IoT Challenge #3
LoRaWAN

Giuliano Crescimbeni, 10712403 - Arimondo Scrivano, 10712429
Politecnico di Milano

April 2025

Contents

1 Exercise EQ1

1.1 Problem Description

We are asked to determine the maximum LoRa spreading factor (SF) that guarantees a success rate greater than 70% for a network with:

- 50 nodes,
- Transmission rate of 1 packet per minute per node,
- Operating in the European 868 MHz band, 125 kHz bandwidth.

The payload size L was set following the instruction: $L = 3+03$ (10712403)

1.2 Methodology

We applied the ALOHA protocol formula for success rate:

$$SuccessRate = e^{-2N\lambda t}$$

where:

- N = number of nodes,
- λ = transmission rate (packets/minute),
- t = transmission time (minutes).

To calculate the transmission time t, we used the **airtime calculator** suggested in the assignment: `https://www.thethingsnetwork.org/airtime-calculator`.

We tested different spreading factors (SF) by:

- Calculating the airtime for each SF,
- Substituting the corresponding airtime into the success rate formula,
- Selecting the smallest SF that guarantees a success rate above 70%.

1.3 Python Code

The following Python script was used to compute the success rates:

```python
import numpy as np

def success_rate(N, lambda_per_minute, t_minutes):
    exponent = -2 * N * lambda_per_minute * t_minutes
    return np.exp(exponent)

N = 50
lambda_per_minute = 1

# SF9
t_minutes = 185.3 / 60000
rate = success_rate(N, lambda_per_minute, t_minutes)
print(f"Success Rate in case of SF9= 185.3 [ms] {rate:.6f} %")

# SF10
t_minutes = 329.7 / 60000
rate = success_rate(N, lambda_per_minute, t_minutes)
print(f"Success Rate in case of SF10= 329.7 [ms] {rate:.6f} %")
```

1.4 Results

- **SF9** (airtime: 185.3 ms): Success Rate $\approx 73.43\%$
- **SF10** (airtime: 329.7 ms): Success Rate $\approx 57.72\%$

Since SF9 already guarantees a success rate slightly above 70%, we selected **SF9** as the optimal spreading factor.

1.5 Conclusion

Using SF9 ensures a success rate greater than 70%, fulfilling the challenge requirements.

2 Exercise EQ2

2.1 Problem Description

We are asked to design a complete system to acquire temperature and humidity data from a DHT22 sensor, transmit it via LoRaWAN using an Arduino MKR WAN 1310, and visualize it on ThingSpeak.

2.2 System Description

We implemented a simple **Node-Red sketch flow** to represent the functioning of the system.

The overall process works as follows:

- The **DHT22 sensor** periodically samples temperature and humidity values.
- The sampled data is sent to the **Arduino MKR WAN 1310**, which formats the readings into a packet.
- The Arduino transmits the packet over **LoRaWAN** to a gateway connected to **The Things Network (TTN)**.
- The TTN platform decodes the received LoRaWAN message and forwards it via **HTTP integration** to the distribution system.
- The distribution system sends the processed data to **ThingSpeak** for visualization.

2.3 Node-Red Flow

In Node-Red, a simplified sketch of the system was created to illustrate the communication chain:

- Input node representing the sensor acquisition,
- Function node simulating data packet creation,
- HTTP Request node simulating the communication with ThingSpeak.

This sketch demonstrates the end-to-end data flow from the sensor to the cloud visualization platform.

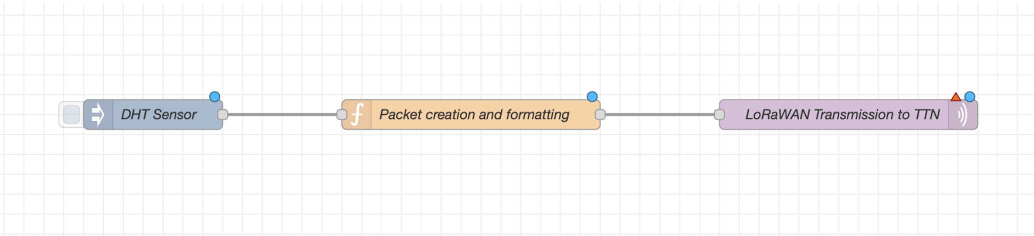

Figure 1: Node-Red Sketch: Data flow from DHT Sensor to TTN via LoRaWAN

2.4 Conclusion

The designed system ensures reliable acquisition and transmission of environmental data, utilizing LoRaWAN and cloud services for remote monitoring.

3 Exercise EQ3

3.1 Objective

The goal is to replicate the results shown in Figure 5 and in Figure 7 of the reference paper, using the LoRaSim simulator.

Specifically, we analyze how the dynamic allocation of communication parameters impacts network scalability in terms of Data Extraction Rate (DER).

3.2 Figure 5

3.2.1 Simulation Parameters

The following parameters were used for the simulation:

- **Communication parameters configurations**:
  - **SN3**: Static conservative configuration (typical LoRaWAN settings).
  - **SN4**: Dynamic setting minimizing airtime.
  - **SN5**: Dynamic setting minimizing airtime and transmission power.

3.2.2 Methodology

We used the provided `LoRaSim` Python implementation, adjusting the configuration to correspond to the dynamic parameter assignment described in the paper.

For SN3 we used experiment 4, for SN4 experiment 3 and for SN5 experiment 5

The DER was computed as the ratio of successfully received packets to the total number of transmitted packets.

3.3 Resulting Diagram

The resulting figure replicates the behavior shown in the reference paper's Figure 5, demonstrating that optimally choosing transmission parameters greatly increases the network capacity.

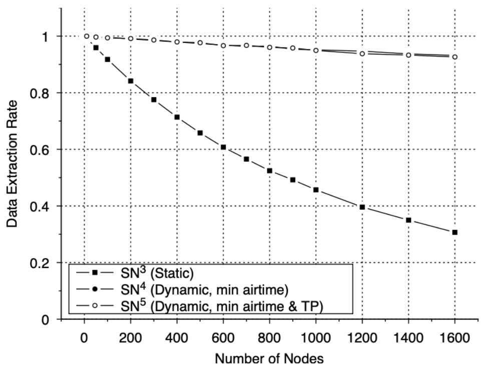

Figure 2: Original Figure 5 from LoRaSim paper

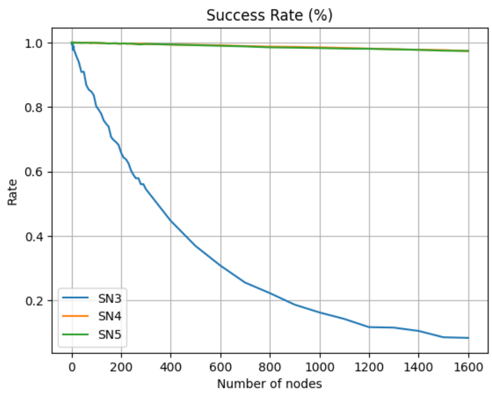

Figure 3: Replicated Figure 5 from LoRaSim simulation

The comparison shows that the replicated results closely match the original paper's findings. Dynamic parameter optimization significantly increases the supported number of nodes while maintaining a high DER.

3.4 Figure 7

3.4.1 Simulation Parameters

The following parameters were used for the simulation:

- **Communication parameters configuration**:
  - **Experiment 0**: slowest data rate (SF12, BW125, CR4/8).
- **Number of sinks**: variable (1, 2, 3, 4, 8, 24).

3.4.2 Methodology

We used the provided `LoRaSim` Python implementation with **experiment 0** settings, corresponding to the slowest data rate configuration (SF12, BW125, CR4/8).

The DER was provided directly by the simulator output, without the need for additional computation. The headers of the output files were manually corrected due to a typo in the simulator's output generator.

3.5 Resulting Diagram

The resulting figure replicates the behavior shown in the reference paper's Figure 7, demonstrating how increasing the number of sinks improves network performance under conservative communication settings.

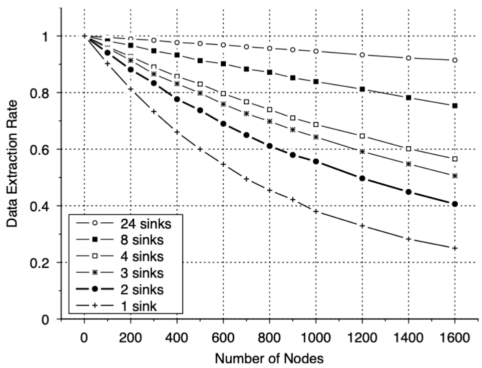

Figure 4: Original Figure 7 from LoRaSim paper

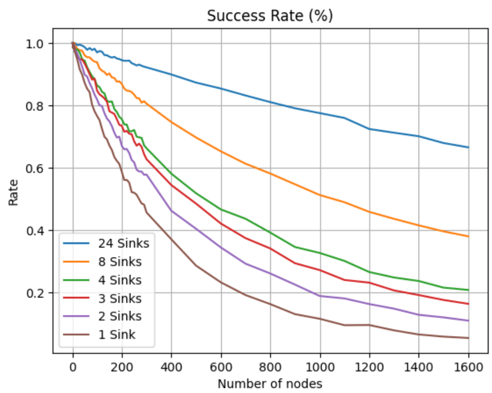

Figure 5: Replicated Figure 7 from LoRaSim simulation

The comparison shows that the replicated results closely match the original paper's findings. Deploying multiple sinks drastically improves the DER, even under very low data rate settings.