH UDP PROTOCOL VERSION 2



UPSTREAM COMMUNICATION



1. RF packet

end node

gateway

2. PUSH_DATA

3. PUSH_ACK

LoRaWAN network server

4. Process PUSH_DATA

UPSTREAM COMMUNICATION

- (1) When a gateway receives a RF packet from an end node, (2) the gateway creates a PUSH_DATA packet which is sent to a LoRaWAN network server.

- (3) After the server received the PUSH_DATA packet, the server sends a PUSH_ACK back to the gateway and (4) then processes the PUSH_DATA packet.

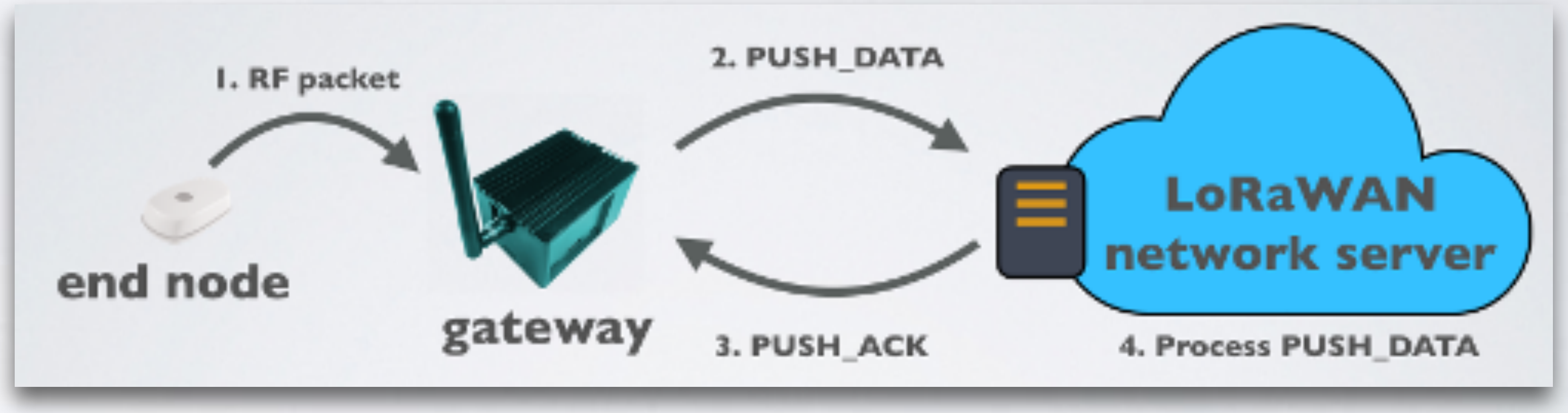
PUSH_DATA AND PUSH_ACK MESSAGE FORMAT

Byte 0	1-2	3	4-11	12-end
protocol version 0x02	Random token	PUSH DATA 0x00	Gateway EUI	JSON object rxpk and/or stat

PUSH_DATA packet
Max packet size = 2408 bytes

Byte 0	1-2	3
protocol version 0x02	Token from PUSH_DATA	PUSH ACK 0x01

PUSH_ACK packet
Packet size = 4 bytes



These are called identifiers.

The Semtech UDP protocol version 2 is used, in the packet this is represented by 0x02.

PUSH_DATA JSON OBJECT

- The PUSH_DATA JSON object can contain either or both:
  - an array called **rxpk** (received packet) which contains one or more JSON objects each containing an RF packet and associated metadata.
  - an object called **stat** (status) which contains the status of the gateway.

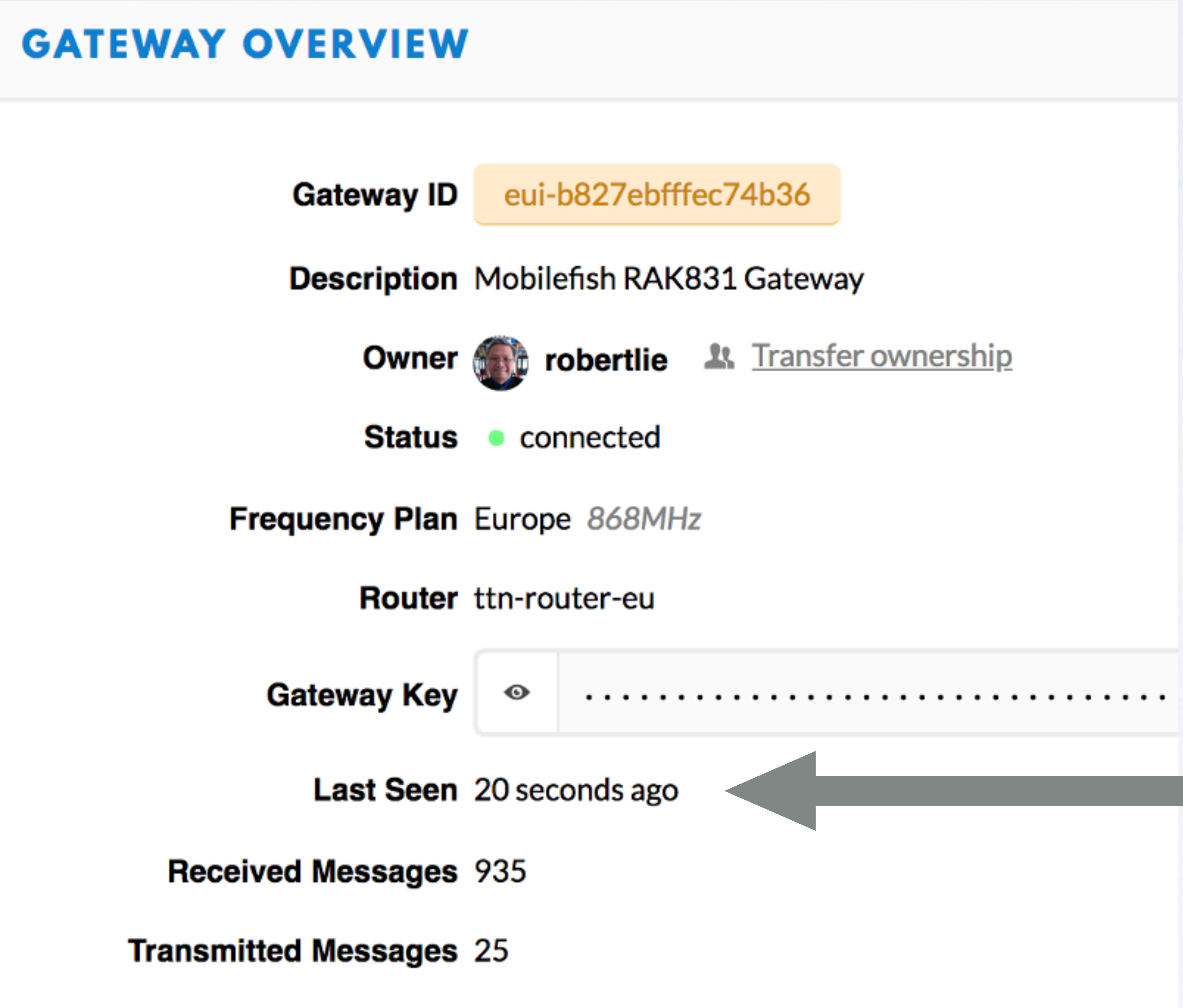
```
PUSH_DATA JSON object:
{
  "rxpk":[ {...}, ...],
  "stat":{...}
}
```

PUSH_DATA JSON OBJECT

- In LoRa systems JSON objects can only use ASCII characters.

- At regular time intervals the stat object is send to the server. This is set by the **stat_interval** key in the global_conf.json or local_conf.json file.

STAT_INTERVAL



GATEWAY OVERVIEW

Gateway ID eui-b827ebfffec74b36

Description Mobilefish RAK831 Gateway

Owner robertlie Transfer ownership

Status ● connected

Frequency Plan Europe *868MHz*

Router ttn-router-eu

Gateway Key ....

Last Seen 20 seconds ago

Received Messages 935

Transmitted Messages 25

**Nice to know,
if stat_interval = 30,
the Last Seen status will be
updated every 30 seconds.**

RXPK JSON OBJECT KEYS

Name	Required	Type	Function
time	Yes, if GPS enabled	string	Transform RX packet internal counter based timestamp to UTC time, microseconds precision. ISO 8601 'compact' format.
tmms	Yes, if GPS enabled	unsigned integer	Transform RX packet internal counter based timestamp to GPS time, number of milliseconds since 06 Jan 1980
tmst	Yes, if GPS enabled	unsigned integer $<2^{32}$	Gateway internal time counter at the instant the radio packet was received. Value will rollover approximately every 72 minutes.
freq	Yes	unsigned float	Received signal centre frequency in MHz
chan	Yes	unsigned integer	Concentrator "IF" channel used for RX
rfch	Yes	unsigned integer	Concentrator "RF chain" used for RX
stat	Yes	signed integer	CRC status: 1 = OK, -1 = fail, 0 = no CRC
modu	Yes	string	Modulation identifier "LORA" or "FSK"

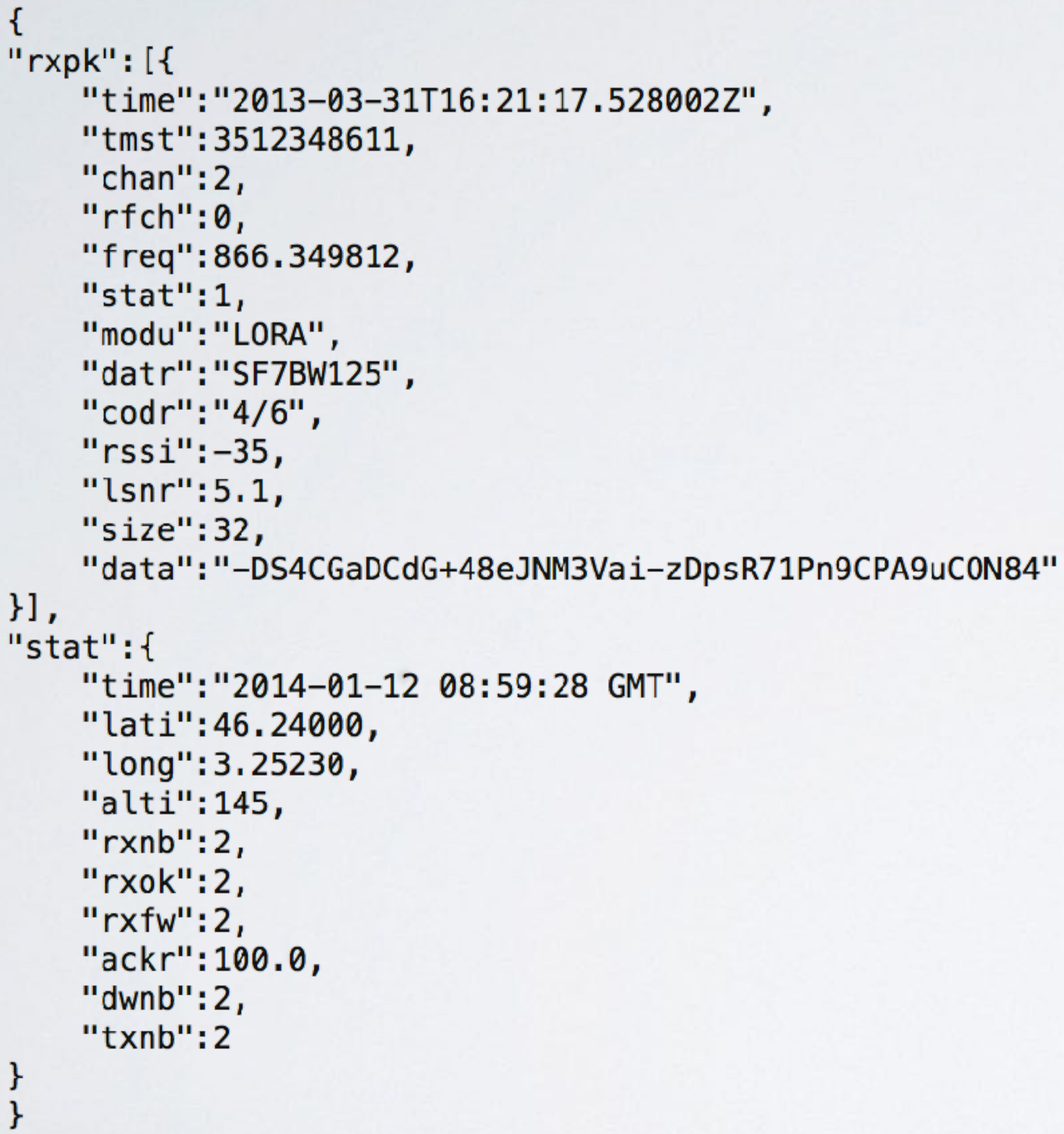
RXPK JSON OBJECT KEYS

Name	Required	Type	Function
datr	Yes	string / unsigned integer	Datarate identifier. If modu=LORA, datr comprises a string "SFnBWm", where n=spreading factor and m=bandwidth in kHz If modu=FSK, datr comprises an unsigned integer representing the frame's bit rate in Hz
codr	Yes, if modu=LORA	string	ECC coding rate "k/n" where k=carried bits and n=total number of bits received
rssi	Yes	signed float	RSSI in dBm, 1 dB precision
lsnr	Yes, if modu=LORA	signed float	Lora SNR ratio in dB, 0.1 dB precision
size	Yes	unsigned integer	RF packet payload size in bytes
data	Yes	string	Base64 encoded RF packet payload. The Base64 padding characters shall not be added.

STAT JSON OBJECT KEYS

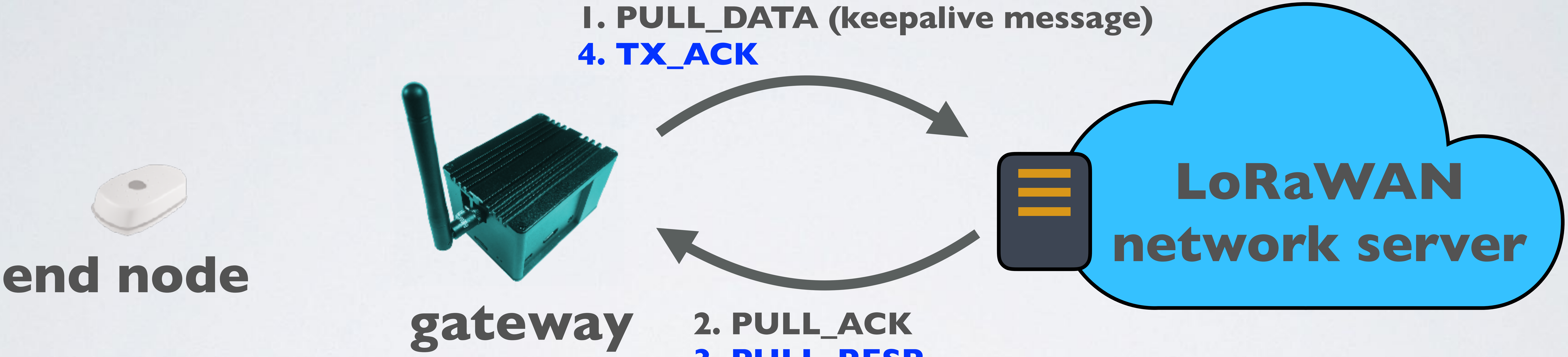
Name	Required	Type	Function
time	No	string	If GPS or fake GPS enabled: UTC system time of the gateway, one second precision. ISO 8601 'expanded' format.
lati	No	float, max 5 decimals	If GPS or fake GPS enabled: GPS latitude of the gateway in degrees (N is +)
long	No	float, max 5 decimals	If GPS or fake GPS enabled: GPS latitude of the gateway in degrees (E is +)
alti	No	signed integer	If GPS or fake GPS enabled: GPS altitude of the gateway in meters
rxnb	No	unsigned integer	Number of radio packets received since gateway start
rxok	No	unsigned integer	Number of radio packets received with correct CRC since gateway start
rxfw	No	unsigned integer	Number of radio packets forwarded to the server since gateway start
ackr	No	signed float	Percentage of radio packets that were forwarded and acknowledged by the server since gateway start
dwnb	No	unsigned integer	Number of radio packets received from the server since gateway start
txnb	No	unsigned integer	Number of radio packets transmitted since gateway start

RXPK AND STAT JSON OBJECT EXAMPLE



```
{
"rxpk":[{
    "time":"2013-03-31T16:21:17.528002Z",
    "tmst":3512348611,
    "chan":2,
    "rfch":0,
    "freq":866.349812,
    "stat":1,
    "modu":"LORA",
    "datr":"SF7BW125",
    "codr":"4/6",
    "rssi":-35,
    "lsnr":5.1,
    "size":32,
    "data":"-DS4CGaDCdG+48eJNM3Vai-zDpsR71Pn9CPA9uC0N84"
}],
"stat":{
    "time":"2014-01-12 08:59:28 GMT",
    "lati":46.24000,
    "long":3.25230,
    "alti":145,
    "rxnb":2,
    "rxok":2,
    "rxfw":2,
    "ackr":100.0,
    "dwnb":2,
    "txnb":2
}
}
```

DOWNSTREAM COMMUNICATION



DOWNSTREAM COMMUNICATION

- (1) At regular time intervals the gateway sends a PULL_DATA packet (aka keepalive message) to the network server. If the gateway is behind a firewall it impossible for the network server to send packets to the gateway. The PULL_DATA packets keeps any intervening firewall open by informing the server of the gateway UDP port number which it can use.

- The time interval is set by the **keepalive_interval** key in the global_conf.json or local_conf.json file.

- (2) After the server received the PULL_DATA packet, the server sends a PULL_ACK back to the gateway to confirm that the network route is open and that the server can send PULL_RESP packets at any time to the gateway.

DOWNSTREAM COMMUNICATION

- (3) When the gateway receives a PULL_RESP packet, (4) the gateway sends a TX_ACK feedback to the server to inform if the downlink request has been accepted or rejected by the gateway.
- The TX_ACK feedback can contain a JSON object to give more details on success or failure. If no JSON is present (empty string), this means no error occurred.

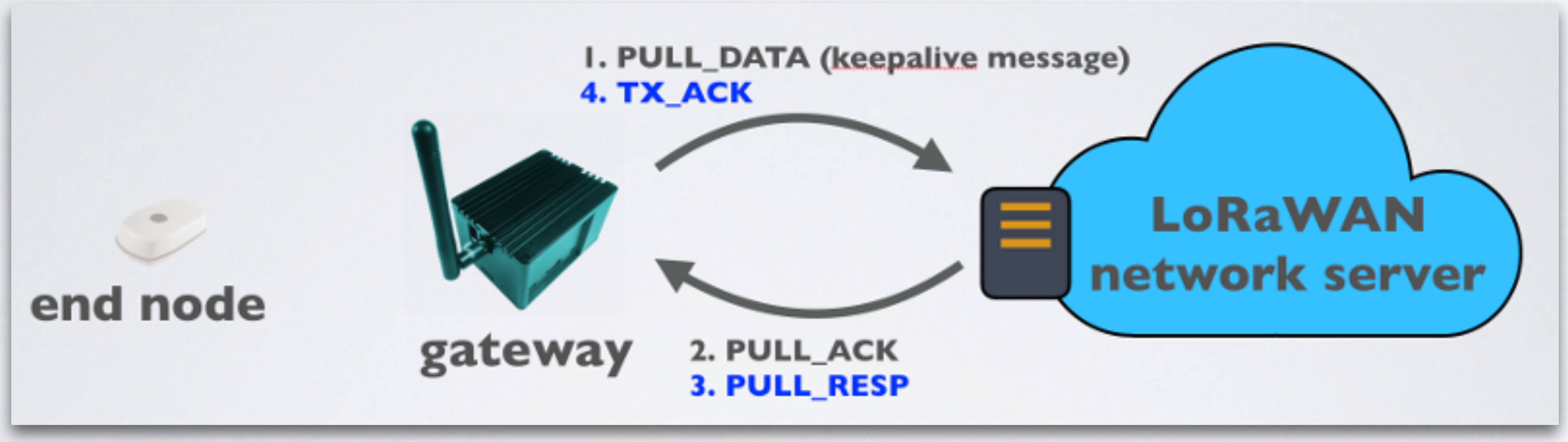
PULL_DATA AND PULL_ACK MESSAGE FORMAT

Byte 0	1-2	3	4-11
protocol version 0x02	Random token	PULL DATA 0x02	Gateway EUI

PULL_DATA message format
Packet size = 12 bytes

Byte 0	1-2	3
protocol version 0x02	Token from PULL_DATA	PULL ACK 0x04

PULL_ACK message format
Packet size = 4 bytes



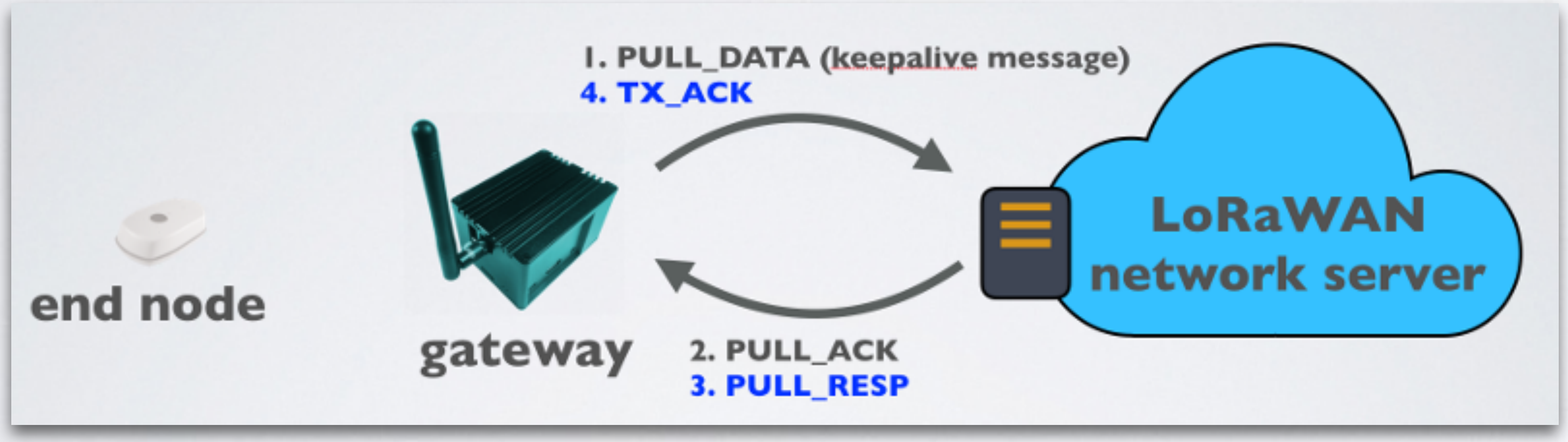
PULL_RESP AND TX_ACK MESSAGE FORMAT

Byte 0	1-2	3	4-end
protocol version 0x02	Random token	PULL RESP 0x03	JSON object txpk

PULL_RESP message format
Max packet size = 1000 bytes

Byte 0	1-2	3	4-11	12-end
protocol version 0x02	Token from PULL_RESP	TX ACK 0x05	Gateway EUI	Optional JSON object txpk_ack

TX_ACK message format
Packet size = 12 - 41 bytes



PULL_RESP JSON OBJECT

- The PULL_RESP JSON object contains an object called **txpk** (transmit packet) which contains a RF packet to be emitted and associated metadata.

```
PULL_RESP JSON object:
{
  "txpk":{...}
}
```

TX_ACK JSON OBJECT

- The TX_ACK JSON object contains an object called **txpk_ack** (transmit packet acknowledge) which contains status information concerning the associated PULL_RESP packet.

```
TX_ACK JSON object:
{
  "txpk_ack":{...}
}
```

- If no error is reported, the 'Payload' field comprises one byte of value '\0'.
  If an error is reported, the field contains a JSON "error" object.

TXPK JSON OBJECT KEYS

Name	Required	Type	Function
imme	No	bool	If true, the gateway is commanded to transmit the frame immediately (will ignore tmst & time)
tmst	No	unsigned integer < 2^{32}	If "imme" is not true and "tmst" is present, the gateway is commanded to transmit the frame when its internal timestamp counter equals the value of "tmst" (will ignore time)
tmms	Yes	string	UTC time, one microsecond precision. ISO 8601 ‘compact’ format. If "imme" is false or not present and "tmst" is not present, the gateway is commanded to transmit the frame at GPS time (GPS synchronization required).
freq	Yes	unsigned float	Transmitted signal centre frequency in MHz
rfch	Yes	unsigned integer	Concentrator "RF chain" used for TX (radio 0 or 1)
powe	No	signed integer	TX output power in dBm
modu	Yes	string	Modulation identifier "LORA" or "FSK"
datr	Yes	string / unsigned integer	Datarate identifier. If modu=LORA, datr comprises a string "SFnBWm", where n=spreading factor and m=bandwidth in kHz If modu=FSK, datr comprises an unsigned integer representing the frame's bit rate in Hz

TXPK JSON OBJECT KEYS

Name	Required	Type	Function
codr	Yes, if modu=LORA	string	ECC coding rate "k/n" where k=carried bits and n=total number of bits received, including those used by the error checking/ correction algorithm.
fdev	Yes	unsigned integer	FSK frequency deviation in Hz
ipol	No	bool	If true, gateway inverts the polarity of the transmitted bits. Server sets value to true when modu=LORA, otherwise the value is omitted.
prea	No	unsigned integer	RF preamble size
size	Yes	unsigned integer	RF packet payload size in bytes
data	Yes	string	Base64 encoded RF packet payload. Base64 padding characters shall not be not added.
ncrc	No	bool	If true, disable physical layer CRC generation by the transmitter

TXPK_ACK JSON OBJECT KEYS

Name	Type	Function
error	string	Create a JSON string if there is an error

Error Value	Definition
TOO_LATE	Rejected because it was already too late to program this packet for downlink
TOO_EARLY	Rejected because downlink packet timestamp was received by the gateway too long before the scheduled transmission time
COLLISION_PACKET	Rejected because there was already a packet programmed in requested timeframe
COLLISION_BEACON	Rejected because there was already a beacon planned in requested timeframe
TX_FREQ	Rejected because requested frequency is not supported by TX RF chain
TX_POWER	Rejected because requested power is not supported by gateway
GPS_UNLOCKED	Rejected because GPS is unlocked, so GPS timestamp cannot be used
UNKNONW	If the error is of unknown origin

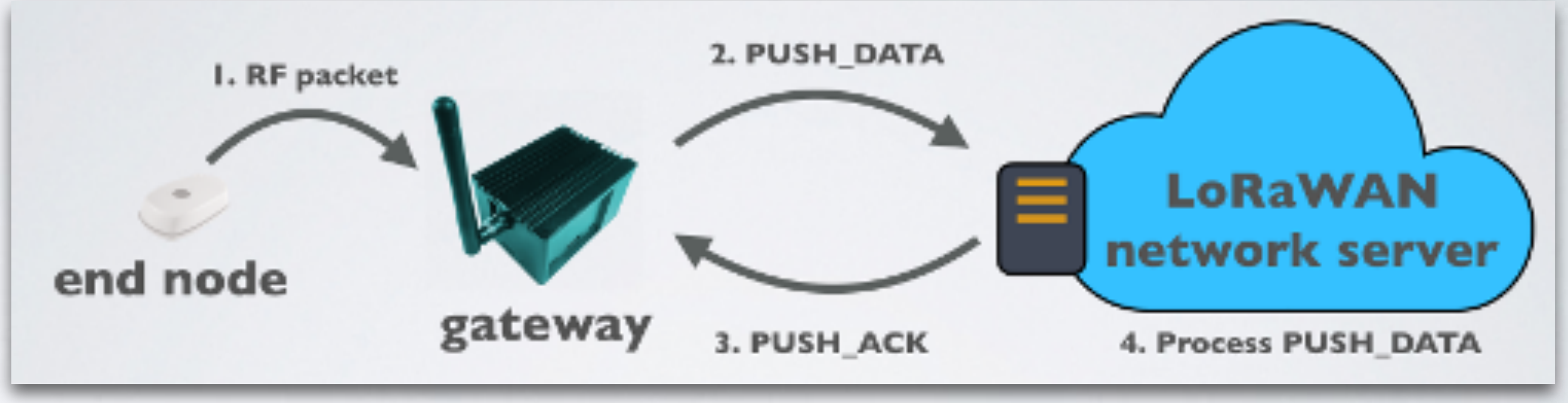
TXPK AND TXPK_ACK JSON OBJECT EXAMPLE

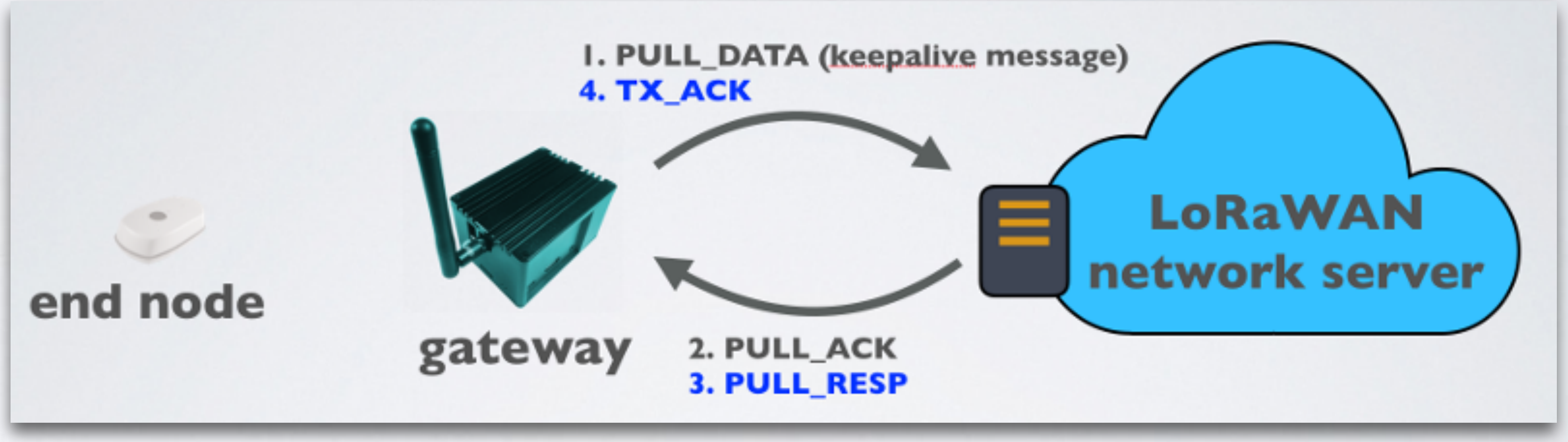
```
{"txpk":{
    "imme":true,
    "freq":864.123456,
    "rfch":0,
    "powe":14,
    "modu":"LORA",
    "datr":"SF11BW125",
    "codr":"4/6",
    "ipol":false,
    "size":32,
    "data":"H3P3N2i9qc4yt7rK7ldqoeCVJGBybzPY5h1Dd7P7p8v"
}}
```

```
{"txpk_ack":{
    "error":"COLLISION_PACKET"
}}
```

UDP PORTS

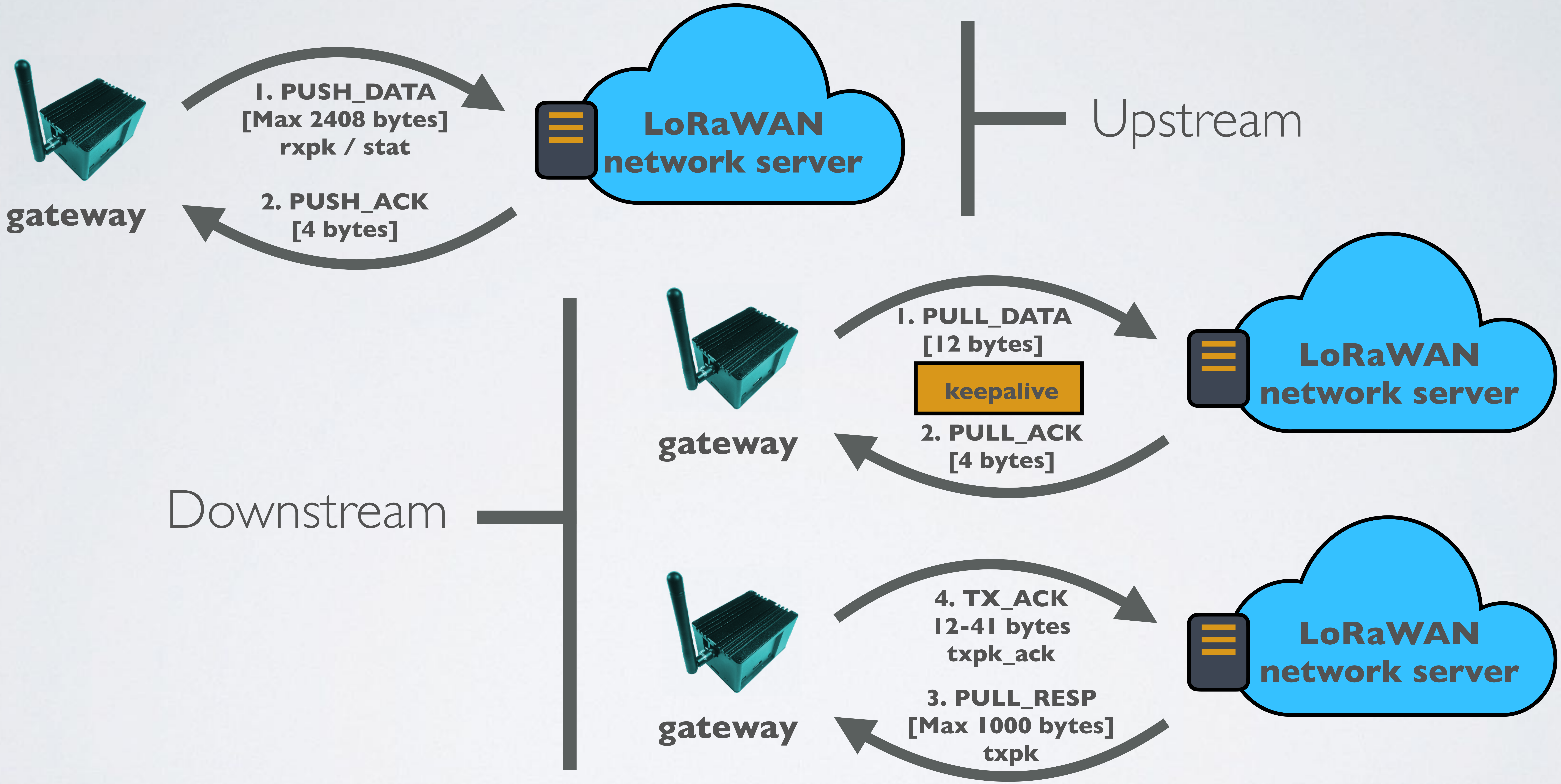
Type	From [UDP Port]	To [UDP Port]
PUSH_DATA	Gateway [~port A]	Server [1700]
PUSH_ACK	Server [1700]	Gateway [~port A]
PULL_DATA	Gateway [~port B]	Server [1700]
PULL_ACK	Server [1700]	Gateway [~port B]
PULL_RESP	Server [1700]	Gateway [~port of the most recent PULL_DATA message]
TX_ACK	Gateway [~port of the most recent PULL_DATA message]	Server [1700]





1) ~port means arbitrary port.
2) TTN server uses port 1700 for uplinks and downlinks.
   See: global_conf.json or local_conf.json

SEMTECH UDP PROTOCOL OVERVIEW



Upstream

gateway

1. PUSH_DATA
[Max 2408 bytes]
rxpk / stat

2. PUSH_ACK
[4 bytes]

LoRaWAN network server

Downstream

gateway

1. PULL_DATA
[12 bytes]

keepalive

2. PULL_ACK
[4 bytes]

LoRaWAN network server

gateway

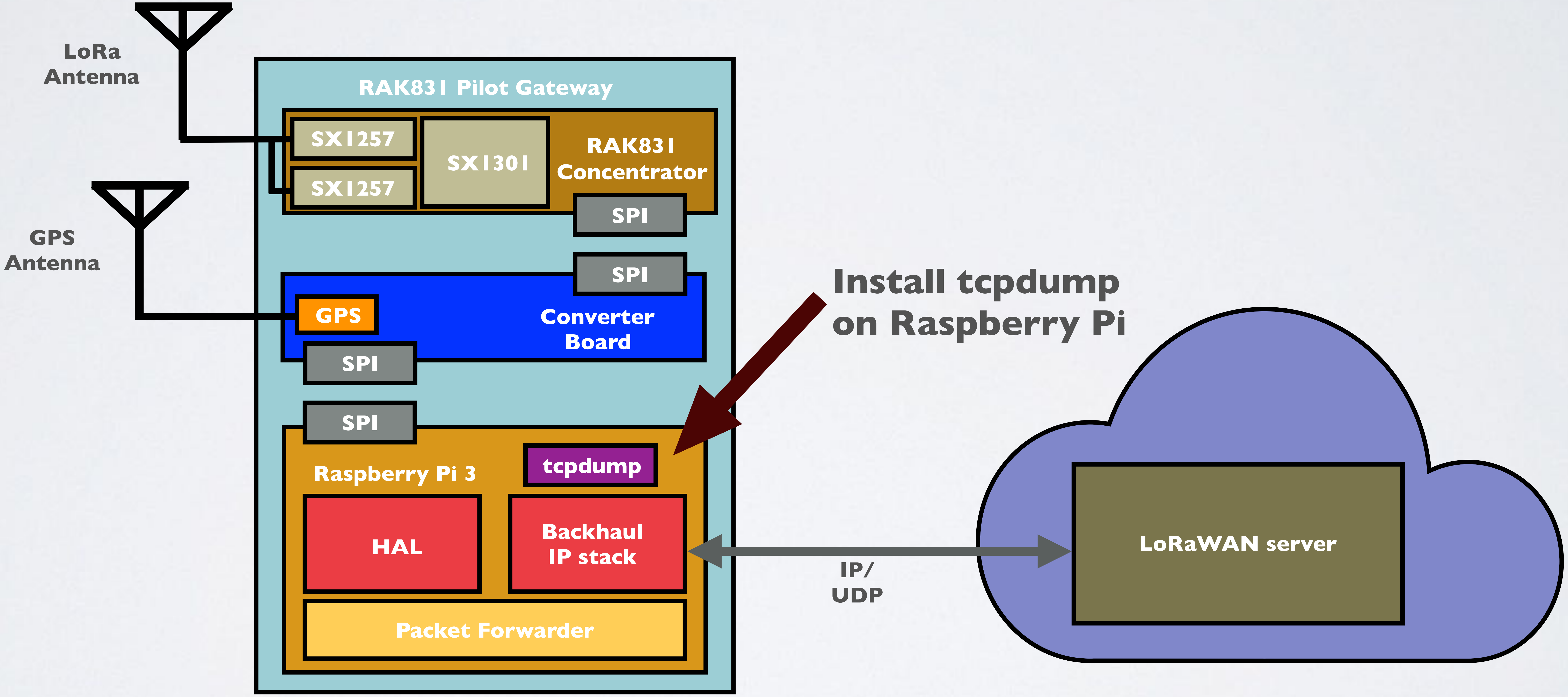
4. TX_ACK
12-41 bytes
txpk_ack

3. PULL_RESP
[Max 1000 bytes]
txpk

LoRaWAN network server

TCPDUMP

- Tcpdump is a command line packet analyser that monitors and logs TCP/IP traffic and other packets passing between a network and the computer on which it is executed.



Install tcpdump on Raspberry Pi

TCPDUMP

- To install tcpdump:
  - Upgrade the Raspberry Pi packages:

    ```
    sudo apt-get update && sudo apt-get upgrade -y
    ```

  - Install tcpdump:

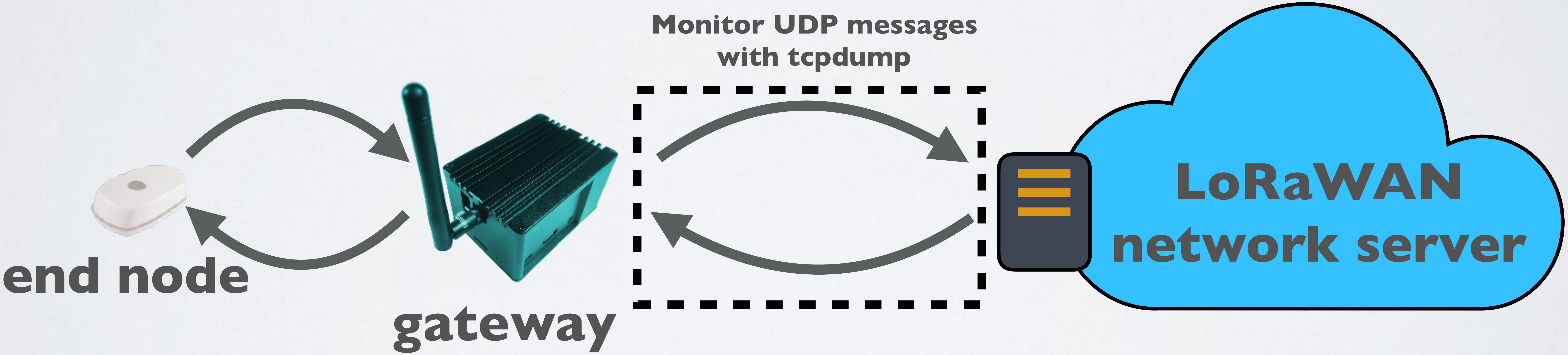
    ```
    sudo apt-get install tcpdump -y
    ```

TCPDUMP

- An end node sends sensor data to TTN via the RAK831 Pilot Gateway. The end node also receives data from TTN to switch LEDs on/off.

- Monitor UDP messages between the RAK831 Pilot Gateway and TTN server:

```
sudo tcpdump -XUq port 1700  (ASCII and Hex)
sudo tcpdump -AUq port 1700  (only ASCII)
```



TCPDUMP

- Show UDP messages on the console and also write to a file:

```
sudo tcpdump -XUq port 1700 | tee tcpdump_output.txt
```

- The generated raw tcpdump output:
  https://www.mobilefish.com/download/lora/tcpdump_output.txt

- Some notes added for more detailed explanation:
  https://www.mobilefish.com/download/lora/tcpdump_output_with_notes.txt

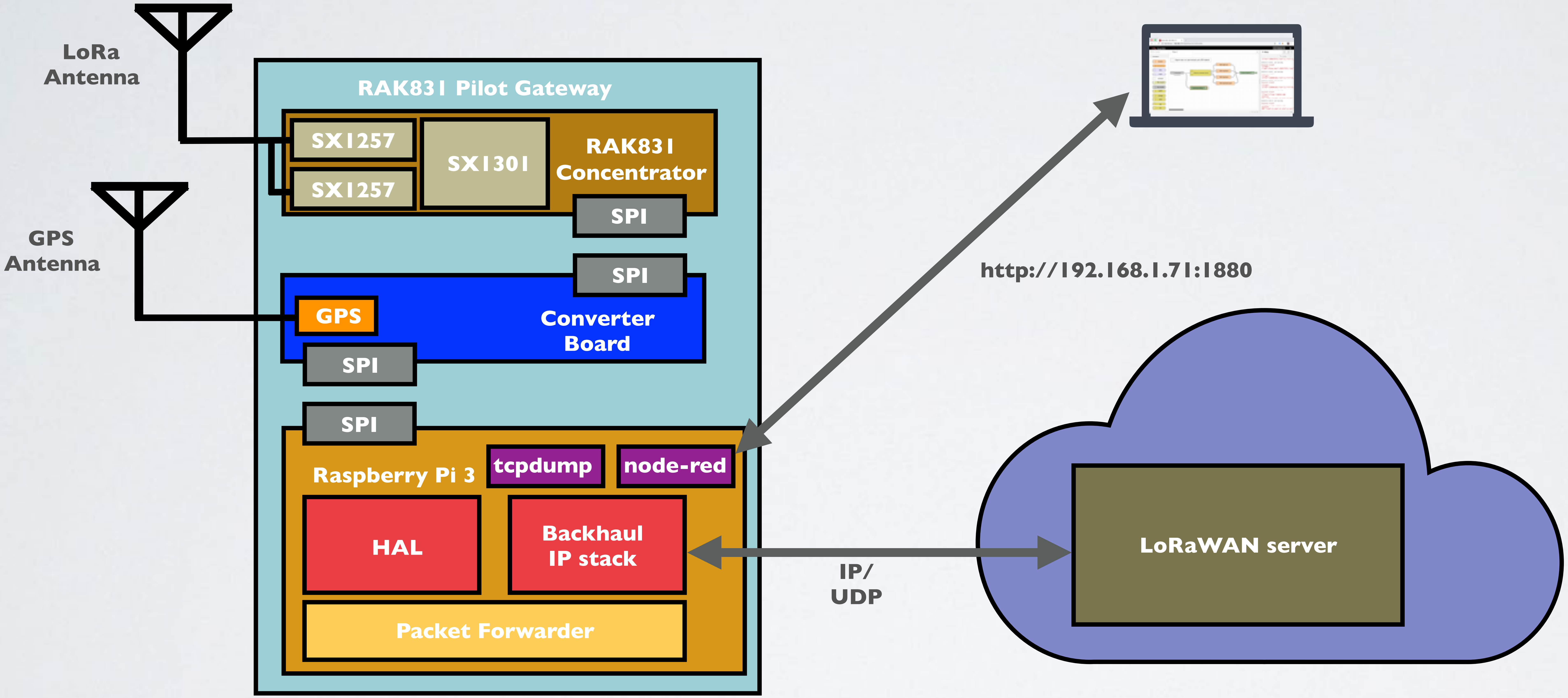
NODE-RED

- Node-RED is a browser-based development tool for wiring together hardware devices, APIs and online services.

- Node-RED can be used to monitor the traffic between the RAK831 Pilot Gateway and TTN server.

- In this tutorial Node-Red will be installed on the gateway itself.

NODE-RED


LoRa Antenna
GPS Antenna
RAK831 Pilot Gateway
SX1257
SX1257
SX1301
RAK831 Concentrator
SPI
SPI
GPS
Converter Board
SPI
SPI
Raspberry Pi 3
tcpdump
node-red
HAL
Backhaul IP stack
Packet Forwarder
http://192.168.1.71:1880
IP/ UDP
LoRaWAN server

NODE-RED

- Install Node-Red:
  - Goto pi's home directory:

    ```
    cd ~
    ```

  - Lets install Node-Red in User directory /home/pi/.node-red:

    ```
    bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
    ```

  - Answer the questions:
    Are you really sure you want to do this? **y**
    Would you like to install the Pi-specific nodes? **y**

NODE-RED

- Use Node-Red:
  - Goto pi’s home directory:

    ```
    cd ~
    ```

  - Start Node-Red:

    ```
    node-red-start
    ```

  - View the recent Node-Red logs:

    ```
    node-red-log
    ```

  - Stop Node-Red:

    ```
    CTRL+C
    ```

    (Node-Red is still running in the background)

    ```
    node-red-stop
    ```

NODE-RED

- To make the Node-Red flow work, a command need to be executed:

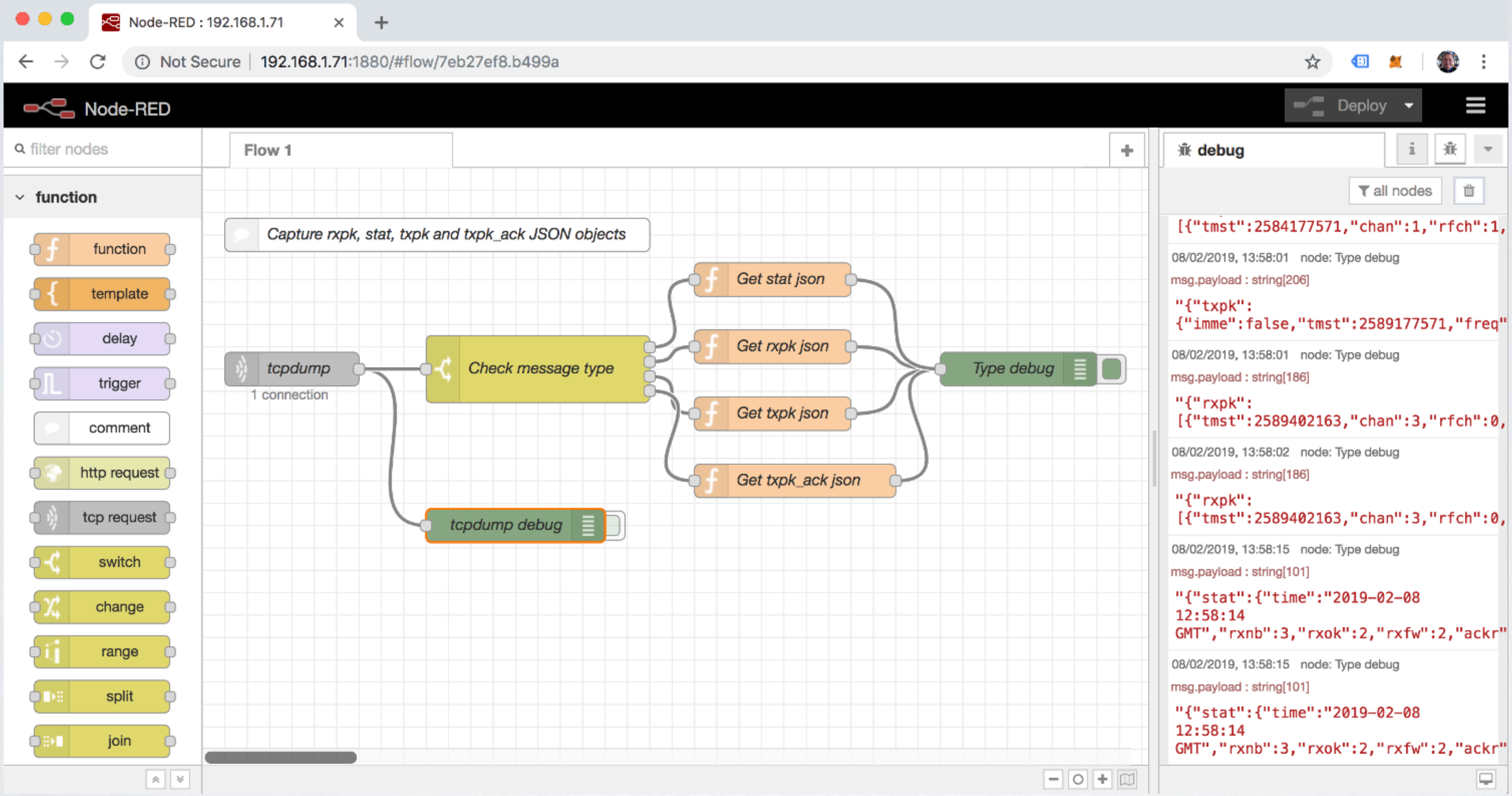
```
sudo tcpdump -Alqn port 1700 | nc localhost 8888 &
```

- To stop the running background process:

`jobs`	(show list of background jobs)
`fg <number>`	(Eg: **`fg 2`**, bring job 2 to foreground)
`CTRL+C`	(Stop the job)

- Import a very simple Node-Red flow to capture the rxpk (received packet), stat (status), txpk (transaction packet) and txpk_ack (transaction packet acknowledge) JSON objects which are sent to/from the gateway:
  https://www.mobilefish.com/download/lora/capture_gateway_lorawan_network_server_packets.json

NODE-RED



Node-RED : 192.168.1.71

Not Secure | 192.168.1.71:1880/#flow/7eb27ef8.b499a

Capture rxpk, stat, txpk and txpk_ack JSON objects

```
[{"tmst":2584177571,"chan":1,"rfch":1,
08/02/2019, 13:58:01   node: Type debug
msg.payload : string[206]
"{"txpk":
{"imme":false,"tmst":2589177571,"freq"
08/02/2019, 13:58:01   node: Type debug
msg.payload : string[186]
"{"rxpk":
[{"tmst":2589402163,"chan":3,"rfch":0,
08/02/2019, 13:58:02   node: Type debug
msg.payload : string[186]
"{"rxpk":
[{"tmst":2589402163,"chan":3,"rfch":0,
08/02/2019, 13:58:15   node: Type debug
msg.payload : string[101]
"{"stat":{"time":"2019-02-08
12:58:14
GMT","rxnb":3,"rxok":2,"rxfw":2,"ackr"
08/02/2019, 13:58:15   node: Type debug
msg.payload : string[101]
"{"stat":{"time":"2019-02-08
12:58:14
GMT","rxnb":3,"rxok":2,"rxfw":2,"ackr"
```