



**POLITECNICO**  
**MILANO 1863**

Internet of things

**IoT Challenge #3**  
**LoRaWAN**  
**Exercise PDF**

**Authors:**

Daniel Shala - 10710181

Jurij Diego Scandola - 10709931

**Academic Year:**

2024 - 2025

# 1 LoRaWAN network - Problem description

A LoRaWAN network in Europe (carrier frequency 868 MHz, bandwidth 125 kHz) is composed by one gateway and 50 sensor nodes. The sensor nodes transmit packet with payload size of L byte according to a Poisson process with intensity  $\lambda = 1$  packet / minute.

## 1.1 Find the biggest LoRa SF for having a success rate of at least 70%.

"For the payload size L of your packet, take it as follows: Take XY = Last two digits of your person code (leader code)  $L = 3 + XY$  bytes e.g. person code = 10692911  $\rightarrow XY = 11 \rightarrow L = 3 + 11 = 14$ "

So we started by calculating the payload size, having the Leader code equals to 10710181, our Payload is  $L = 3 + 81$  bytes = 84 Bytes

The probability  $P_s$  for a packet transmission to be successful is the probability that no other packet starts transmission in "conflict" period of  $2T$ .

$$P_s = e^{-2G}$$

In the previous formula, G is the total fraction of time that the channel is occupied (or "offered") by transmissions from all devices — whether successful or not - distributed according to Poisson Process.

G is defined as:

$$G = \frac{N \cdot T_{tx}}{T_{frame}}$$

where:

- $N$  is the number of nodes,
- $T_{tx}$  is the airtime of a single packet,
- $T_{frame}$  is the average interval between transmissions.

Alternatively, in our case where the transmissions follow a Poisson process with rate  $\lambda$  (1 packet / minute):

$$G = N \times T_{tx} \times \lambda$$

By using the suggested website ([https : //www.thethingsnetwork.org/airtime – calculator](https://www.thethingsnetwork.org/airtime-calculator)) to calculate the total airtime, we made a table with the different SF utilized and their results:

Spreading Factor (SF)	Airtime (s)	$G = \frac{50 \cdot T_{tx}}{60}$	Success Rate $P_s = e^{-2G}$ or $e^{-2N\lambda t}$
7	0.1692	0.141	0.754
8	0.2975	0.248	0.610
9	0.5335	0.445	0.406

Table 1: Success rate for different LoRa SFs with  $N = 50$  nodes,  $\lambda = \frac{1}{60}$  packets/sec

## 1.2 Conclusions


From the analysis of the data and the required specifications, it is clear that *SF7* is the only option that meets the required 70% success rate (75.4%), with our payload of 84 bytes. As indicated on the [www.thethingsnetwork.org](http://www.thethingsnetwork.org) website, models starting from *SF10* are not suitable, as they do not meet the necessary specifications for our use case. Below, we include a screenshot from the website to confirm this information.

Therefore, the best (and only) solution is to use *SF7* to ensure proper system functionality and compatibility with the required specifications.

## 1.3 Related images

The screenshot shows the 'THE THINGS NETWORK' calculator interface. At the top, there are navigation links: Learn, Hardware, Forum, Communities, Conference, and Enterprise (highlighted in blue). Below these, there are four input fields with dropdown menus: 'Input Bytes' (84), 'Spreading Factor' (SF7), 'Region' (EU868), and 'Bandwidth' (125 kHz). Each field has a small blue circle with a question mark icon. Below the input fields, the word 'Result' is displayed in large blue text. To the right of 'Result', there is a large blue box containing the text '169.2 ms'. Below this box, there is a smaller white box with the text 'Time on air'.

Figure 1: Example of SF7 with airtime.



[Learn](#)[Hardware](#)[Forum](#)[Communities](#)[Conference](#)[Enterprise](#)

Input Bytes

84

Spreading Factor

SF10

Region

EU868

Bandwidth

125 kHz

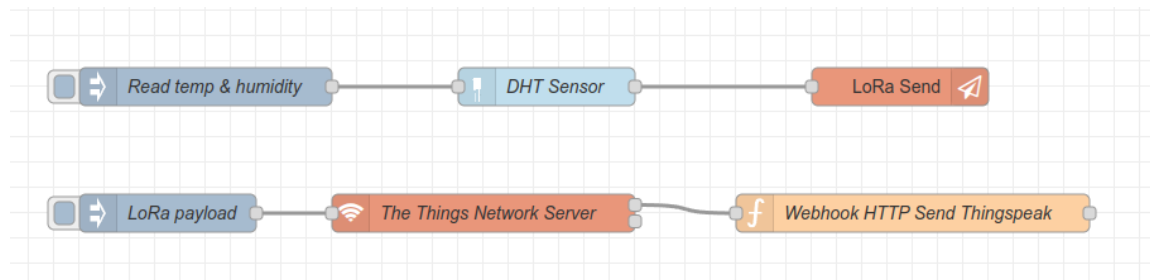
Max payload size exceeded. Please check the tables below.

Figure 2: Max payload exceeded for SF10 and higher.

## 2 Design network

You have purchased an Arduino MKR WAN 1310 and wish to create a system that reads temperature and humidity data from a DHT22 sensor and sends this data wirelessly to ThingSpeak over LoRaWAN.

### 2.1 Design a complete system block diagram (sketch in Node-Red) and describe, in detail, the steps you would need to take to get the system fully operational.



To get the system fully operational, the first step is to set up a LoRaWAN server through The Things Network (TTN) and configure the necessary settings for the Arduino controller. This involves registering the device, ensuring proper data encoding and transmission over LoRaWAN, and establishing communication between the device and the TTN server.

Next, we will need to create two distinct ThingSpeak channels: one dedicated to tracking temperature data and the other for humidity. We will generate and obtain the required API key, which will be used for sending updates to the channels.

Finally, we will configure a webhook in TTN that triggers upon receiving a payload from the LoRaWAN device. This webhook will forward the received data to the corresponding ThingSpeak channels via HTTP, ensuring that both temperature and humidity readings are updated in real time. The webhook will act as the bridge between the LoRaWAN network and ThingSpeak, automating the entire data flow process.

### 3 Reproduce figures from paper

#### 3.1 Using the paper “Do LoRa Low-Power Wide-Area Networks Scale?” by M. Bor et al. and the LoRa simulator available at LoRaSim, your task is to reproduce Figure 5 and Figure 7 from the paper

##### 3.1.1 simulate

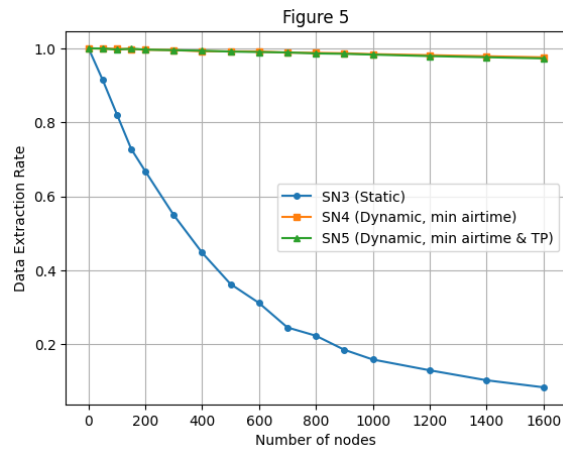
**Purpose:** Executes a single-sink LoRa simulation using the script `lorasim/loradir.py`.

**Parameters:**

- `n_nodes`: Number of sensor nodes
- `tx_rate`: Transmission rate
- `exp`: Experiment identifier
- `duration`: Simulation time

To reproduce Fig. 5 from the paper, we configure the experiment using the values 3 ( $SN^4$ ), 4 ( $SN^3$ ), and 5 ( $SN^5$ ). Additionally, to align the results more closely with those presented in the original figure, we enable the collision flag by setting it to 1 prior to executing the simulation script.

```
1 duration = 86400000
2 tx_rate = 1e6
3
4 for n_nodes in list(range(1,200,50)) + list(range(200,1000,100)) + list(
5     range(1000,1800,200)):
6     print(f"Simulating {n_nodes} nodes")
7     simulate(n_nodes, tx_rate, 3, duration)
8     simulate(n_nodes, tx_rate, 4, duration)
9     simulate(n_nodes, tx_rate, 5, duration)
```



### 3.1.2 simulate\_multi\_sink

**Purpose:** Executes a LoRa network simulation with multiple sinks (base stations) by running the Python 2 script `lorasim/loradirMulBS.py`.

**Parameters:**

- `n_nodes`: Number of sensor nodes
- `tx_rate`: Transmission rate
- `exp`: Experiment identifier
- `duration`: Duration of the simulation
- `n_sinks`: Number of base stations

To reproduce Fig. 7 from the paper, we configure the experiment with the value 0 ( $SN^1$ ) and vary the number of sinks across 1, 2, 3, 4, 8, and 24. As in the previous case, we enable the collision flag by setting it to 1 before executing the simulation script.

```
1 duration = 86400000
2 tx_rate = 1e6
3
4 for n_nodes in list(range(1,200,50)) + list(range(200,1000,100)) + list(
5     range(1000,1800,200)):
6     print(f"Simulating {n_nodes} nodes")
7     simulate_multi_sink(n_nodes, tx_rate, 0, duration, 1)
8     simulate_multi_sink(n_nodes, tx_rate, 0, duration, 2)
9     simulate_multi_sink(n_nodes, tx_rate, 0, duration, 3)
10    simulate_multi_sink(n_nodes, tx_rate, 0, duration, 4)
11    simulate_multi_sink(n_nodes, tx_rate, 0, duration, 8)
12    simulate_multi_sink(n_nodes, tx_rate, 0, duration, 24)
```

