MPI Simulation

In remote regions, where sensors are not available, we can create new data by simulating the environment of the interested area. Simulations are based on population dynamics and each noise is influenced by the presence of people or cars.

Each simulation needs many parameters:

* Width and length of the rectangular area in meters that we are interested to simulate
* Number of people and cars present in the area
* Noise in dB produced by each person or car
* Distance in meters that each person or car affects the surrounding
* Moving speed of people and cars
* Timestep in seconds tells the simulation how much time we have to re-calculate the position of people and cars
* Granularity in meters tells how much data we are aggregating together to send aggregated messages
* Latitude and Longitude of the upper left corner of the area

Each simulation is divided into three main steps that are continuously repeated:

1. Calculate noise
2. Recompute position
3. Gather noise

Message Passing Interface is a standardised and portable message-passing designed to function on parallel computing architectures. Each simulation is parallelised by instantiating multiple processes that work together.

**Calculate noise:**

Each process will be in charge of simulating a small number of people and cars in the entire simulation area. Each person and car are simulated independently. Each process will fill a matrix, simulating the area, with all noises generated by the population.

**Recompute position:**

Here, each process will recompute the position of each people and car in the simulation, using simple or complex population dynamics.

These are the most complex and high computational load parts of the whole simulation, that are balanced throughout all processes running in parallel.

**Gather noise:**

In this last step, the master process oversees gathering all matrices from all processes and summing them all up using an MPI\_Reduce. When the master has all data, it is in the end able to send the noise of the simulation to the backend using an MQTT broker.

At the end of each loop between these three main steps, all processes will sleep for the time step specified in the simulation parameters.

Choice to use MPI for the simulation:

* Parallel and distributed discrete event simulation allows the execution of a single simulation program on multiple processors. By splitting up the simulation into logical processes, each logical process can be executed by a different processor. This allow very large-scale simulations by leveraging increased processing power and memory availability.