Exercise

Chiara Thien Thao Nguyen Ba 10727985

April 27, 2025

1 EQ1

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. Find the biggest LoRa SF for having a success rate of at least 70%. Hint: use https://www.thethingsnetwork.org/airtime-calculator to compute the airtime of a packet.

The payload size L is computed as follows:

$$L = 3 + 85 = 88 \ bytes \tag{1}$$

Using the LoRaWAN Airtime Calculator, I found the airtime for an 88-byte payload at different SFs:

- SF7: 174.3 ms
- SF8: 307.7 ms
- SF9: 554 ms

Since each node transmits one packet per minute, there are overall 50 transmissions per minute. The probability of success is:

$$P = e^{-2 \cdot Tot\ Net\ Load \cdot Airtime}$$

The total network load is:

$$Tot\ Net\ Load = 50\ packets/60\ seconds = 0,833\ packets/second$$

The probability of success for each SF is:

- SF7: $P = e^{-2 \cdot 0,833 \cdot 0,1743} = 0,748$
- SF8: $P = e^{-2 \cdot 0,833 \cdot 0,3077} = 0,599$
- SF9: $P = e^{-2 \cdot 0,833 \cdot 0,554} = 0,397$

Thus, the biggest LoRa SF for having a success rate of at least 70% is SF7, with about 75% of success rate.

2 EQ2

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. 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.

The steps to make the system fully operational are:

- DHT22 sensor collects temperature and humidity readings.
- Arduino MKR WAN 1310 reads data from DHT22 at regular intervals, performs basic validation, formats data into compact binary payload and handles LoRaWAN communication protocol.
- Arduino transmits data using appropriate Spreading Factor and LoRaWAN Gateway receives the LoRaWAN signal and forwards to The Things Network (TTN) backend.
- The Things Network (TTN) processes and decodes the binary message.
- HTTP Webhook node listens for incoming messages from TTN and JSON Parser extracts the payload data from the HTTP message.
- ThingSpeak Node stores payload data in fields of a channel making an HTTP request, eventually saves the temperature and humidity measures in a csv file and plots graphs and charts about temperature and humidity values.

Figure 1 shows the block diagram in Node-Red that tries to sketch the system.

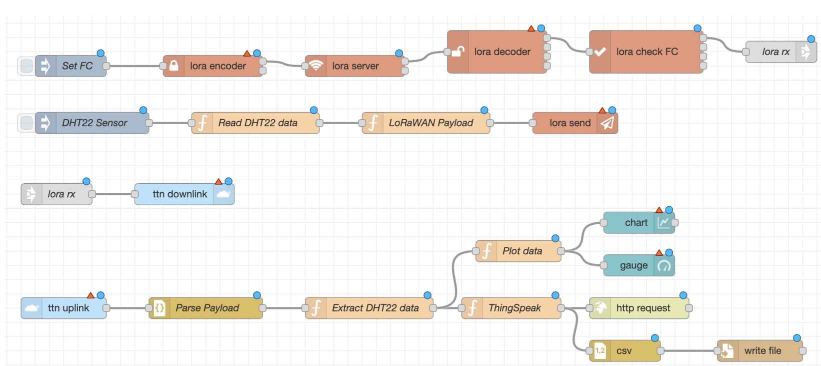

Figure 1: The system block diagram

In particular, the nodes are:

- LoRaWAN Communication
  - Inject - Set FC (Frame Counter): Initializes the frame counter before encoding for LoRaWAN transmission.
  - LoRaWAN Encoder: Encodes and encrypts the payload into a LoRaWAN-compliant packet.
  - LoRaWAN Server: Simulates a local LoRaWAN server to receive and forward LoRa packets.
  - LoRaWAN Decoder: Decodes the received encrypted LoRaWAN packets back into usable data.
  - LoRaWAN CheckFC: Validates the frame counter to ensure message sequence is correct.
  - Link out - LoRaWAN rx: Outputs the decoded LoRaWAN data to another flow part.
- Arduino MKR WAN 310 and DHT22 sensor
  - Inject - DHT22 Sensor: Simulates data acquisition from the DHT22 sensor (temperature and humidity).
  - Function - Read DHT22 data: Simulates the Arduino reading of the DHT22 sensor values.
  - Function - LoRaWAN Payload: Prepares the sensor data to be encoded into a LoRaWAN format.
  - LoRaWAN send: Sends the prepared message over LoRaWAN communication protocol.
- ThingSpeak
  - Link in - LoRaWAN rx: Inputs the data from the previous "link out" connection.
  - TTN downlink: Sends the LoRaWAN message to a specific device on The Things Network.
  - TTN uplink: Receives data uplinked from the specific device via The Things Network (TTN).
  - JSON - Parse Payload: Parses the JSON payload received.
  - Function - Extract DHT22 data: Extracts meaningful values like temperature and humidity from the payload.
  - Function - ThingSpeak: Prepares the data for uploading to the ThingSpeak.
  - HTTP Request: Sends the data via an HTTP GET request to ThingSpeak's server.
  - CSV: Converts the payload into CSV format for saving.

- File: Saves the CSV data into a file on the local system.
- Function - Plot data: Formats the sensor data to be visualized in the dashboard.
- UI chart & UI Gauge: Displays the data as chart and/or gauge.

3 EQ3

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.

The figure 2 shows the reproduction of Figure 5 of the paper.

As we can see from the picture, SN5 and SN4 overlap each other and have DER value between 0.8 and 1.0, while SN3 is a curve and its DER value decreases exponentially with higher number of nodes, as happened in the paper figure. In particular, for simulating this figure I run *lorasim/loraDir.py* and simulated three time with different exp: 3,4 and 5, and same number of nodes (from 1 to 1600), transmission rate (1e6) and duration (86400000). For plotting the three DER I used for SN3 exp4, for SN3 exp4 and for SN5 exp5, having as reference the paper parameters.

The figure 3 shows the reproduction of Figure 7 of the paper.

As shown in the graph, a higher number of sinks corresponds to a higher DER value at each node count. Moreover, the curves for different sink counts align closely with those presented in the reference paper. In particular, for simulating this figure I run *lorasim/loraDirMulBS.py* and simulated 6 times, for each number of sinks (1,2,3,4,8,24) and for each number of nodes (from 1 to 1600). For all simulations, I set duration 86400000, transmission rate 1e6, collisions 1 and exp 0, since in the paper SN1 experiment uses SF12, BW=125KHz and CR 4/8 which is the configuration for exp 0.

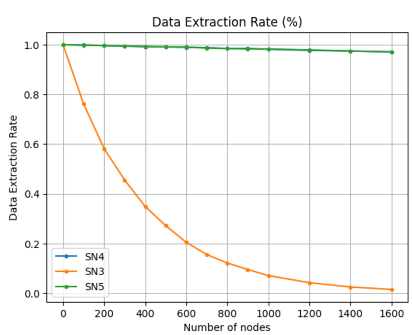

Figure 2: Simulation of Figure 5

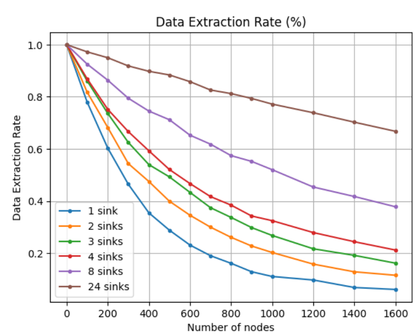

Figure 3: Simulation of Figure 7