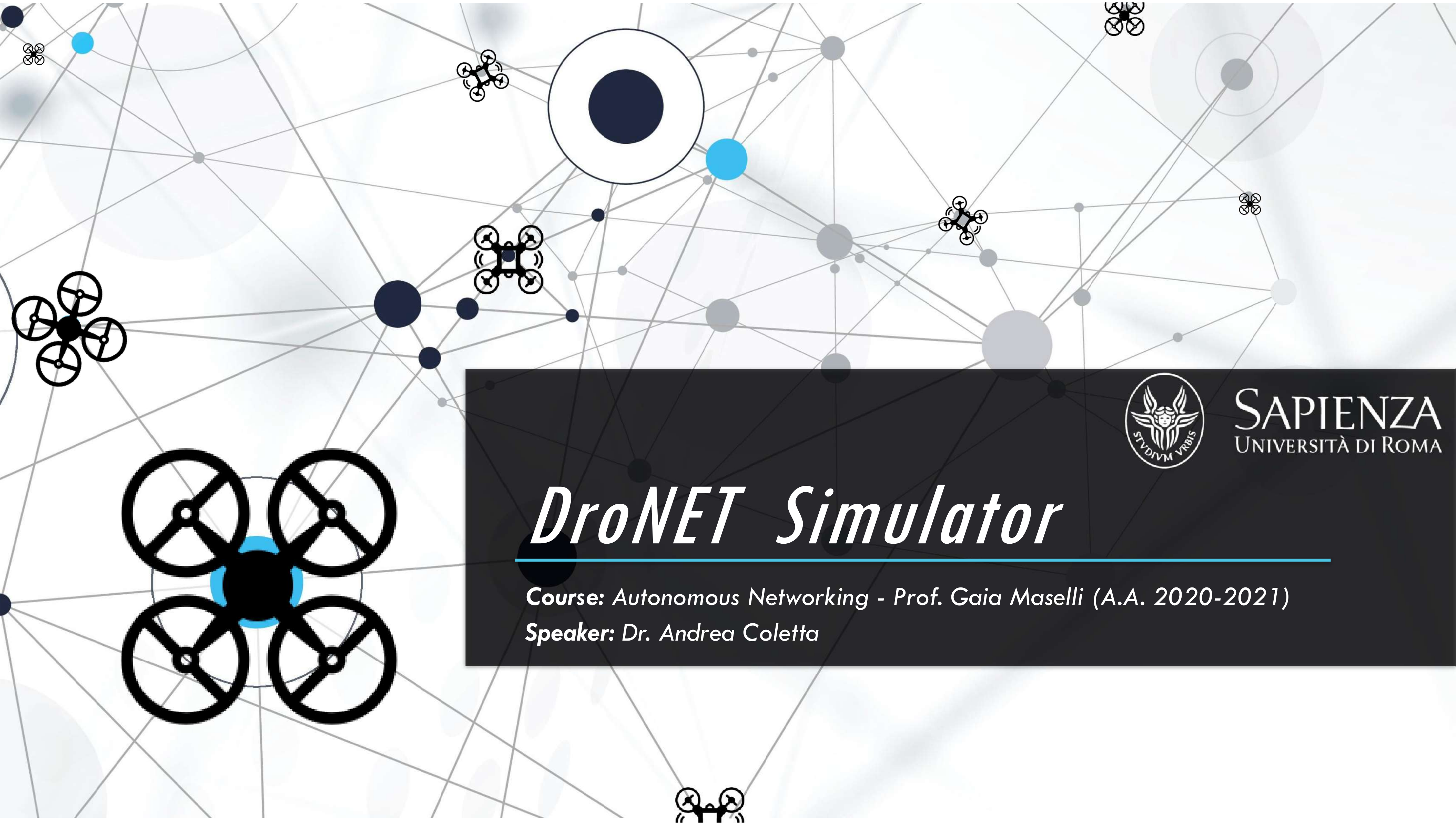


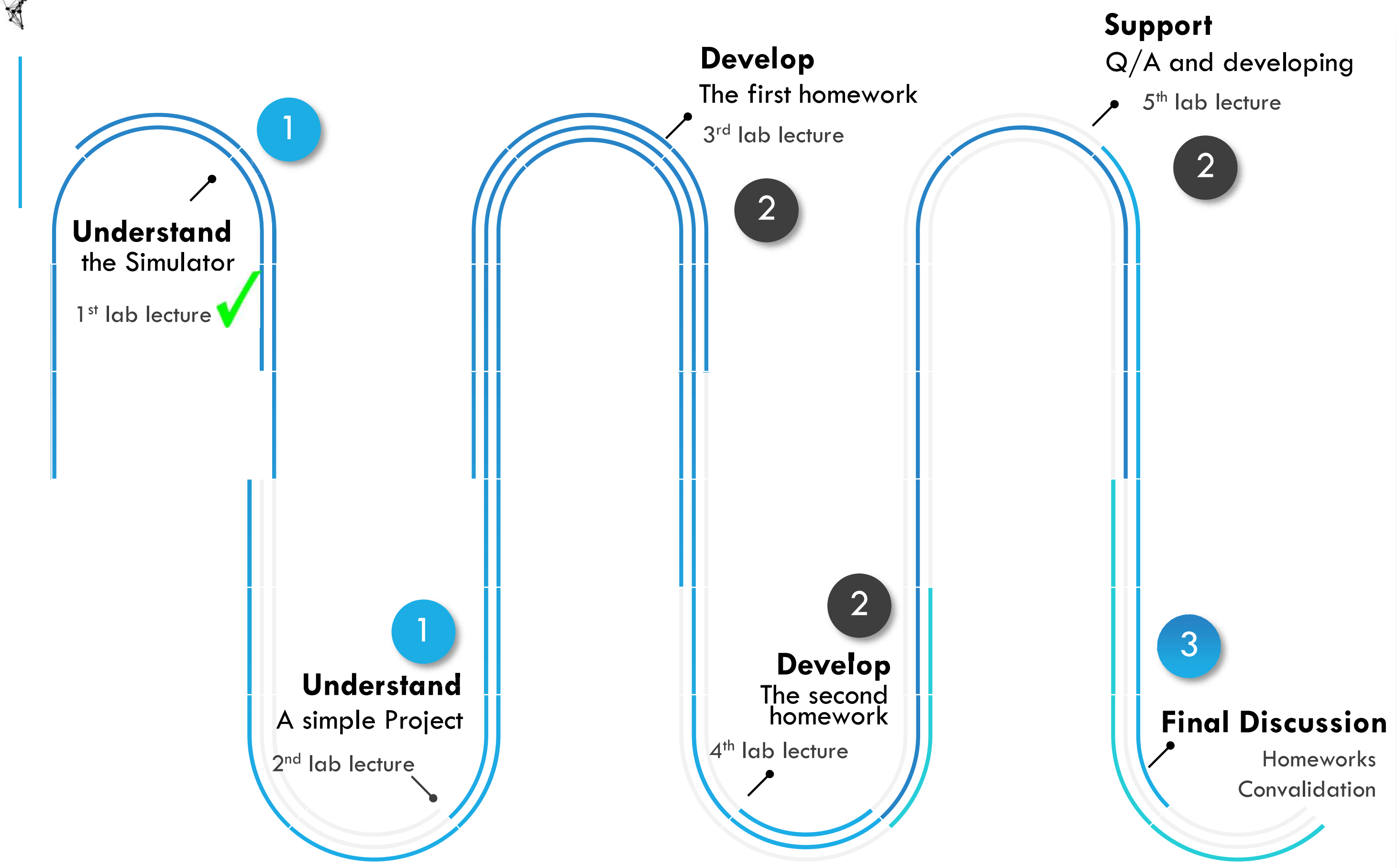
SAPIENZA
UNIVERSITÀ DI ROMA

DroNET Simulator

Course: *Autonomous Networking - Prof. Gaia Maselli (A.A. 2020-2021)*

Speaker: *Dr. Andrea Coletta*

AUTONOMOUS NETWORKING – A.A. 20/21



1 **Understand** the Simulator

1st lab lecture ✓

1 **Understand** A simple Project

2nd lab lecture

2 **Develop** The first homework

3rd lab lecture

2 **Develop** The second homework

4th lab lecture

2 **Support** Q/A and developing

5th lab lecture

3 **Final Discussion**

Homeworks Convalidation

HOW TO PASS THE LAB

1 Understand

2 **Develop**

3 Oral Discussion

SAPIENZA
UNIVERSITÀ DI ROMA

NETWORK SIMULATORS

EVENT BASED SIMULATOR

DISCRETE TIME SIMULATOR

NS3 - https://www.nsnam.org/
OMNeT++ - https://omnetpp.org/

NETWORK SIMULATORS - OVERVIEW

- Discrete Time Simulator

- Python3

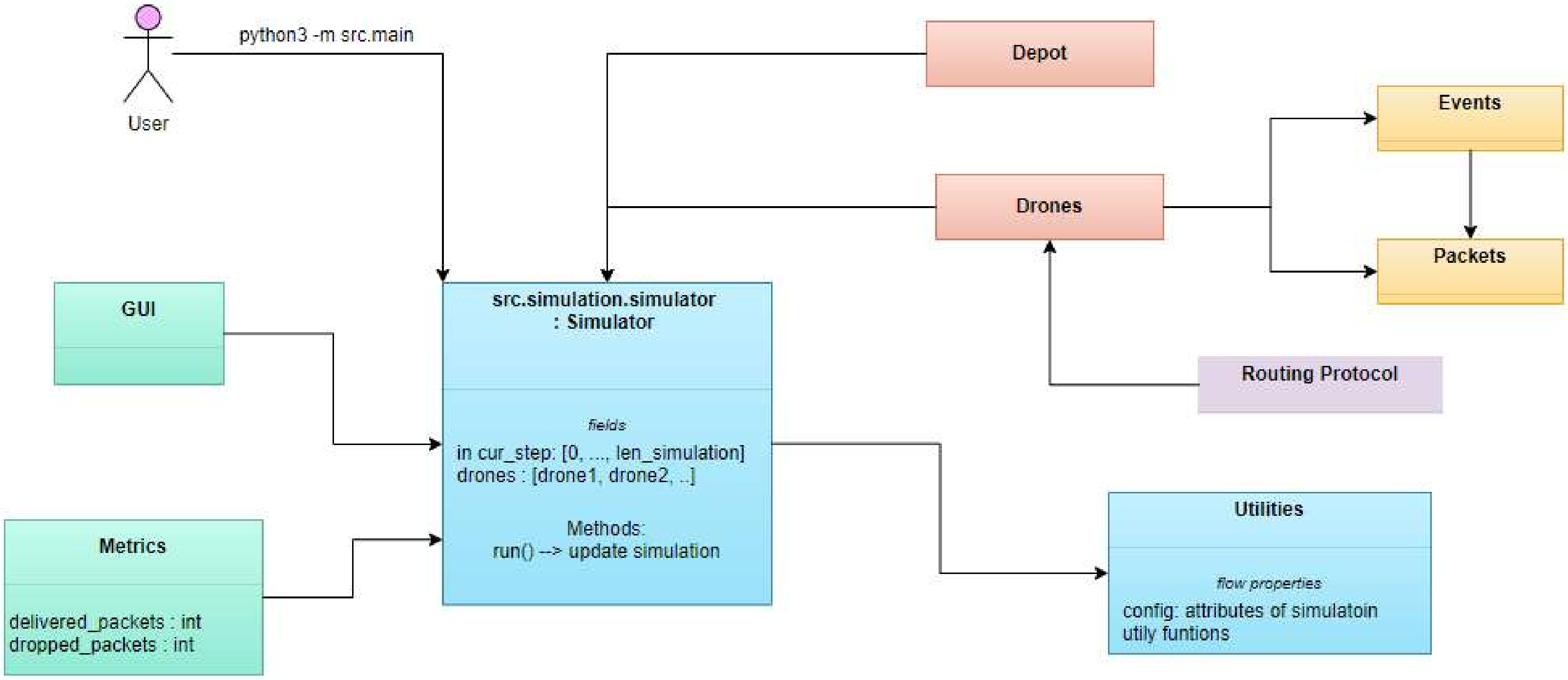
- Linux (the code works on Ubuntu 20.04)

- GitHub (fork the project: **https://github.com/Andrea94c/DroNETworkSimulator)**

- Install libraries using the ***requirements.txt*** file (pip3 install –r requirements.txt)

- **Readme.md** contains useful info and **contacts**

NETWORK SIMULATORS - ARCHITECTURE



NETWORK SIMULATORS - ARCHITECTURE

- .
  - **README.md**
  - **data**
    - tours
      - RANDOM_missions1.json
      - ...
      - RANDOM_missions90.json
  - **src**
    - main.py
    - drawing
      - color.py
      - picture.py
      - pp_draw.py
      - stddraw.py
    - entities
      - **uav_entities.py**
    - ~~experiments~~
    - routing_algorithms
      - BASE_routing.py
      - georouting.py
      - net_routing.py
      - random_routing.py
    - simulation
      - metrics.py
      - **simulator.py**
    - utilities
      - **config.py**
      - random_waypoint_generation.py
      - utilities.py

SIMULATION- RUN

src.simulation.simulator.Simulator.run()

Discrete Time iteration (increase simulation time)

Medium Simulator - It is responsible to simulate the physical layer (drop, delay and delivery packets in the wireless channel)

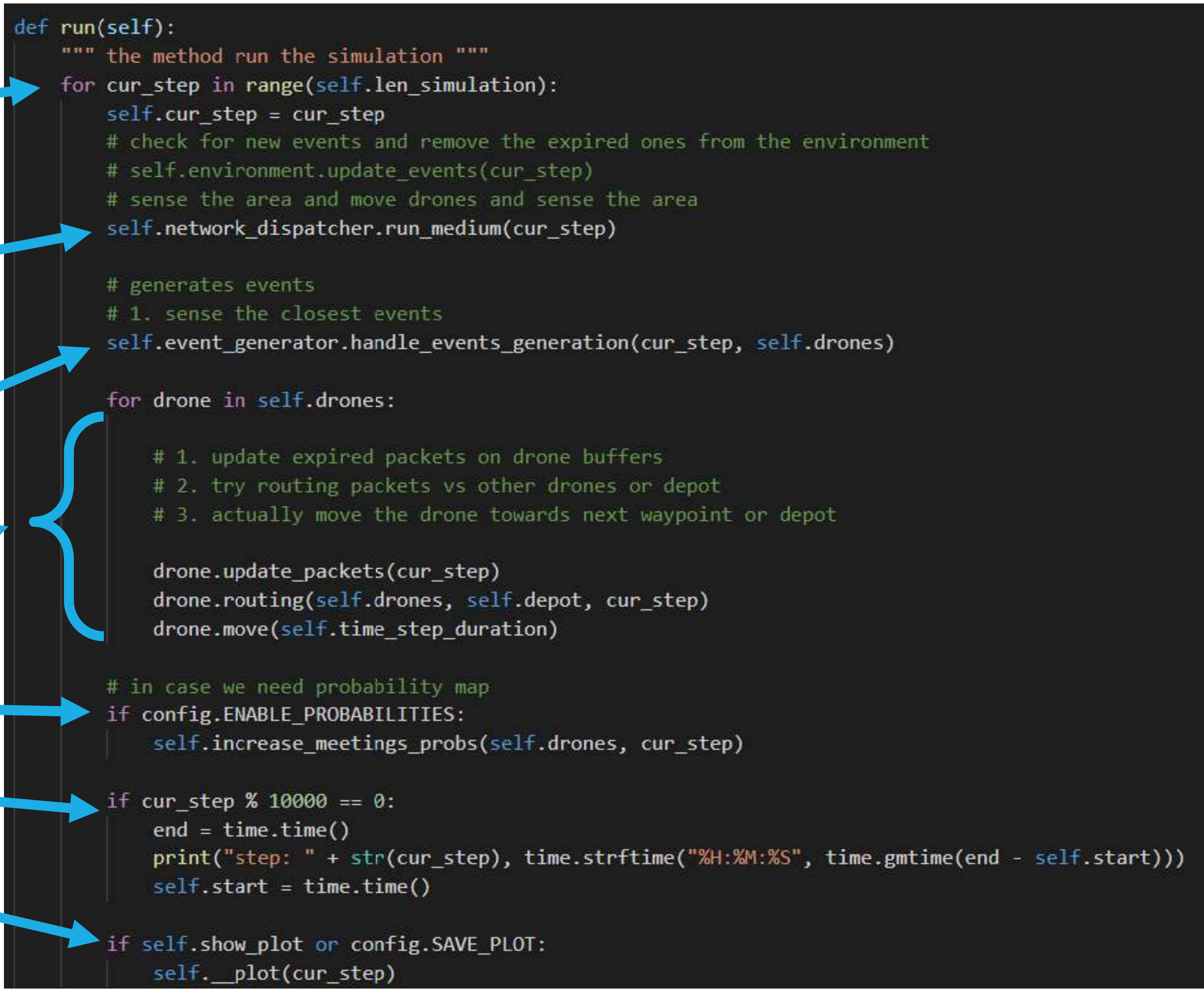
Traffic Generator - It is responsible to generates events/packets on drones.

Traffic Generator - It is responsible to generates events/packets on drones.

Compute probabilities to meet drones

Debug print – notify the current state

If enable, display the simulation or save it



```python
def run(self):
    """ the method run the simulation """
    for cur_step in range(self.len_simulation):
        self.cur_step = cur_step
        # check for new events and remove the expired ones from the environment
        # self.environment.update_events(cur_step)
        # sense the area and move drones and sense the area
        self.network_dispatcher.run_medium(cur_step)

        # generates events
        # 1. sense the closest events
        self.event_generator.handle_events_generation(cur_step, self.drones)

        for drone in self.drones:

            # 1. update expired packets on drone buffers
            # 2. try routing packets vs other drones or depot
            # 3. actually move the drone towards next waypoint or depot

            drone.update_packets(cur_step)
            drone.routing(self.drones, self.depot, cur_step)
            drone.move(self.time_step_duration)

        # in case we need probability map
        if config.ENABLE_PROBABILITIES:
            self.increase_meetings_probs(self.drones, cur_step)

        if cur_step % 10000 == 0:
            end = time.time()
            print("step: " + str(cur_step), time.strftime("%H:%M:%S", time.gmtime(end - self.start)))
            self.start = time.time()

        if self.show_plot or config.SAVE_PLOT:
            self.__plot(cur_step)
```

CONFIG – HOW TO CHANGE SIMULATION PARAMETERS

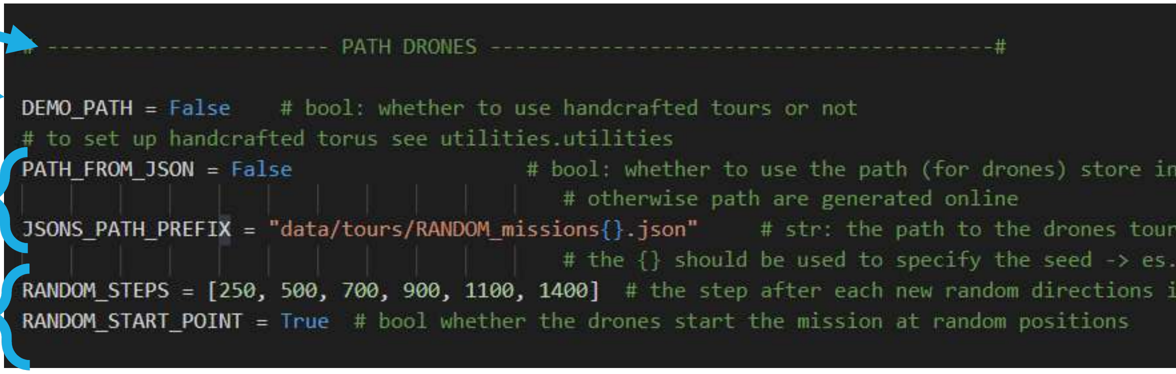
src.utilities.config

Change drones mobility

Use demo (manual specified) paths for ad-hoc test and debug.

Use default paths in data/tours/RANDOM….mission{seed}.json

Build random path at runtime.

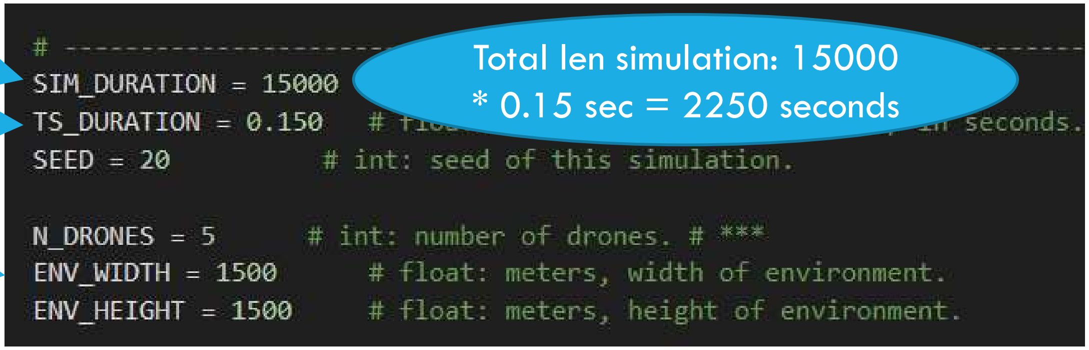


```
# ---------------------- PATH DRONES ----------------------------------------#

DEMO_PATH = False     # bool: whether to use handcrafted tours or not
# to set up handcrafted torus see utilities.utilities
PATH_FROM_JSON = False                         # bool: whether to use the path (for drones) store in
                                                  # otherwise path are generated online
JSONS_PATH_PREFIX = "data/tours/RANDOM_missions{}.json"      # str: the path to the drones tour
                                                  # the {} should be used to specify the seed -> es.
RANDOM_STEPS = [250, 500, 700, 900, 1100, 1400]  # the step after each new random directions i
RANDOM_START_POINT = True  # bool whether the drones start the mission at random positions
```

Number of iteration of the simulation

Simulation world seconds for each iteration

Environment and drones

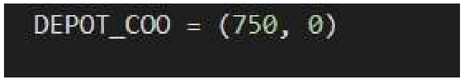


```
# ---------------------------------------------------------------------------
SIM_DURATION = 15000
TS_DURATION = 0.150    # floa                                    in seconds.
SEED = 20             # int: seed of this simulation.

N_DRONES = 5        # int: number of drones. # ***
ENV_WIDTH = 1500      # float: meters, width of environment.
ENV_HEIGHT = 1500     # float: meters, height of environment.
```

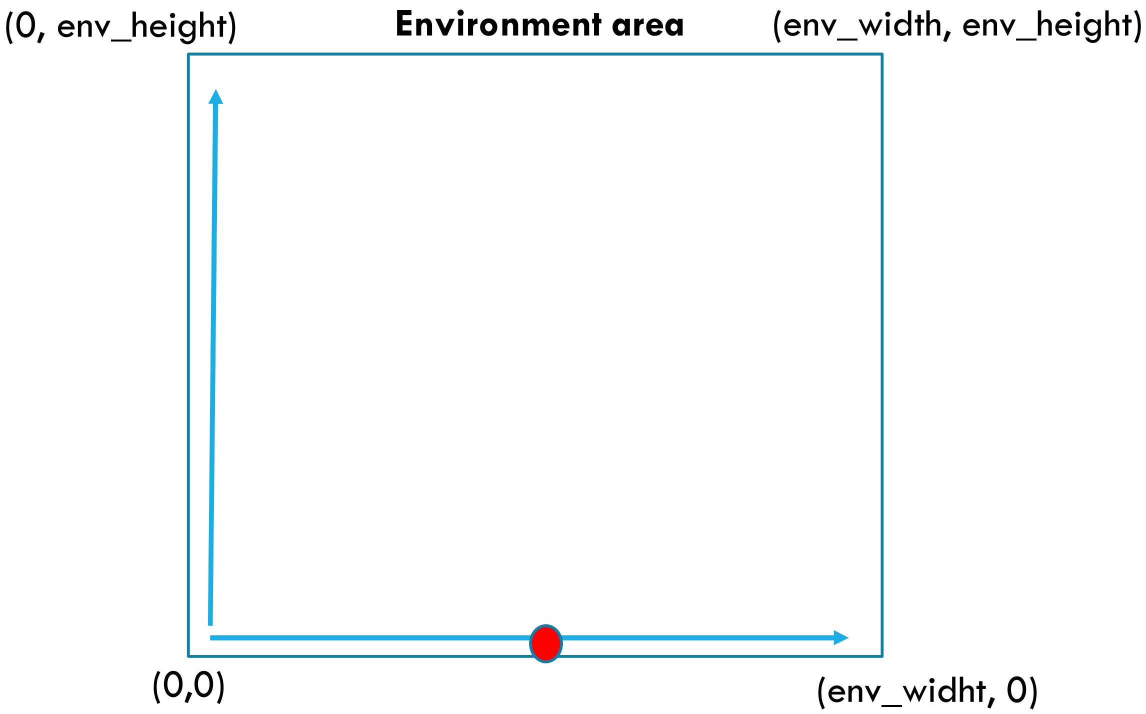
Total len simulation: 15000 * 0.15 sec = 2250 seconds

AREA AND PATHS

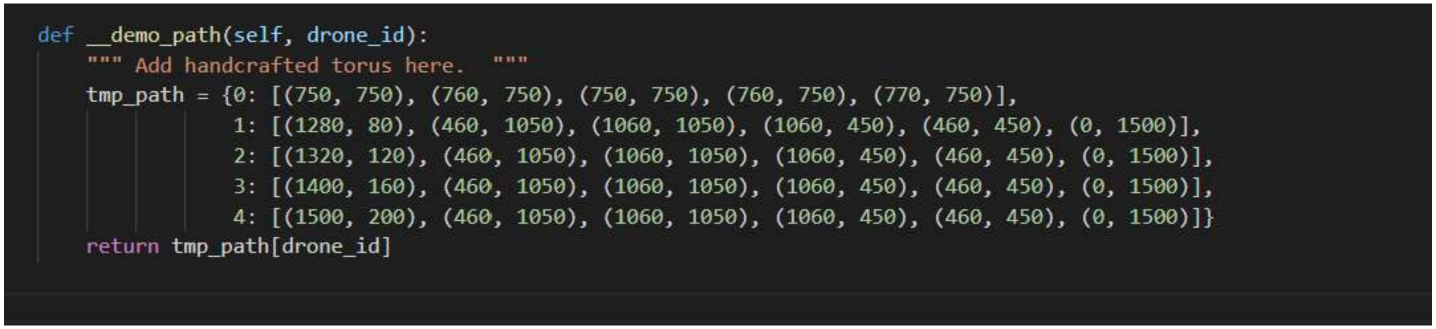
src.utilities.config



```
DEPOT_COO = (750, 0)
```



src.utilities.utilities.PathManager.__demo_path(self, drone_id):

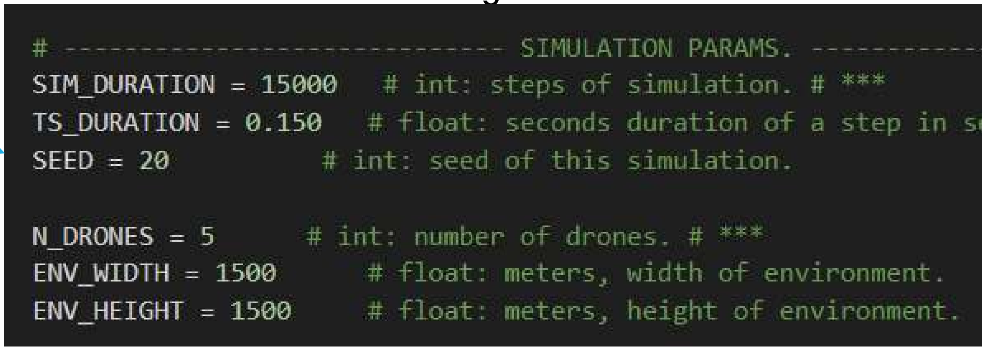


```
def __demo_path(self, drone_id):
    """ Add handcrafted torus here.  """
    tmp_path = {0: [(750, 750), (760, 750), (750, 750), (760, 750), (770, 750)],
                1: [(1280, 80), (460, 1050), (1060, 1050), (1060, 450), (460, 450), (0, 1500)],
                2: [(1320, 120), (460, 1050), (1060, 1050), (1060, 450), (460, 450), (0, 1500)],
                3: [(1400, 160), (460, 1050), (1060, 1050), (1060, 450), (460, 450), (0, 1500)],
                4: [(1500, 200), (460, 1050), (1060, 1050), (1060, 450), (460, 450), (0, 1500)]}
    return tmp_path[drone_id]
```

SIMULATION - SEED

SEED!

src.utilities.config



```
# ----------------------------- SIMULATION PARAMS. -----------
SIM_DURATION = 15000   # int: steps of simulation. # ***
TS_DURATION = 0.150   # float: seconds duration of a step in s
SEED = 20          # int: seed of this simulation.

N_DRONES = 5      # int: number of drones. # ***
ENV_WIDTH = 1500      # float: meters, width of environment.
ENV_HEIGHT = 1500     # float: meters, height of environment.
```

Seed function is used to save the state of a random function, so that it can generate same random numbers on multiple executions of the code on the same machine or on different machines (for a specific seed value).

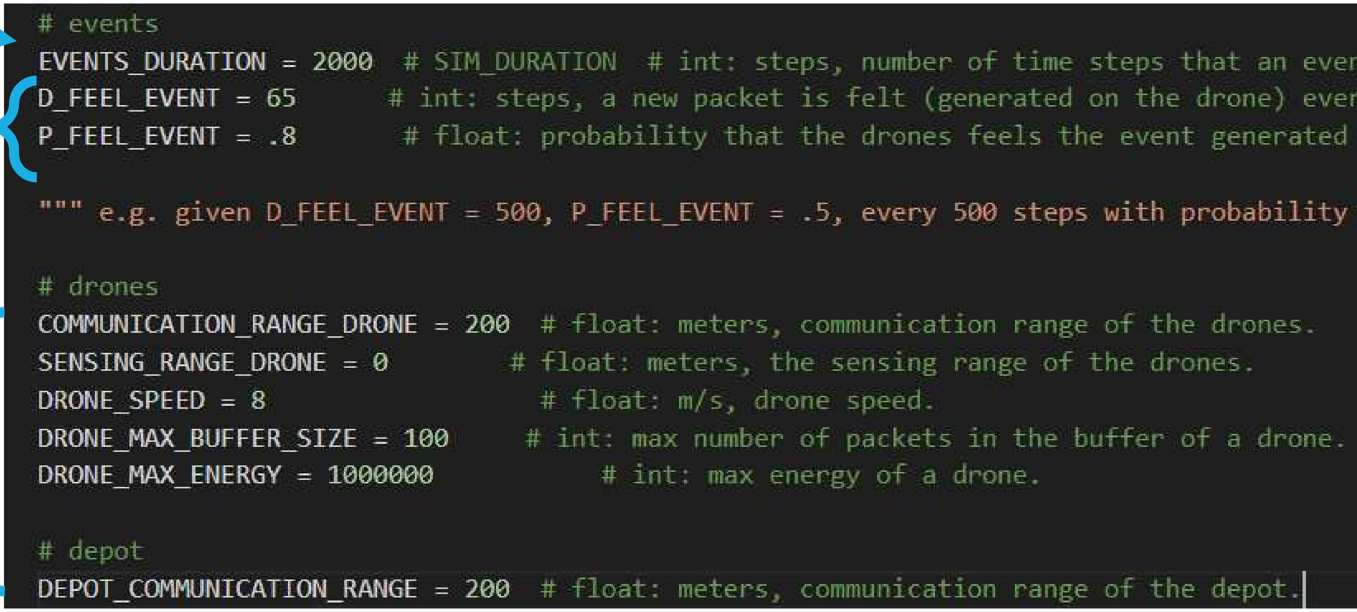
- Reproducibility of experiments
- Debug
- Deterministic behavior

CONFIG – HOW TO CHANGE SIMULATION PARAMETERS

A new event/packet of a drone lasts for EVENTS_DURATION

Parameters for the generation of new events

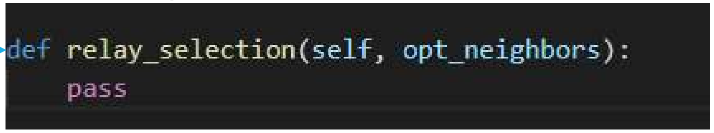
Drones and depot capabilities



```
# events
EVENTS_DURATION = 2000  # SIM_DURATION  # int: steps, number of time steps that an ever
D_FEEL_EVENT = 65      # int: steps, a new packet is felt (generated on the drone) ever
P_FEEL_EVENT = .8       # float: probability that the drones feels the event generated

""" e.g. given D_FEEL_EVENT = 500, P_FEEL_EVENT = .5, every 500 steps with probability

# drones
COMMUNICATION_RANGE_DRONE = 200  # float: meters, communication range of the drones.
SENSING_RANGE_DRONE = 0         # float: meters, the sensing range of the drones.
DRONE_SPEED = 8                  # float: m/s, drone speed.
DRONE_MAX_BUFFER_SIZE = 100     # int: max number of packets in the buffer of a drone.
DRONE_MAX_ENERGY = 1000000          # int: max energy of a drone.

# depot
DEPOT_COMMUNICATION_RANGE = 200  # float: meters, communication range of the depot.
```

HOW TO CREATE AND SELECT A ROUTING PROTOCOL?

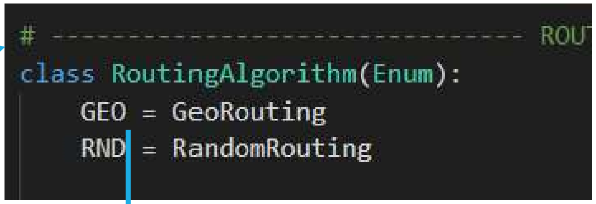
1) Create a new routing protocol in **src.routing_algorithms** extending **BASE_routing class**
2) Implement the **relay_selection** method



```
def relay_selection(self, opt_neighbors):
    pass
```

3) Import the new routing protocol in src.utilities.config file.

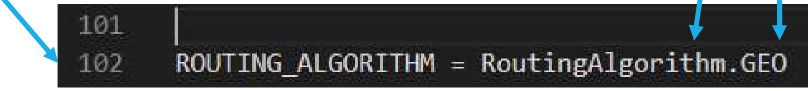
4) Add the routing protocol to the src.utilities.config.RoutingAlgorithm Enum



```
# ---------------------------------- ROU
class RoutingAlgorithm(Enum):
    GEO = GeoRouting
    RND = RandomRouting
```

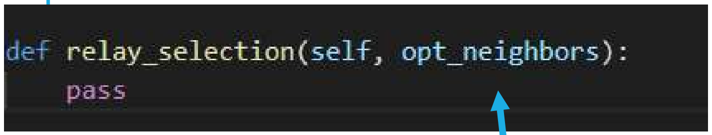
5) Select it:

src.utilities.config.ROUTING_ALGORITHM = src.utilities.config.RoutingAlgorithm .new_routing_protocol



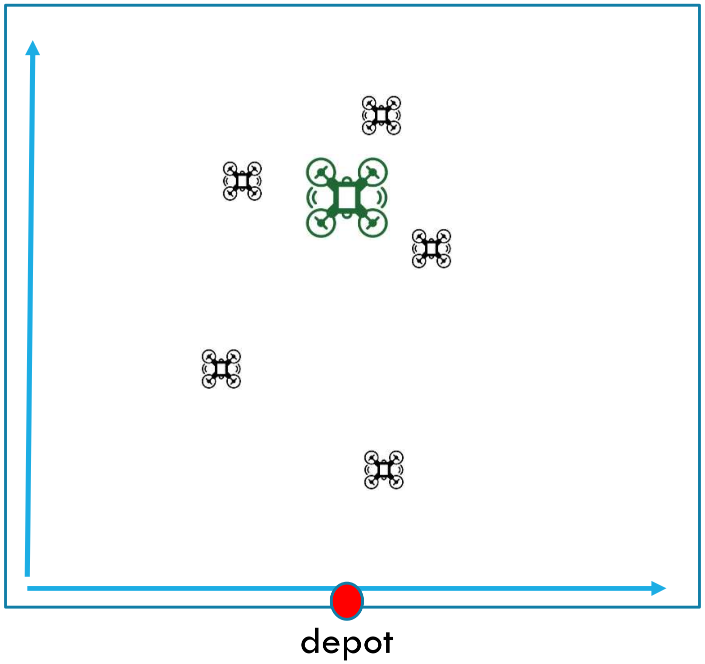
```

ROUTING_ALGORITHM = RoutingAlgorithm.GEO
```

HOW TO CREATE A ROUTING PROTOCOL - CONTINUE

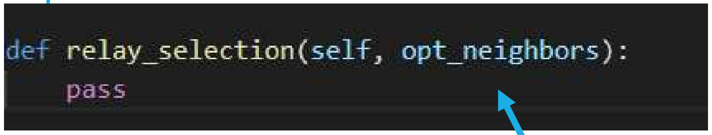


```
def relay_selection(self, opt_neighbors):
    pass
```

Who are my neighbors?



HOW TO CREATE A ROUTING PROTOCOL - CONTINUE

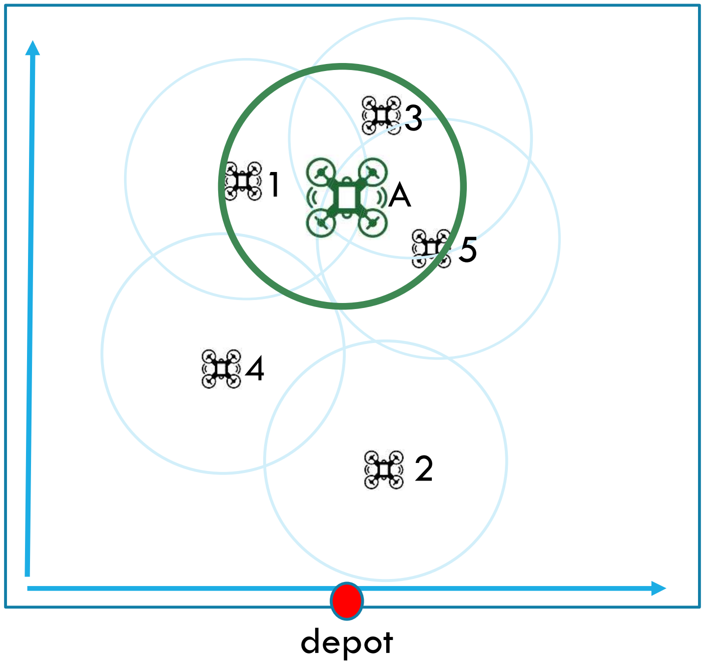


```
def relay_selection(self, opt_neighbors):
    pass
```

Who are my neighbors?

- **All the drones in my communication range!**

Drone A has: **1,3,5** as neigh drones

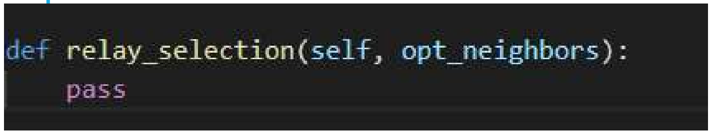


How drone A know its neighboorhood?

Each **src.utilities.config. HELLO_DELAY** a drone sends a “Hello Message”

When **drone A** receives the hello message it stores in **self.hello_messages : {drone : last_hello_message)**

HOW TO CREATE A ROUTING PROTOCOL - CONTINUE

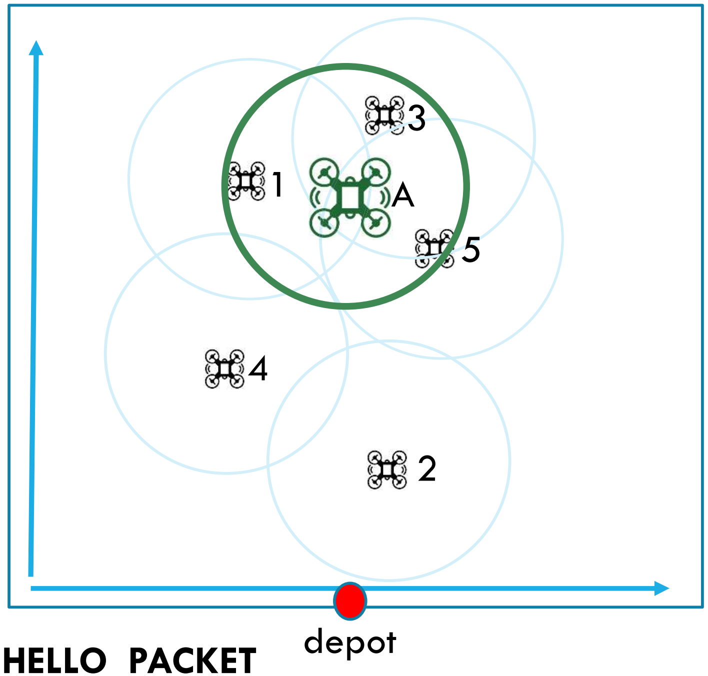


```
def relay_selection(self, opt_neighbors):
    pass
```

A drone, at time t, knows its neighbors by looking at this dictionary:
self.hello_messages : {drone : last_hello_message)

An hello message has several info (fields):

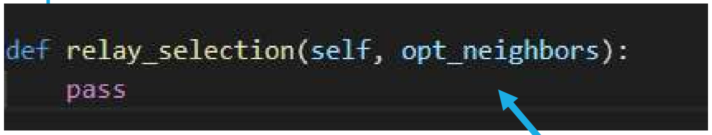
-src_drone
-time_step_creation
-cur_pos (at time_step_creation)
-speed (at time_step_creation)
-next_target (at time_step_creation)



In particular, a drone X is my neighbor, if I have a recent hello message s.t.:
if **self.hello_messages[X]**.time_step_creation > t - **config.OLD_HELLO_PACKET**

Define the threshold to still consider a hello message as valid

HOW TO CREATE A ROUTING PROTOCOL - CONTINUE



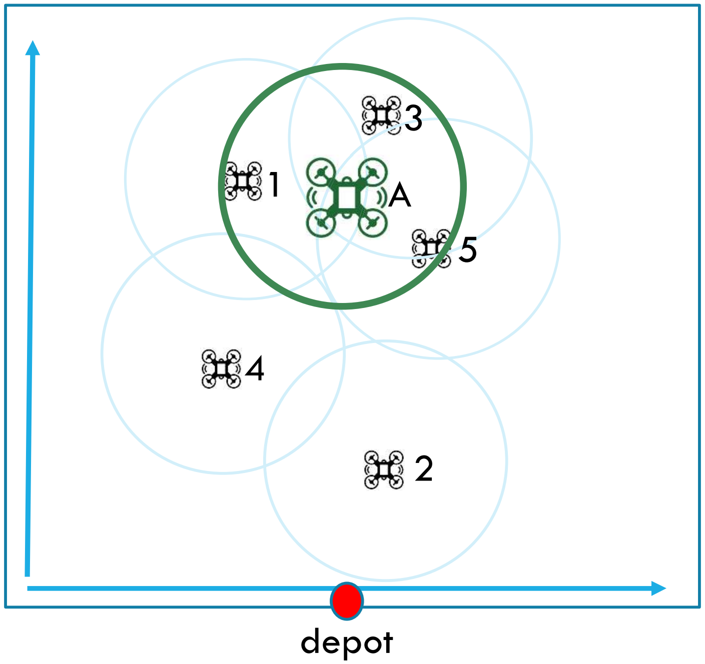
```python
def relay_selection(self, opt_neighbors):
    pass
```

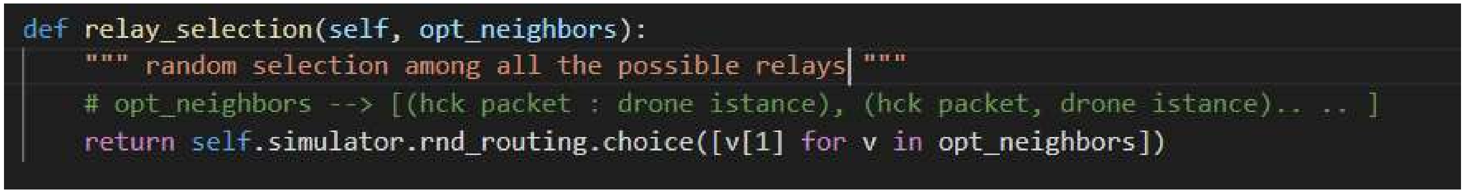
What are opt_neighbors?

A list of [(hello_message1, drone1), (hello_message2, drone2), …..]

The goal of the method:

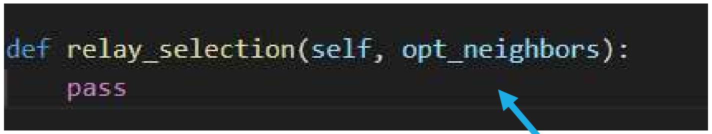
-Return a drone, s.t., base on the field on its hello_message is the most suitable to delivery a packet vs the depot.





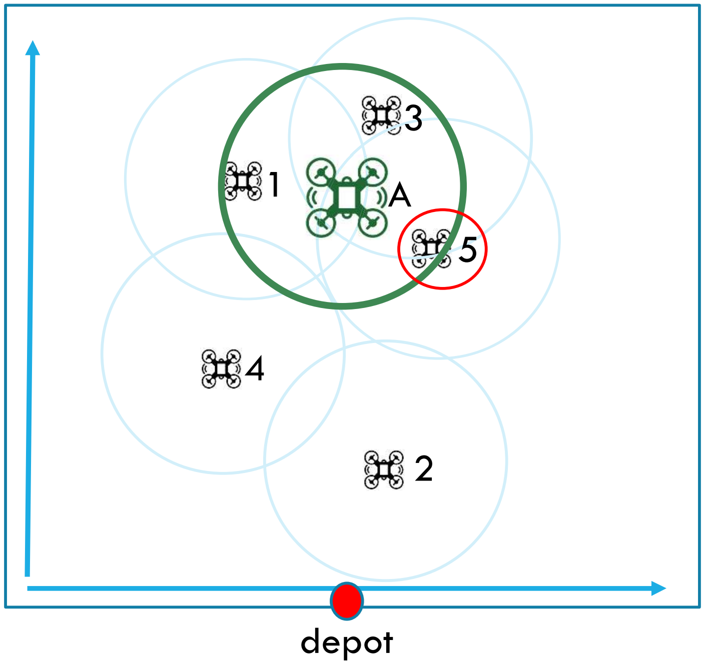
```python
def relay_selection(self, opt_neighbors):
    """ random selection among all the possible relays """
    # opt_neighbors --> [(hck packet : drone istance), (hck packet, drone istance).. .. ]
    return self.simulator.rnd_routing.choice([v[1] for v in opt_neighbors])
```

NEXT TIME – GEOGRAPHICAL ROUTING



```
def relay_selection(self, opt_neighbors):
    pass
```

Select the closest drone to the destination!



Implement **src.routing_algorithms.georouting** script

HOW TO RUN

Inside the project directory (but outside src) type:

```
python3 –m src.main
```

CONTACTS

Andrea Coletta:

coletta@di.uniroma1.it