Elias Borge-Ilseth, 11075526
Eskil Myklatun Østbø, 11074713 27.04.2025

Internet of Things Exercise 3

EQ1

LoRa uses ALOHA for data transmissions. Hence, we compute the ALOHA success rate (SR) with the formula:

$$\mathrm{SR} = \mathrm{e}^{-2G} = \mathrm{e}^{-2\lambda t} \geq 0.7$$

By using this formula, we can compute the maximum value of t which ensures a minimum success rate of 70%:

$$t \leq -\frac{\ln(0.7)}{2N \cdot \lambda} = -\frac{\ln(0.7)}{2 \cdot 50 \cdot \left(\frac{1}{60}\right)} = 0.214\,\mathrm{s} = 214\,\mathrm{ms}$$

where $N = 50$ number of nodes, $\lambda = 1$ packets/min = 1/60 packets/sec is the transmission rate of each node and t is packet airtime in seconds.

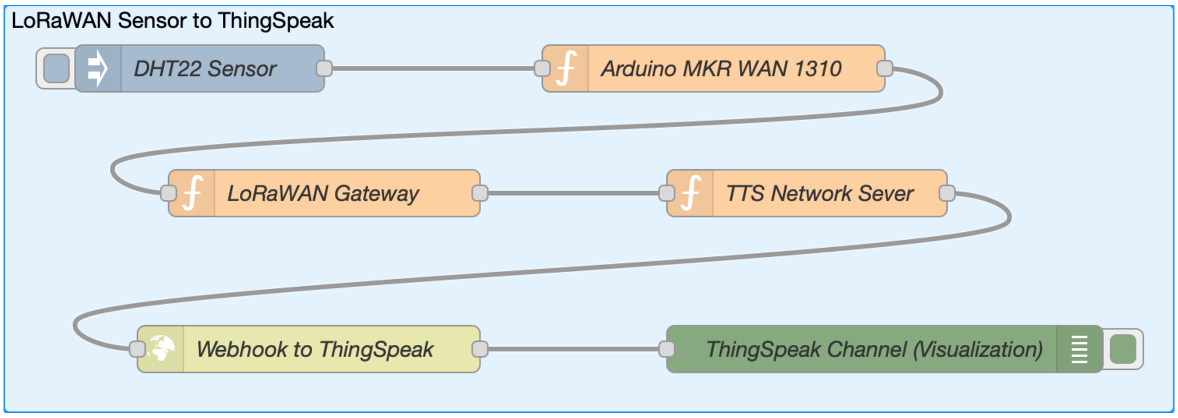
With $L = 3 + 26 = 29$ bytes input payload, EU region carrier frequency of 868 MHz, bandwidth 125 kHz, and using https://www.thethingsnetwork.org/airtime-calculator to calculate the airtime of the packet:

- SF8 → $t = 154.1$ ms
- SF9 → $t = 287.7$ ms

The highest spreading factor that achieves ≥ 70% success rate ($t \leq 214$ ms) is **SF8**.

EQ2

Node red system block diagram of system:



Steps we need to take to get the system fully operational:

1. **Hardware setup (DHT22 → Arduino MKR WAN 1310)**: The Arduino MKR WAN 1310 microcontroller connects to the DHT22 sensor data's pin via digital pins to read temperature and humidity data values. Power the sensor by connecting the GND pin and 5V.
2. **Software setup (Arduino MKR WAN 1310 → LoRaWAN gateway)**: The microcontroller formats the sensor data and transmits the formatted data using the LoRaWAN protocol. The software programming can be done in Arduino Software IDE to program the microcontroller to read sensor values from the DHT22 sensor, format and encode payload, connect to LoRaWAN and send the uplink messages to the LoRaWAN gateway. Join the LoRaWAN network using Over-The-Air-Activation (OTAA) with JoinEUI, Device EUI (DevEUI) and AppKey. Then, the data is transmitted uplink to the LoRaWAN gateway.
3. **LoRaWAN integration (Gateway and Network Server)**: The Arduino sends data over LoRa radio to a nearby LoRaWAN gateway. The data packets are received by the LoRaWAN gateway, which forwards them to the LoRaWAN network server, e.g. The Things Stack (TTS).
4. **Webhook integration (Forwarding to ThingSpeak)**: In TTS, a webhook integration is configured. We use our ThingSpeak Write API key in the webhook URL to send data to our ThingSpeak channel:
   https://api.thingspeak.com/update?api_key=API_KEY&fieldX=DATA
5. **ThingSpeak visualization**: In ThingSpeak, we create a channel with fields for temperature and humidity. ThingSpeak receives the data messages, and maps the temperature and humidity data as separate fields "field1" and "field2". Now, we can monitor and analyse sensor data remotely.

EQ3

*Refer to **LoRasim_modified.ipynb** for modified code.*

Recreating Figure 5

When recreating figure 5, we use the simulation function:

./loraDir.py <NODES> <AVGSEND> <EXPERIMENT> <SIMTIME> [COLLISION]

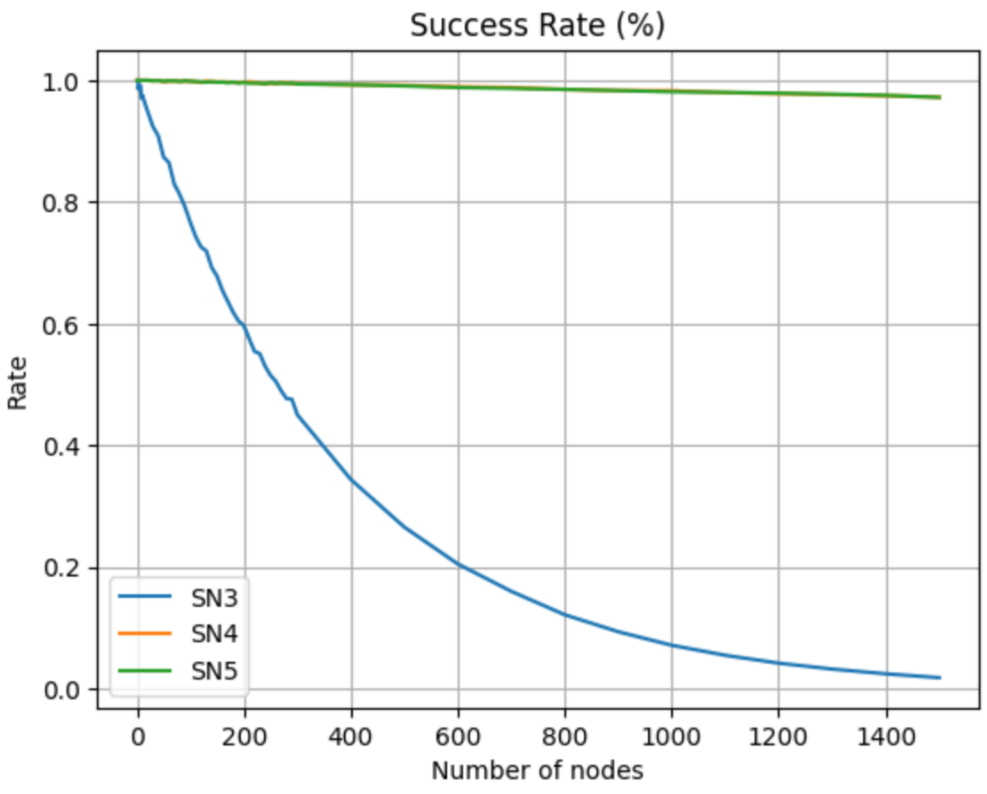
The parameters used for the simulation in to recreate figure 5 is:

- NODES: simulating with **1 to 1600** nodes in the network.
- AVGSEND: 10^6. The average sending interval in milliseconds. In minutes, this is every 16.7 minutes, as indicated in the paper.

- EXPERIMENT:
    - **4 for SN3**: use the settings as defined in LoRaWAN (SF12, BW125, CR4/5).
    - **3 for SN4**: optimise the setting per node based on the distance to the gateway. This is to minimize the airtime.
    - **5 for SN5**: similar to experiment 3, but also optimises the transmit power.
- SIMTIME: 86400000 millisecond. The total running time.

When plotting the results for both figure 5 and 7, we can observe our results are not identical to the plots in the paper. This could be caused by some slightly different simulation settings. However, our simulations are "shape-correct" and we can in both cases deduce similar conclusions based on the plots. Figure 5 displays that SN4 and SN5 performs significantly better than SN3 when the number of nodes increases. Additionally, the performance does not notably increase when using SN5 to minimize the transmission power of each node, compared to SN4. This is why SN4 is not visible on our plots, because it is "under" SN5.

Figure 5, recreated:



Recreating Figure 7

When recreating figure 7 we need to use the simulation function:

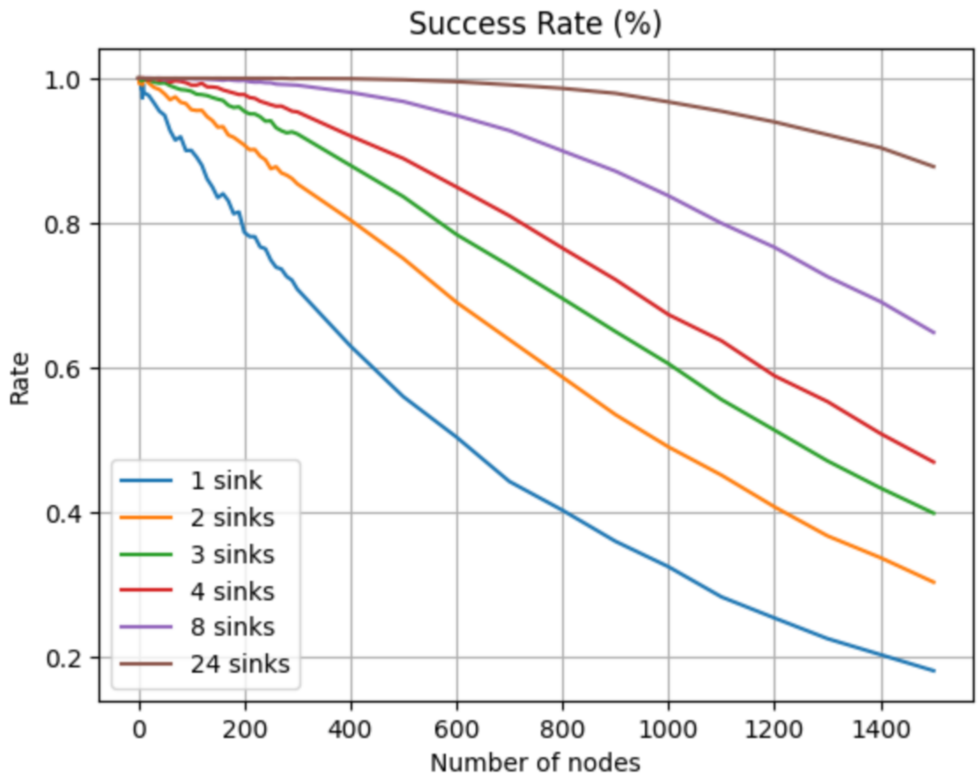
./loraDirMulBS.py <NODES> <AVGSEND> <EXPERIMENT> <SIMTIME> <BASESTATIONS> [COLLISION]

to perform the simulation with multiple base stations. The parameters used for the simulation in this case is:

- NODES, AVGSEND and SIMTIME: same as for figure 5.
- EXPERIMENT: 1. To use the SN1 settings with the the slowest datarate (SF12, BW125, CR4/8). It is similar to experiment 0, but use a random choice of 3 transmit frequencies.
- BASESTATIONS: 1, 2, 3, 4, 8, 24. Equal to the number of base stations (sinks) simulated in the paper.

In real LoRaWAN networks, nodes don't all shout on the same frequency, they randomly pick from multiple channels. Using experiment setting 0 (single frequency) would completely underestimate how effective adding more base stations is. Using experiment setting 1 simulates better a real-world frequency diversity – critical for spatial reuse and collision avoidance. This is why we need to use experiment setting 1 when reproducing Figure 7. Our simulation deviates slightly from Figure 7 in the paper. However, we can conclude that the network success rate increases while increasing the number of nodes in the network.

Figure 7, recreated:


Success Rate (%)
Rate
Number of nodes
1 sink
2 sinks
3 sinks
4 sinks
8 sinks
24 sinks