# Tool user manual

# MULANE

ENG - ENG

# Contents

# Using the MULANE tool

## 1. About the MULANE tool

MULANE, which stands for Multi-methods based spreading factor for LorA Networks is a tool that allows you to simulate with several methods of improving the performance of LoRaWAN networks.

LoRaWAN is a telecommunication protocol using LoRa which is a technology that allows communication between various objects through an internet connection and thus realize a network of objects.

But un LoRaWAN network can encounter four types of collisions:

**Frequency collision:**

A frequency collision happens when two packets have a frequency too close, we can summarize this in three cases:

- If the frequency difference of the two packets is less than or equal to 120KHz and the bandwidth of the two packets is equal to 500.
- If the frequency difference of the two packets is less than or equal to 60KHz and the bandwidth of the two packets is equal to 250
- If the frequency difference of the two packets is less than or equal to 30KHz

**SF collision**

If the values of the SFs of the two packets are equal then an SF collision type collision is detected.

**Energy collision**

A collision of this type detected, if one of the following cases occurs:

- The absolute value of the difference between the rssi (Received Signal Strength Indication) which is a measure of the power level in reception of a signal coming from an antenna of the two packets is less than a threshold of 6 dB (The functional threshold power ), in this case the two packets are too close to each other, the two collide, return both packets as victims, otherwise;
- If, the difference between the rssi of the two packets is strictly less than a threshold of 6 dB (p1.rssi - p2.rssi <(The functional threshold power)), in this case p2 overpowered p1, return p1 as victim.
- Otherwise ; p2 was the weakest packet, return p2 as victim

**Timing collision**

Assuming p1 is the freshly arrived packet and this is the last check, we have already determined that p1 is a weak packet, so the only way to win is to be late enough (only the first n-5 symbols of the preamble overlap).

Assuming 8 preamble symbols make it like a victim.

**Mulane** is there to present several methods that can increase the performance of LoRaWAN networks and learning to use this tool is not complicated, but it assumes to understand the methods to integrate well. We recommend that you study this manual in detail before using the tool.

This user manual provides an overview of the features of the application and gives step-by-step instructions for performing various tasks.

## 2. Required configuration

- Before using MULANE, please make sure that Python 3 is installed on your machine with environment variable “python”.
- To install the necessary libraries run the “requirements.txt” file with the command “pip install –r requirements.txt” before the first use.

**To launch the tool**, open the terminal in the folder which contains “mulane.py” with the command “python mulane.py”.

# Exploring the tool

## 1. Tool operation

The tool works in 4 steps:

- **Choice of fixed parameters** The user chooses fixed parameters which do not change for all methods.

- **Choice of method** The user chooses the method with which he wants to perform simulations.

- **Choice of parameters** Each method has specific parameters that the user can define.

- **Graphics settings** Before launching the simulation, you can choose whether you want to represent the results in the form of graphics or text only.

## 2. Home Page

The start page is a practical platform that provides information and allows you to reach the different parts of the tool.

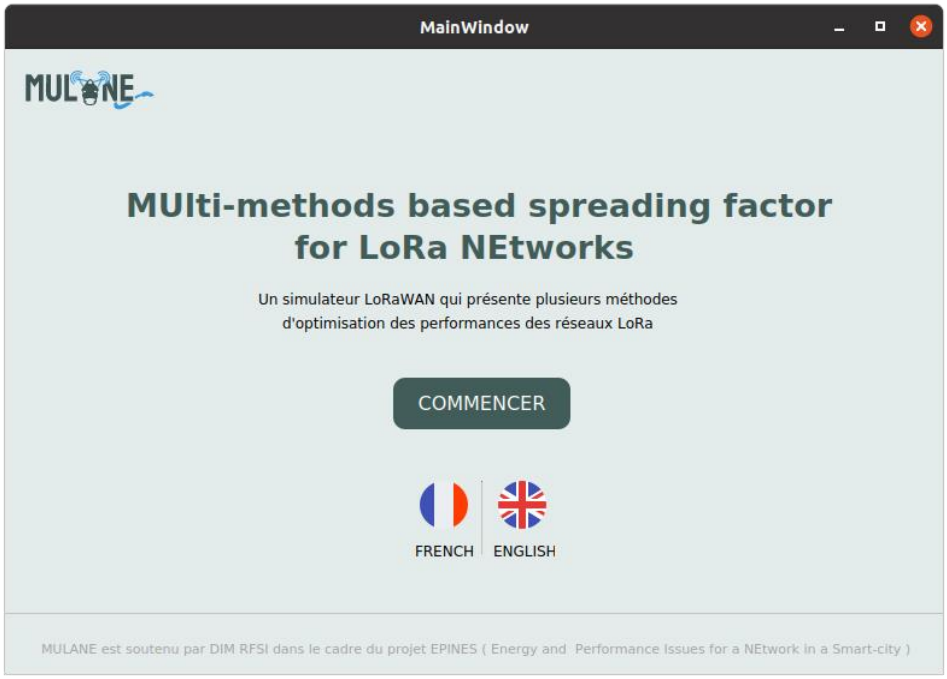


Figure (1): Mulane tool home window

- **To choose the language**, click on the flag of the desired language
- **To start a new simulation**, click on Begin in English otherwise on Begin in French.

# 3. Starting a simulation

## 3.1 Choice of fixed parameters

The different methods have parameters in common which do not change from one method to another and the page for the choice of fixed parameters allows them to be initialized.

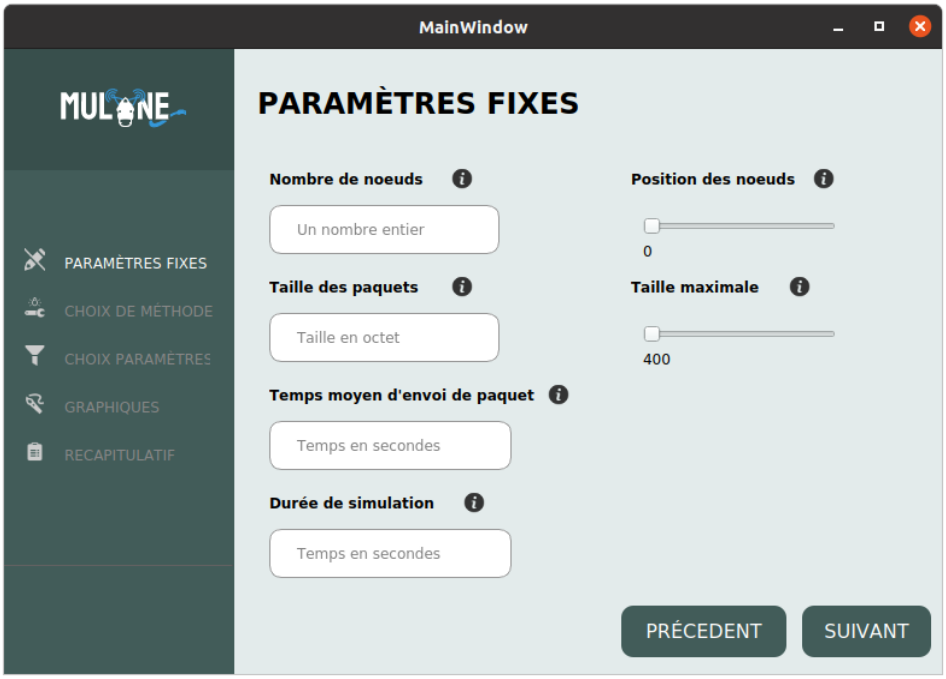


Figure (2): choice of fixed parameters window.

The fixed parameters are:

- **Number of nodes**

Refers to the number of nodes in the network, each node will be placed in the network according to the Node position and Maximum size parameters.

- **Position of nodes**

Its value can go between 0 and 50, this number represents the number of circles around the base station where the nodes will be placed at 0 the placement of the nodes is totally random and the more the number is greater the placement is less less random.

- **Maximum size**

The value of these parameters is between 400 (m) up to the maximum distance which is 8921.359 (m) and simply sets the maximum distance where the nodes can be placed.

In addition to the other parameters which are Packet size, Average packet sending times and simulation time.

- The “i” next to the text fields are used to explain the parameter
- **To go back**, press Back on Previous.
- **To go to the next step**, tap Next.

## 3.2 Choice of method

### 3.2.1 The methods present in Mulane

Mulane presents several methods of improving performance, in this version it has 7 methods.

- **Static Random**

This method is very similar to the basic LoRaSIM simulator with only one modification, it is the initialization of the SF which is done in a random manner.

- **Dynamic Random**

The method works with a constraint on the collision rate, its operation is simple, each node of the base station transmits packets, if a packet collides the packet is sent using another SF chosen at random from among the SFs of the station.

- **Dynamic P-Random**

Its operation resembles that of Dynamic Random but by using an additional value which is the P whose value is chosen by the user, the value that we have chosen is 0.4. As usual, the nodes of the base station transmit packets, if one of our packets encounters a collision, then we draw a random value which varies between 0 and 1, if the number drawn is greater than or equal to P, we change SF otherwise we do nothing and force the transmission of our packet with the SF chosen initially.

- **STEPS Full and STEPS Individual**

In this method we introduced a new idea which is the score table, to put it simply, we have a table which represents the probabilities of choosing an SF and the sum of the values of the table will be 1 and during a collision the choice of the SF will therefore be done randomly according to the probabilities in the table.

Each collision encountered using an SF, the probability of the latter will be decreased in our table, and otherwise it will be increased.

The sum of the table will always be equal to 1 because a normalization is carried out after each modification of the values.

During a collision the SF will therefore be modified to recover the best SF, and if there are several collisions in a row on the same SF, reduce its weight even more to avoid continuing to use it.

Our weight table makes it possible to choose the best SF parameter during transmission, so it is possible to recover this data with the objective of transmitting them to new nodes when they are created in future simulations.

The objective is to use this data when setting up a new simulation, which is why when creating nodes we will retrieve the data from the node closest to the new node which is stored in a database .

STEPS Full and STEPS Individual are the same simulators in their operations, the difference is that STEPS Individual is a simulator in more real conditions than STEPS Full because it has less information on the packets it sent (it does not know if a packet was received only if there is an ack).

**STEPS E-Greedy and Boltzmann**

**Calculation of minimum SF :** The minimum SF represents the SF ∈ {7,8,9,10,11,12} for which the receiver will be able to receive the transmission this SF depends on the distance between the node and the base station that is why it We will need to be interested in the reception power, which depends on the transmission power, and in the sensitivity of the receiver, which depends on the SF and the bandwidth because indeed the transmission can only be successfully received if the reception power is greater than the sensitivity.

First, we must observe the reception power Prx which is given to us with the equation: $Prx = Ptx + GL - Lpl$

Where Ptx is the transmission power which is configurable at the level of each transmitter and is theoretically adjustable from -4dBm to 20 dBm in steps of 1dB but is limited by hardware implementations and frequency regulations. GL is the sum of the transmission and reception gains and losses and Lpl is the following path loss calculation obtained: $Lpl = Lpld0 + 10nlog10\ (d\ d0)$.

Where Lpld0 and n are respectively the loss at the reference distance d0 and the path loss coefficient and are obtained experimentally, d is the distance between the node and the base station.

Now that we have seen how the reception power is obtained, we must check that it is much greater than the sensitivity of the receiver, the sensitivity depends on the chosen SF as well as on the bandwidth. Using the LoRa SX1272 calculator available at this link we obtain the following sensitivity table (in dBm):

| | | Bandwidth | | |
|---|---|---|---|---|
| | | 125 | 250 | 500 |
| SF | 7 | -123 | -120 | -117 |
| | 8 | -126 | -123 | -120 |
| | 9 | -129 | -126 | -123 |
| | 10 | -132 | -129 | -126 |
| | 11 | -134.5 | -131.5 | -128.5 |
| | 12 | -137 | -134 | -131 |

Sensibilité calculé du receveur pour différent SF et BW

The minimal SF is therefore the smallest SF for which the reception power is greater than the sensitivity of the receiver, so we have for each node a set of available SFs S such that S = [Sfmin, 12]

**Estimated rewards:** Each node uses an array containing reward estimates, ri for each of the 6 available SFs, the array is initialized either from the database experiments or based on the distance from the node to the gateway if the database is empty. On each transmission, the node will receive a reward R equal to 1 in the event of acknowledgment and to 0 in all other cases. After the transmission the Eri reward estimate for i chosen in the transmission will be updated according to the formula: Eri = Eri + α (R - Eri)

Where α is a learning factor and α∈ (0,1] chosen by the user.

**E-Greedy:** When selecting an SF for transmission, a $SF \in S$ will be chosen at random with a probability ε, otherwise the SF with the best Er will be chosen. We therefore have Pi the probability of choosing the SF i such that:

$$P_i = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{K} & \text{si } i = \text{argmax } Er_i \\ \frac{\varepsilon}{K} & \text{sinon} \end{cases}$$

Where ε∈ (0,1] is chosen by the user and K is the cardinality of S.

**Boltzmann exploration:** When selecting an SF for transmission, the SF will be chosen randomly according to a probability Pi depending on the estimate Eri of each SF as well as on a parameter τ∈R + such as:

$$P_i = \frac{e^{\frac{Er_i}{\tau}}}{\sum_{k=SF_{min}-7}^{6} e^{\frac{Er_k}{\tau}}}$$

Where τ is chosen by the user, the more τ approaches infinity the more the algorithm behaves randomly and the closer τ is to 0 the more the algorithm behaves like a greedy algorithm.

For all $SFi < SFmin$ we will have $Pi = 0$.

### 3.2.2 Choice of method


Figure (3): method choice window

- The window allows the user to choose the method using a drop-down menu (the drop-down menu has been chosen for the representation of the different methods in order to allow new methods to be added in the future)
- When a method is selected, a description of the method appears just below to explain to the user the specifics of the method
- **To go back**, press Back on Previous.
- **To go to the next step**, tap Next.

### 3.2.3 Choice of the parameters of a method

Each method has specific parameters so everything depends on the method chosen, the parameters are not the same.


Figure (4): choice of method parameters window

Take the STEPS_full method as an examplebecause it has most of the enterable parameters which are:

- **Collision model**

Or collision detection mode, we have two detection modes, full collision and single collision

We know that there are four types of collisions (Frequency collision, SF collision, timing collision and Energy collision)

The full detection mode will detect all these types of collision while the simple collision mode will only detect type collisions (Frequency collision and SF collision).

- **Number of launches**

The number of times the simulator will be launched and the results will be the average over the value of this parameter.

- **Desired database (DB) size**

Represents the size of DB that the simulator starts with, if for example the user chooses a value of 100, the simulator will prepare a DB of 100 entries and then start the simulation.

- **BD update**

During the simulation the user can choose to allow the update of the DB, that is to say to allow the sharing of experience between the nodes or not to allow the update of DB in order to simulate on a number of fixed BD entries.

- The user fills in the fields
- Again, if all the fields are not filled in, the user does not have the right to go to the next step.

# 4. Simulation results

## 4.1 Results in graphical form

The user can choose to generate the results in graphical form thanks to the graphical parameters page which allows to check the desired results in graphical form.

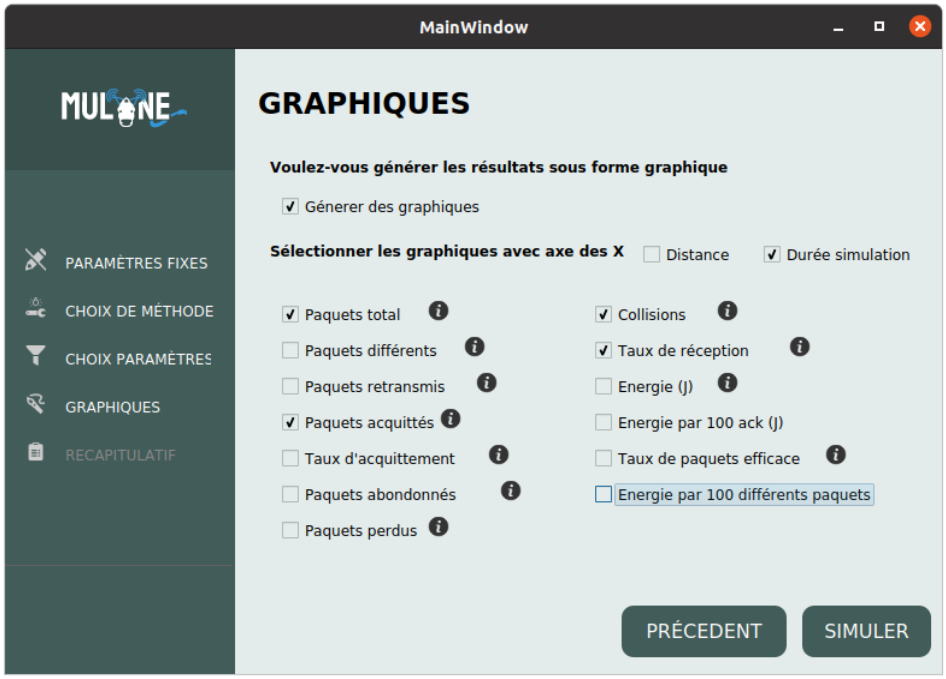


Figure (5): graphic parameters window

- The user has the choice to either generate graphs with the results or not
- If he chooses to generate some he will be able to choose which ones to generate
- He can also choose to generate the graphs either according to the distance or the duration of the simulation and even both at the same time
- If the X axis is not chosen, both will be generated

## 4.2 Presentation of the results

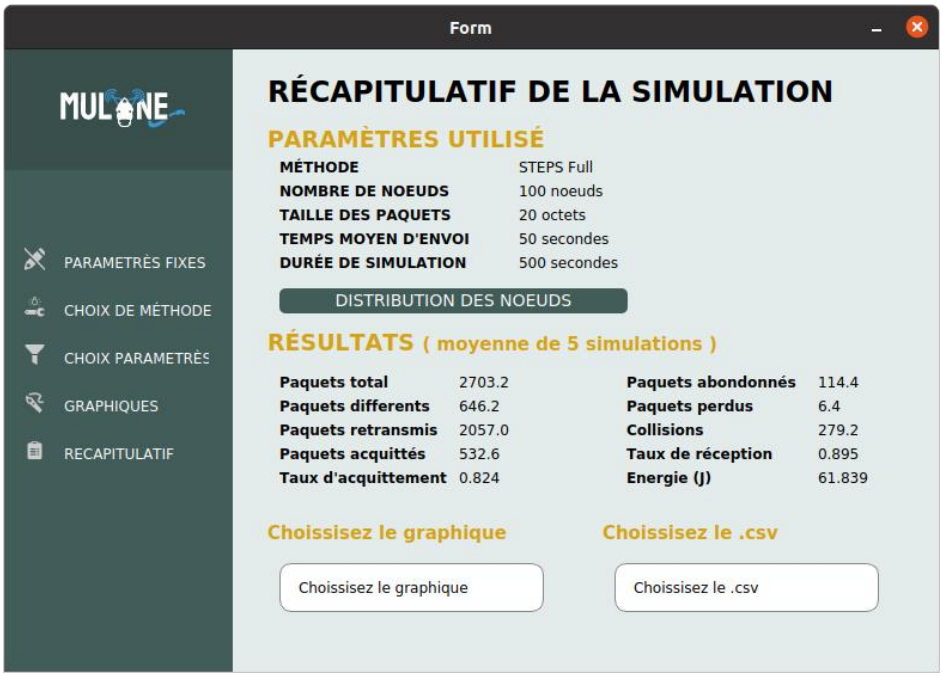


Figure (6): simulation results window

The window displays in a first part a reminder of the parameters used as well as the distribution of the nodes with respect to the base station and the SF used by each node.

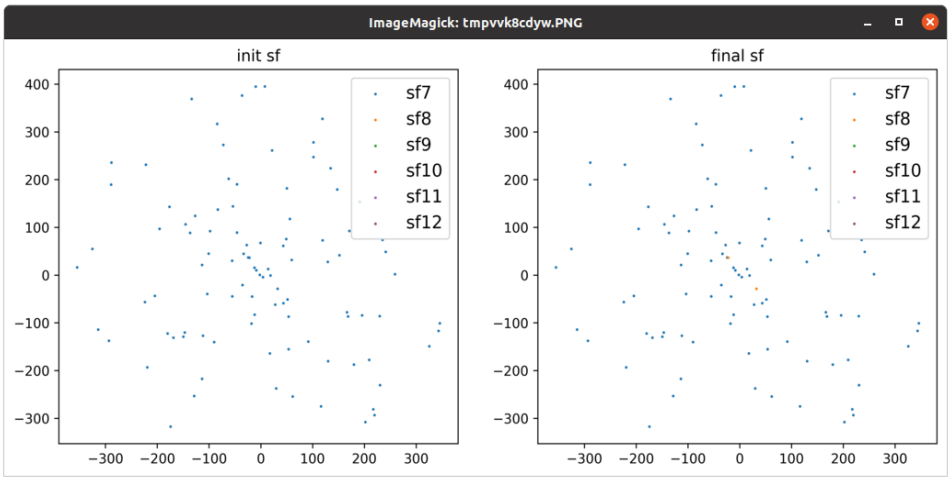


Figure (7): node repair window and their SF.

**In the second part** the results of the simulation, namely:

- **Total packages:** The total number of transmissions made by the nodes of the network can be said to be (total packets = different packets + retransmitted packets).

- **Different packages:** The number of unique packets sent by nodes in the network.

- **Retransmitted packets:** the number of packet retransmission performed by network nodes, a packet is retransmitted if its first transmission was inconclusive.

- **Acknowledged packets:** The number of packets acknowledged by the base station.

- **Acknowledgment rate:** Represents the number of acknowledged packets / number of different packets.

- **Abandoned packages:** number of packets dropped by nodes, a packet is dropped if it has been retransmitted more than 8 times.

- **Lost packages:** Number of packets lost in the network during transmission.

- **Collisions:** Number of collisions detected in the network during transmission.

- **Reception rate:** Represents the number of packets received / number of total packets.

- **Energy:** The energy consumed by the nodes to make the necessary transmissions.

**In the third part** the user can choose the graph he wants to display in addition to the .CSV files to allow him to check the values.

The user will therefore be able to visualize the different graphs that he has checked from the drop-down menu, for example in our simulation we have chosen to generate the graphs of: total packets, acknowledged packets, collisions and reception rate.

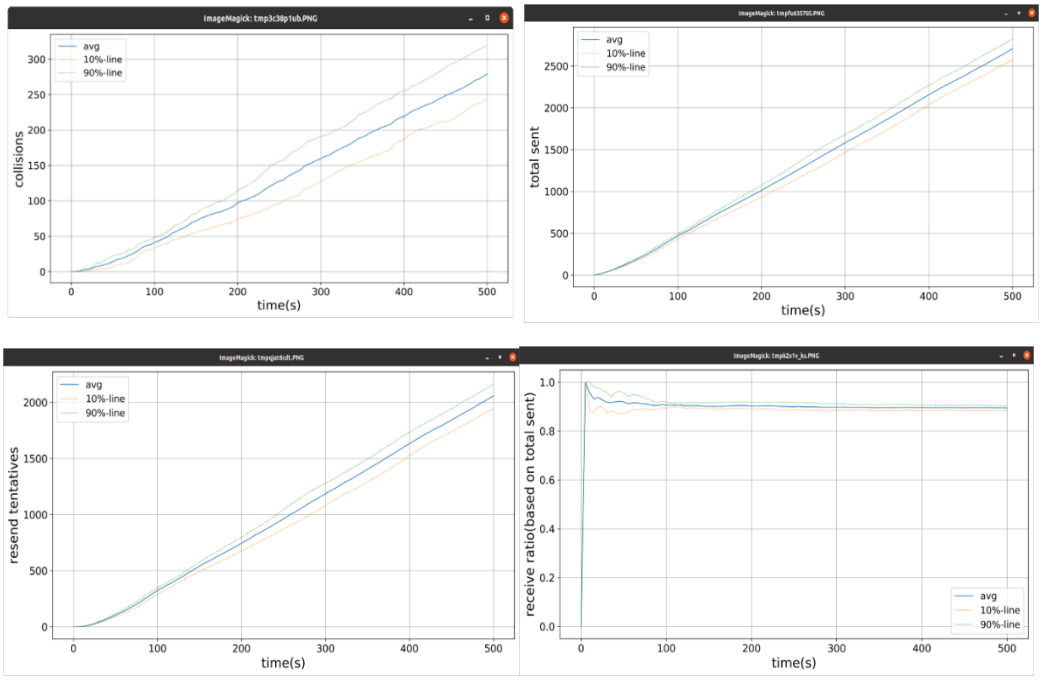


Figure (7): graphic representation of the results

And the same for .CSV files, you should know that for each graph generated, its .CSV file is also generated and the user will be able to visualize the data with which the graphs were drawn, subsequently he can either check the data, draw another type of chart using its data.

| | A | B |
|---|---|---|
| 1 | sim time | collisions |
| 2 | 0 | 0 |
| 3 | 5000 | 2 |
| 4 | 10000 | 2 |
| 5 | 15000 | 2 |
| 6 | 20000 | 2 |
| 7 | 25000 | 4 |
| 8 | 30000 | 4 |
| 9 | 35000 | 4 |
| 10 | 40000 | 4 |
| 11 | 45000 | 4 |
| 12 | 50000 | 5 |
| 13 | 55000 | 5 |
| 14 | 60000 | 5 |
| 15 | 65000 | 6 |
| 16 | 70000 | 6 |
| 17 | 75000 | 8 |
| 18 | 80000 | 9 |
| 19 | 85000 | 13 |
| 20 | 90000 | 13 |
| 21 | 95000 | 13 |
| 22 | 100000 | 13 |
| 23 | 105000 | 13 |

| | A | B | C | D |
|---|---|---|---|---|
| 1 | x | y | distance | collisions |
| 2 | -19.54017628 | -23.45116542 | 30.52500039 | 1 |
| 3 | 214.4471867 | -14.00328005 | 214.9039034 | 0 |
| 4 | 12.78123521 | 155.0745938 | 155.6004165 | 0 |
| 5 | 81.95404787 | 332.1726193 | 342.1331831 | 0 |
| 6 | 9.979289283 | 8.880503129 | 13.35850105 | 0 |
| 7 | -32.76900868 | -169.1656209 | 172.3102296 | 0 |
| 8 | -222.9247223 | 244.945408 | 331.2003694 | 4 |
| 9 | 110.1513331 | -291.5468964 | 311.6615295 | 2 |
| 10 | 136.6529455 | -114.778751 | 178.4606096 | 2 |
| 11 | -2.948895646 | 133.7589595 | 133.7914617 | 1 |
| 12 | 339.1876743 | -51.87423013 | 343.1314823 | 0 |
| 13 | -121.2951936 | -25.02496377 | 123.8497993 | 3 |
| 14 | 116.7039284 | -144.7745413 | 185.9555719 | 0 |
| 15 | 25.53573676 | 8.565486931 | 26.93401972 | 0 |
| 16 | -50.14135651 | 41.87195857 | 65.3254663 | 2 |
| 17 | 257.3976922 | 194.6862998 | 322.7325941 | 0 |
| 18 | 62.95818954 | -299.1027204 | 305.6569498 | 1 |
| 19 | -52.42164265 | -166.7133331 | 174.7608768 | 0 |
| 20 | 111.9083397 | 342.2204934 | 360.0532497 | 1 |
| 21 | 332.3745308 | -7.572232048 | 332.4607758 | 3 |
| 22 | 2.030091914 | -264.2418852 | 264.2496834 | 1 |
| 23 | 303.9815645 | -172.5273252 | 349.5289251 | 2 |

Figure (8): example of a CSV file of the results

## 5. Conclusion

The Mulane tool allows you to simulate with different methods created to optimize the performance of LoRaWAN networks.

A second tool “Mulane & Comparison” was created to allow comparison between the basic methods (LoRaSIM, LoRaFREE and LoRaMAB) and all the new methods of Mulane.