SAPIENZA
UNIVERSITÀ DI ROMA

DroNET Simulator

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

Speaker: *Dr. Andrea Coletta - 06-11-2020*

DISPLAY – SAVE THE SIMULATION

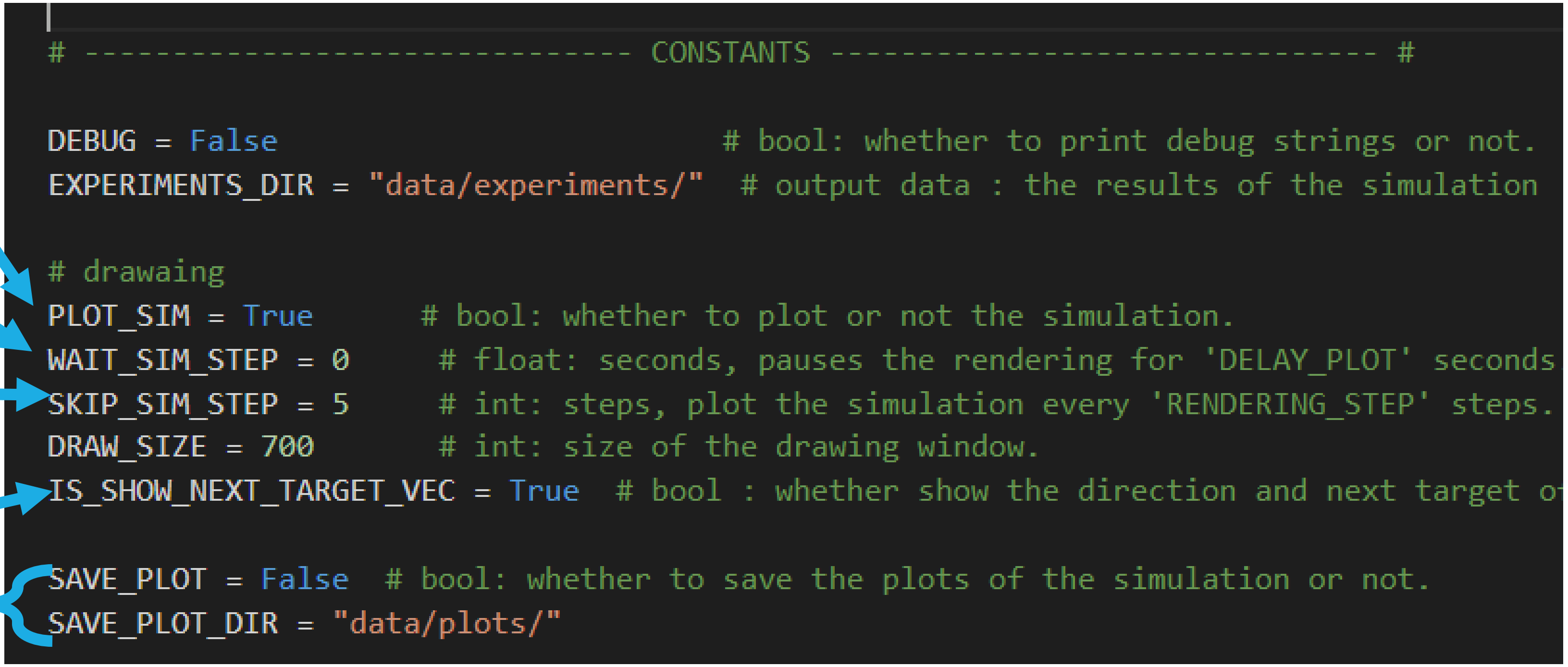
Display the simulation in real time

How may seconds wait between different frames (to slow down simulation)

How many frames skip in the visualization. 5 -> 1/5 of frames are shown

It shows where the drones are going

Save screenshot of the frames in the directory data/plots/ (be sure to create this directory).



```
# ------------------------------ CONSTANTS ------------------------------ #

DEBUG = False                               # bool: whether to print debug strings or not.
EXPERIMENTS_DIR = "data/experiments/"  # output data : the results of the simulation

# drawaing
PLOT_SIM = True      # bool: whether to plot or not the simulation.
WAIT_SIM_STEP = 0     # float: seconds, pauses the rendering for 'DELAY_PLOT' seconds
SKIP_SIM_STEP = 5     # int: steps, plot the simulation every 'RENDERING_STEP' steps.
DRAW_SIZE = 700       # int: size of the drawing window.
IS_SHOW_NEXT_TARGET_VEC = True  # bool : whether show the direction and next target o

SAVE_PLOT = False  # bool: whether to save the plots of the simulation or not.
SAVE_PLOT_DIR = "data/plots/"
```

For big/fast simulations disable (False) PLOT_SIM and SAVE_PLOT.

METRICS – ALGORITHM PERFORMANCE

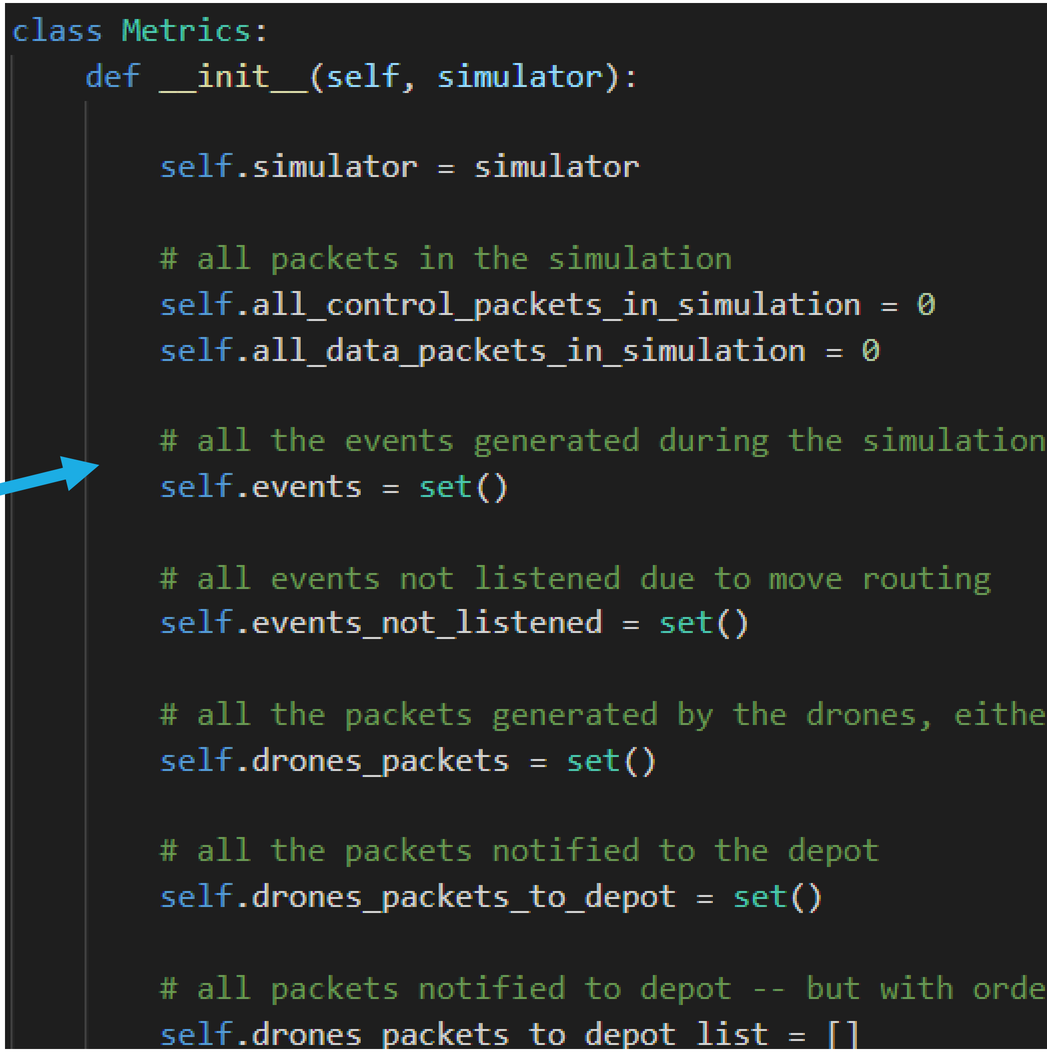
At the end of simulation, the code prints some statistics:

Src.simulation.simulator.Simulator.print_metrics()

And it saves a json with all the simulation info in:

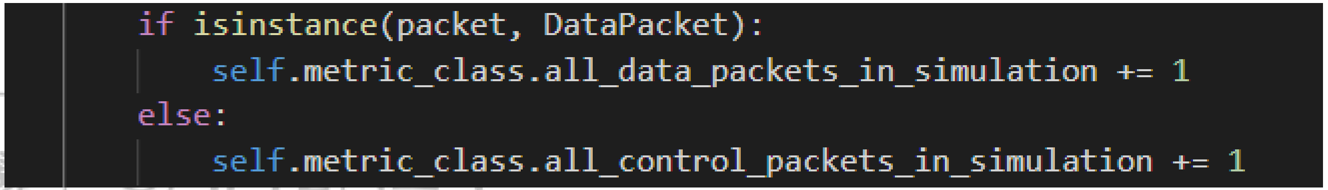
*ROOT_EVALUATION_DATA = **"data/evaluation_tests/"***

The class metric contains several list and counters, that are accessible from «simulator.metrics….» which allow to trace everthing in the simulation. (You can add more fields if needed)



```
class Metrics:
    def __init__(self, simulator):

        self.simulator = simulator

        # all packets in the simulation
        self.all_control_packets_in_simulation = 0
        self.all_data_packets_in_simulation = 0

        # all the events generated during the simulation
        self.events = set()

        # all events not listened due to move routing
        self.events_not_listened = set()

        # all the packets generated by the drones, eithe
        self.drones_packets = set()

        # all the packets notified to the depot
        self.drones_packets_to_depot = set()

        # all packets notified to depot -- but with orde
        self.drones_packets_to_depot_list = []
```

Example in the medium simulator



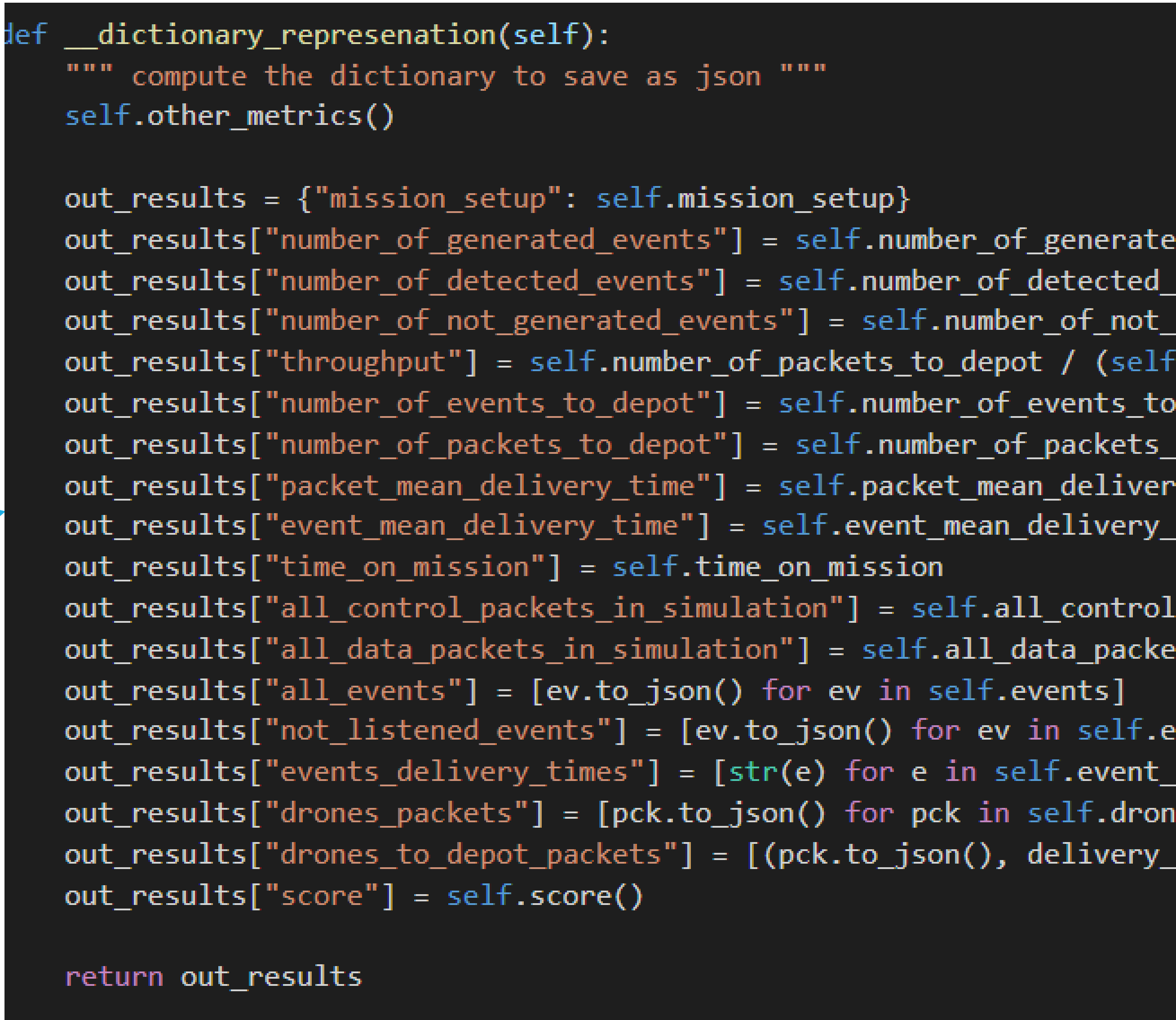
```
if isinstance(packet, DataPacket):
    self.metric_class.all_data_packets_in_simulation += 1
else:
    self.metric_class.all_control_packets_in_simulation += 1
```

METRICS – ALGORITHM PERFORMANCE

And it saves a json with all the simulation info in:

ROOT_EVALUATION_DATA = "data/evaluation_tests/"

At the end of simulation, all the needed fields can be stored in the output json, using **out_results** inside this method.



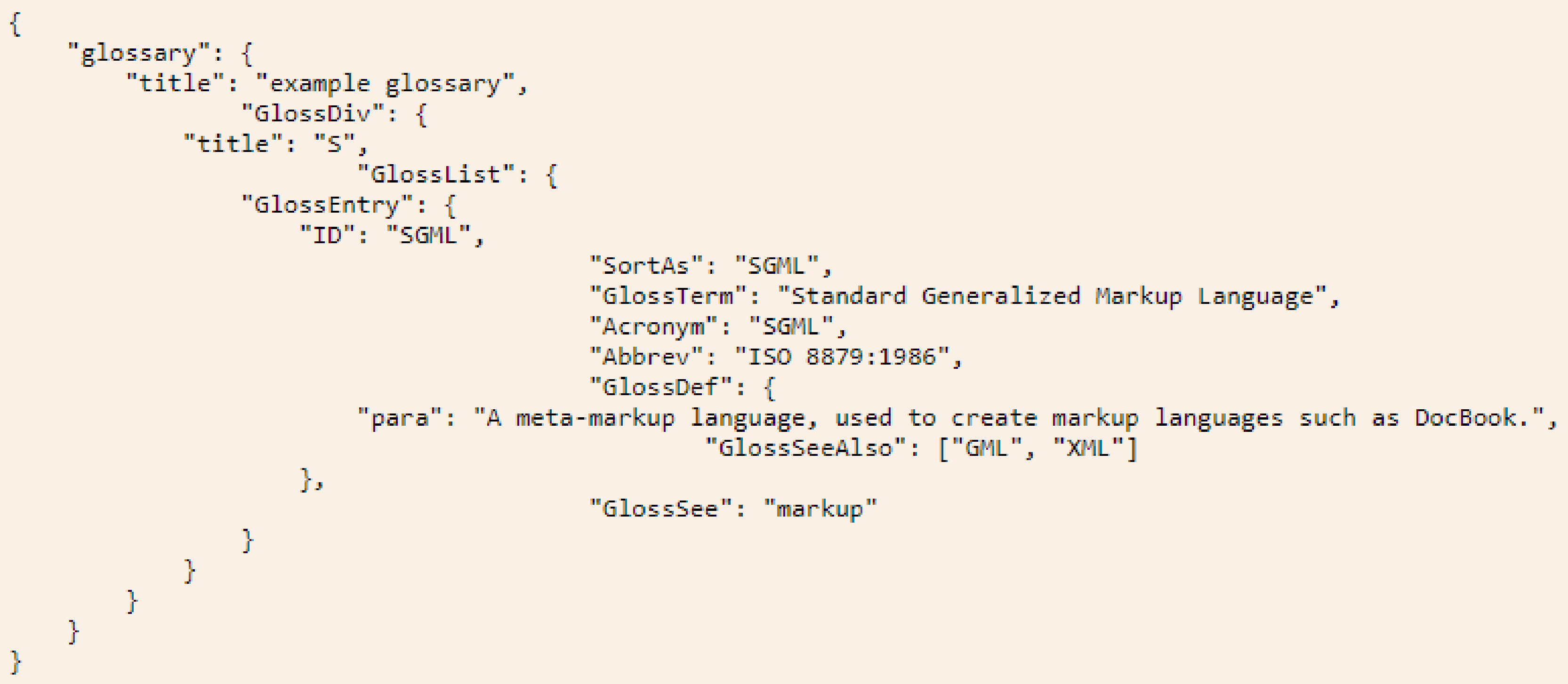
```
def __dictionary_represenation(self):
    """ compute the dictionary to save as json """
    self.other_metrics()

    out_results = {"mission_setup": self.mission_setup}
    out_results["number_of_generated_events"] = self.number_of_generate
    out_results["number_of_detected_events"] = self.number_of_detected_
    out_results["number_of_not_generated_events"] = self.number_of_not_
    out_results["throughput"] = self.number_of_packets_to_depot / (self
    out_results["number_of_events_to_depot"] = self.number_of_events_to
    out_results["number_of_packets_to_depot"] = self.number_of_packets_
    out_results["packet_mean_delivery_time"] = self.packet_mean_deliver
    out_results["event_mean_delivery_time"] = self.event_mean_delivery_
    out_results["time_on_mission"] = self.time_on_mission
    out_results["all_control_packets_in_simulation"] = self.all_control
    out_results["all_data_packets_in_simulation"] = self.all_data_packe
    out_results["all_events"] = [ev.to_json() for ev in self.events]
    out_results["not_listened_events"] = [ev.to_json() for ev in self.e
    out_results["events_delivery_times"] = [str(e) for e in self.event_
    out_results["drones_packets"] = [pck.to_json() for pck in self.dron
    out_results["drones_to_depot_packets"] = [(pck.to_json(), delivery_
    out_results["score"] = self.score()

    return out_results
```

METRICS – ALGORITHM PERFORMANCE

And it saves a json with all the simulation info in:

ROOT_EVALUATION_DATA = ***"data/evaluation_tests/"***

Json format like below:



```
{
    "glossary": {
        "title": "example glossary",
                "GlossDiv": {
            "title": "S",
                        "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                                        "SortAs": "SGML",
                                        "GlossTerm": "Standard Generalized Markup Language",
                                        "Acronym": "SGML",
                                        "Abbrev": "ISO 8879:1986",
                                        "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                                "GlossSeeAlso": ["GML", "XML"]
                    },
                                        "GlossSee": "markup"
                }
            }
        }
    }
}
```

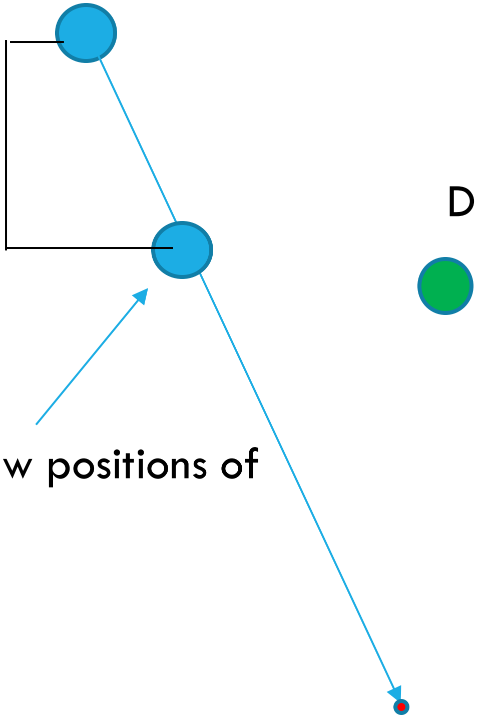
HOW TO RUN

Inside the project directory (but outside src) type:

```
python3 –m src.main
```

GEOGRAPHICAL ROUTING

Drone that sends a message a time t:
Hello pck generation (src_drone, position, cur_time, t, speed)



Elapsed time:
t' – t

Drone that communicates at time t'

Estimated new positions of
the relay

CONTACTS

Andrea Coletta:

coletta@di.uniroma1.it