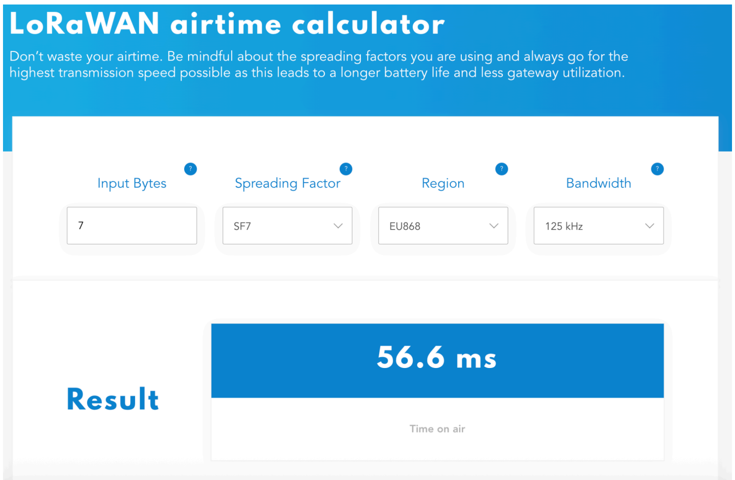
Third Challenge - Internet Of Things

Professor: Redondi Alessandro - Academic Year: 2024/2025

Pianalto Riccardo (Person Code: 10835980)
Pica Mirko (Person Code: 10811404)

EQ1

We first used the LoRaWAN airtime calculator to find the correspondent airtime for all spreading factors.



LoRaWAN airtime calculator

Don't waste your airtime. Be mindful about the spreading factors you are using and always go for the highest transmission speed possible as this leads to a longer battery life and less gateway utilization.

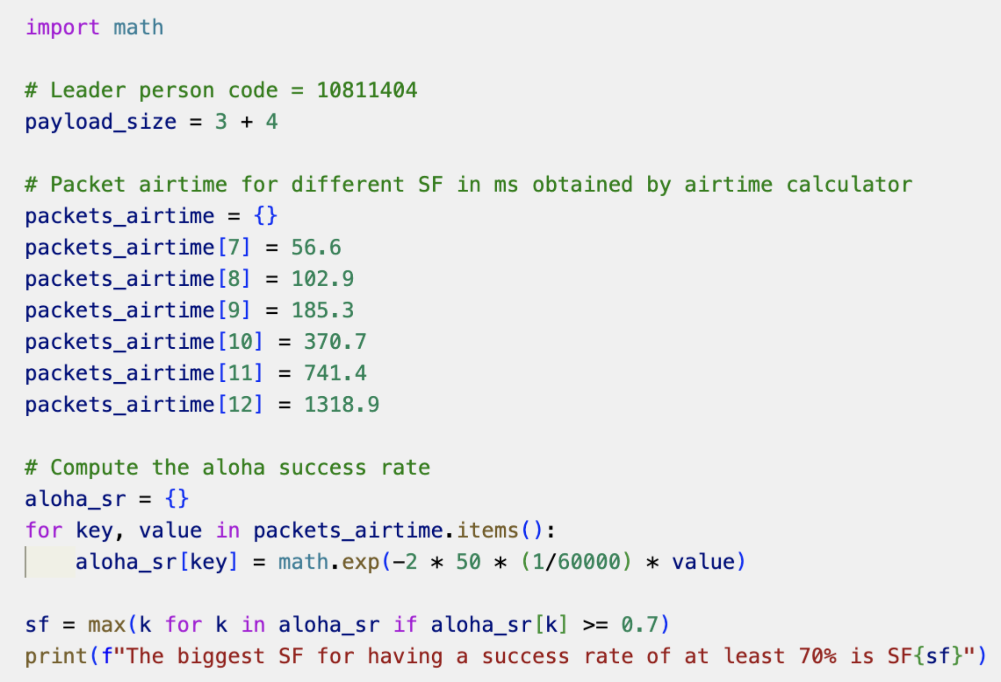
Input Bytes: 7
Spreading Factor: SF7
Region: EU868
Bandwidth: 125 kHz

Result

56.6 ms

Time on air

We know that In pure ALOHA, the success probability Ps for a packet is given by $Ps = e^{-2G}$ where G is the normalized traffic load, $G = N\times\lambda\times Tp$. We used a little python script to find the biggest SF such that $Ps \geq 70\%$.



```python
import math

# Leader person code = 10811404
payload_size = 3 + 4

# Packet airtime for different SF in ms obtained by airtime calculator
packets_airtime = {}
packets_airtime[7] = 56.6
packets_airtime[8] = 102.9
packets_airtime[9] = 185.3
packets_airtime[10] = 370.7
packets_airtime[11] = 741.4
packets_airtime[12] = 1318.9

# Compute the aloha success rate
aloha_sr = {}
for key, value in packets_airtime.items():
    aloha_sr[key] = math.exp(-2 * 50 * (1/60000) * value)

sf = max(k for k in aloha_sr if aloha_sr[k] >= 0.7)
print(f"The biggest SF for having a success rate of at least 70% is SF{sf}")
```

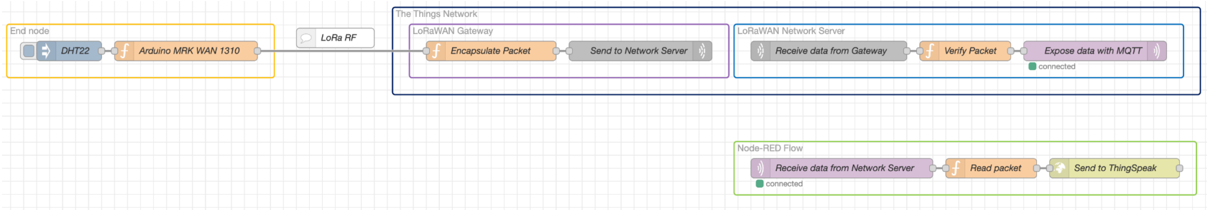
The biggest SF for having a success rate of at least 70% is SF9.

EQ2

We can create a system in which:

- Connect the DHT22 to the Arduino MKR WAN 1310;
- Configure Arduino MKR WAN 1310 with a arduino sketch that reads the data from DHT22, prepare the payload, connects to The Things Network (TTN) using OTAA (Over The Air Activation) and finally sends the data through LoRaWAN;
- Configure TTN creating an application on the portal, registering the device with DevEUI, AppEUI and AppKey. Set the correct decoding of the data.
- Create Node-RED flow that connects to TTN using MQTT, subscribes to the topic for uplink messages, makes the desired operations on the data and finally sends them with a HTTP POST Request to ThingSpeak.

We can emulate all this system architecture with Node-RED by replacing the arduino and the TTN respectively with a random number generator and an essential implementation of TTN components.



More in detail:

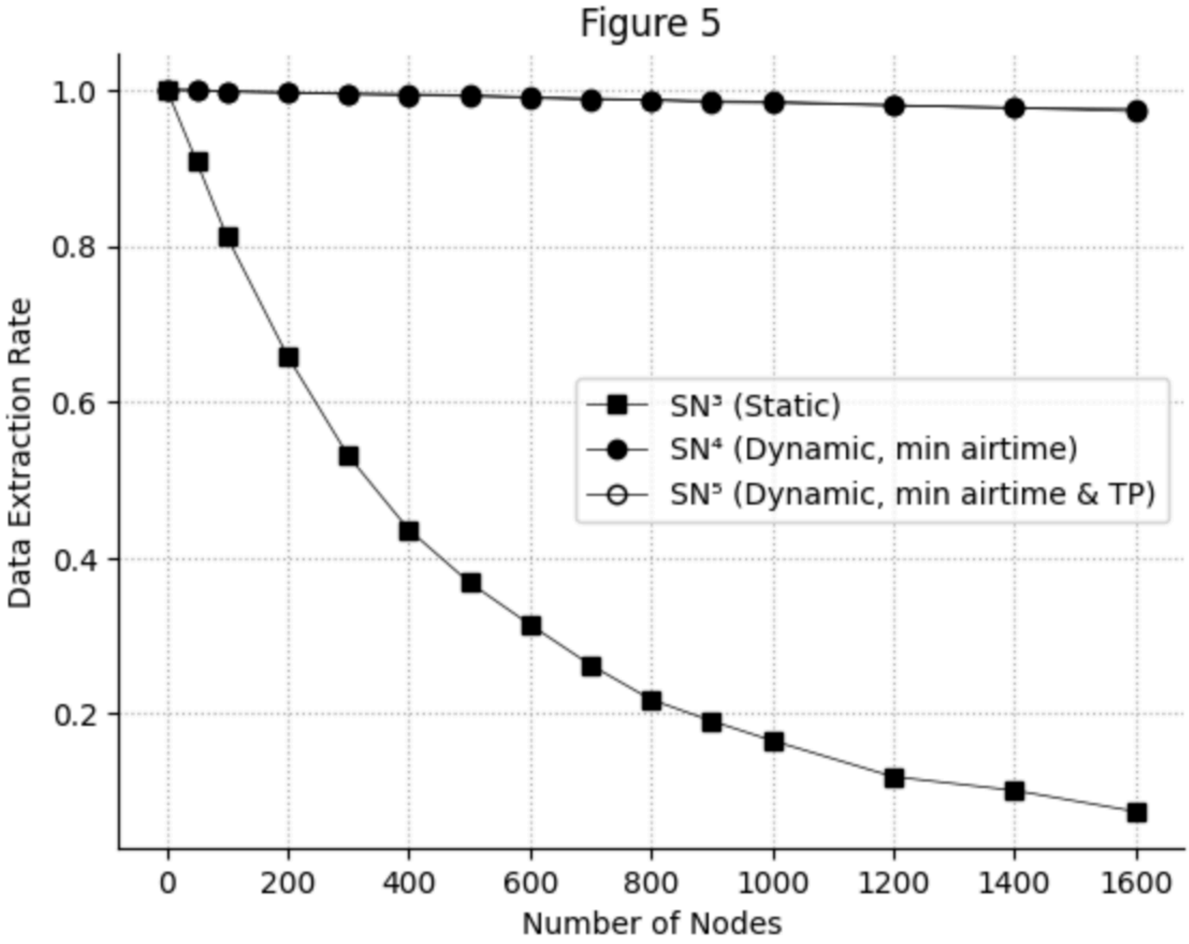
- **DHT22 Sensor:**
  - Measure temperature and humidity and it is wired with the Arduino MKR WAN 1310
- **Arduino MKR WAN 1310**:
  - Receive the data from the sensor
  - Prepare the data and encapsulate it in a LoRaWAN packet
  - Sends the packet via radio (LoRa frequencies) to the LoRaWAN Gateway
- **LoRaWAN Gateway**:
  - Receive the radio packet from the Arduino
  - Encapsulate the packet in in a UDP packet (without elaborate the data)
  - Forward the packet via Internet to the LoRaWAN Network Server
- **LoRaWAN Network Server**:
  - Receive the packet from the Gateway
  - Perform some network functions like security checks, de–duplication and packet decoding
  - Expose the data with MQTT
- **Node-RED Flow:**
  - Connect to TTN using MQTT
  - Receive the uplink data coming from Arduino
  - Perform some operations (like parsing)
  - Send the data to ThingSpeak with a HTTP POST Request
- **ThingSpeak:**
  - Receive the data (temperature and humidity)
  - Store them in a channel

- ○ Show them in graphs

EQ3

To reproduce figure 5 and figure 7 we needed to use LoRaSim with the right parameters. The lora simulator takes as parameters the number of nodes, the transmission rate, the experiment, the duration of the simulation and the collision mode.
From the paper we found that the transmission rate is 1 packet every 16.7 minutes (1000000ms), the duration of the simulation is 58 days (we used one day because of RAM constraints on Colab) and we have to use the full collision mode (setting collision = 1).
Graphs seem a bit off because the paper was made with a bugged version of the simulator which would give higher DER, but ours was made with the latest (look at the changelog for clarification).

At this point for figure 5 we simulated the experiments with an increasing number of nodes (we tried to copy the same step size as in the paper). Experiment specifications in the paper were indicated as SN3, SN4 and SN5, but reading the description of the experiment we noticed that the number wasn't matching the one in the simulator and in fact the experiment were respectively 4, 3 and 5.



For figure 7 we simulated experiment 0 (indicated as SN1 on the paper), with different number of base stations and increasing number of nodes to obtain the following figure.

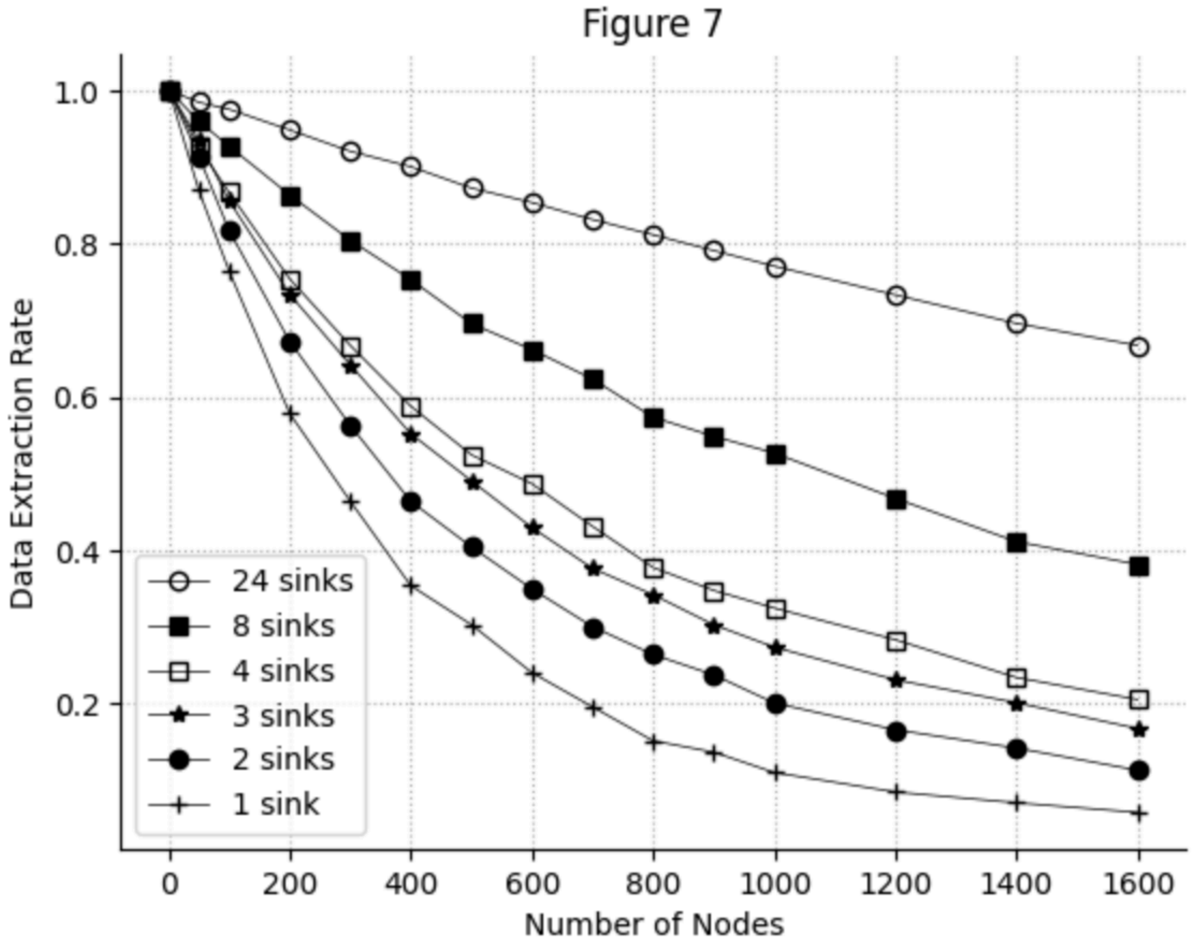


Figure 7
Data Extraction Rate
Number of Nodes
24 sinks
8 sinks
4 sinks
3 sinks
2 sinks
1 sink