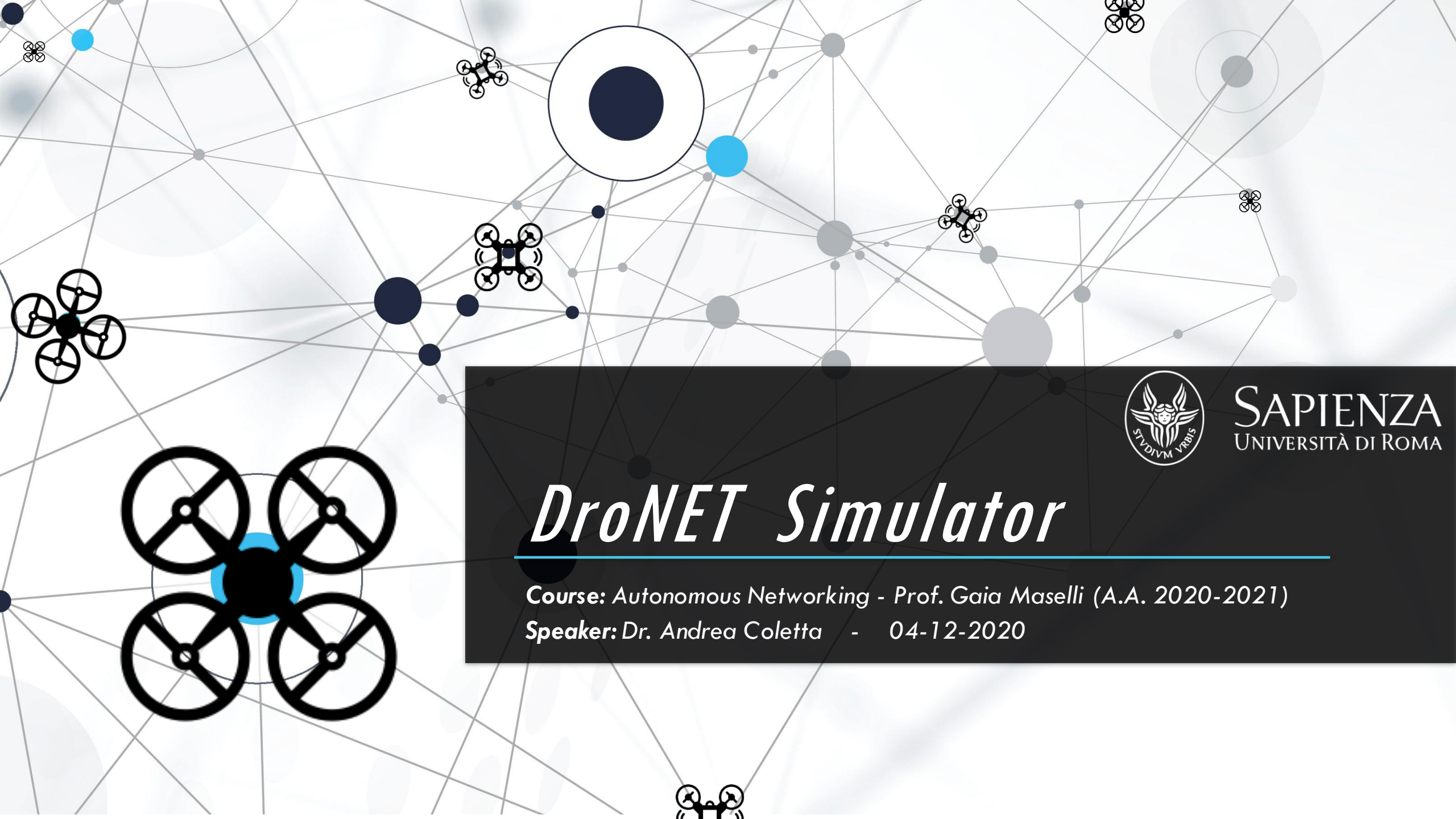


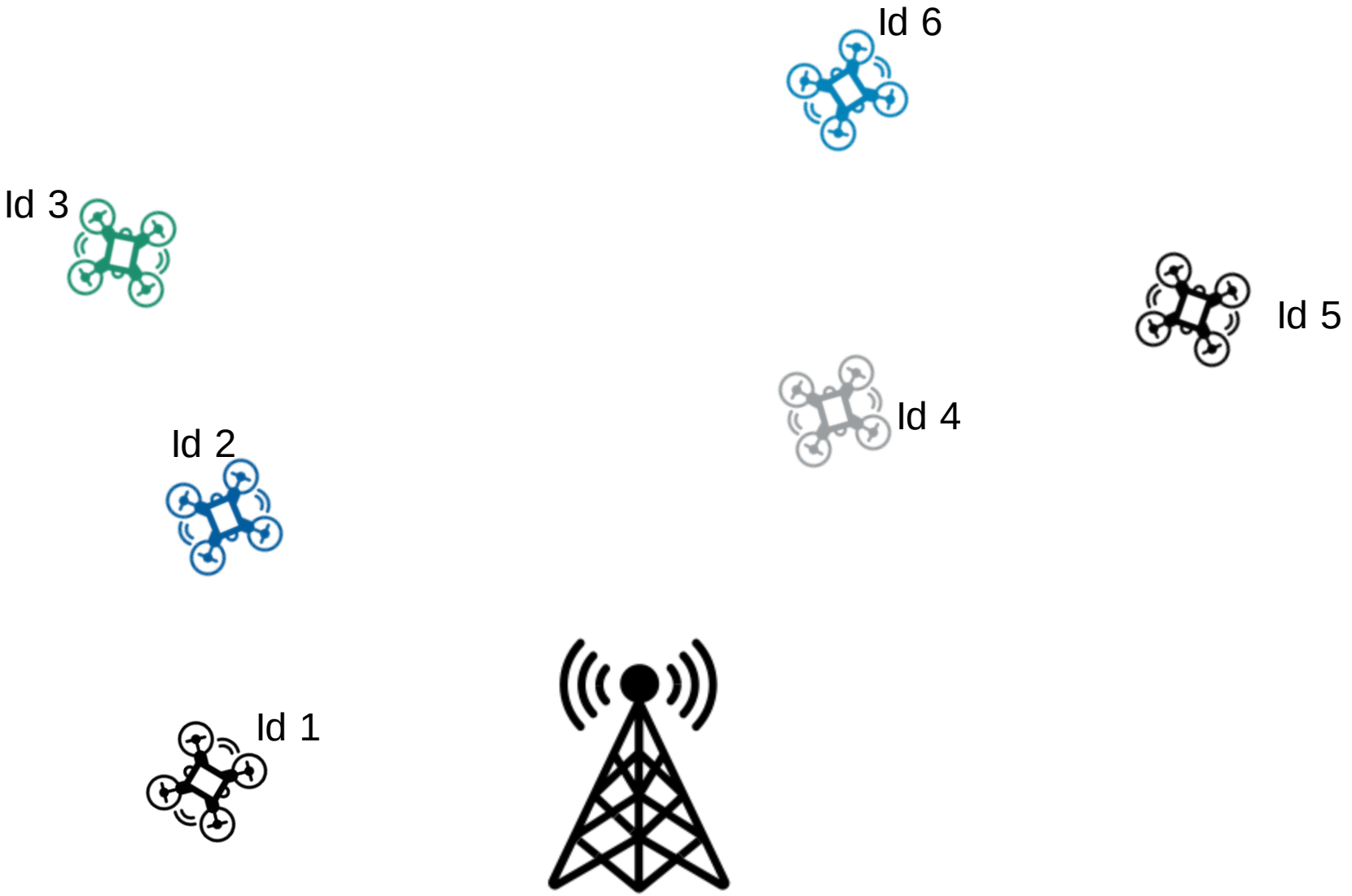
DroNET Simulator

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

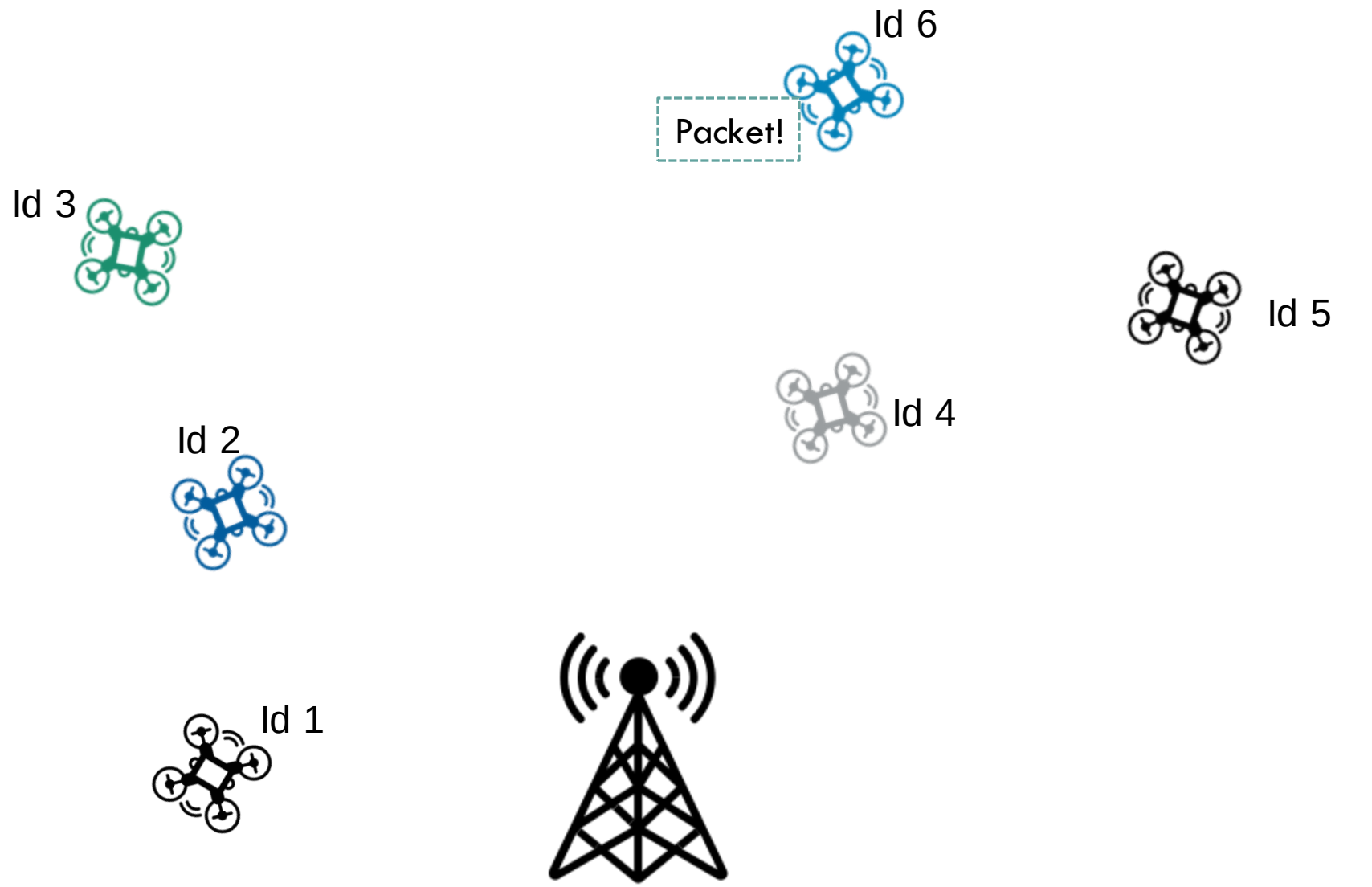
Speaker: *Dr. Andrea Coletta - 04-12-2020*

SAPIENZA
UNIVERSITÀ DI ROMA

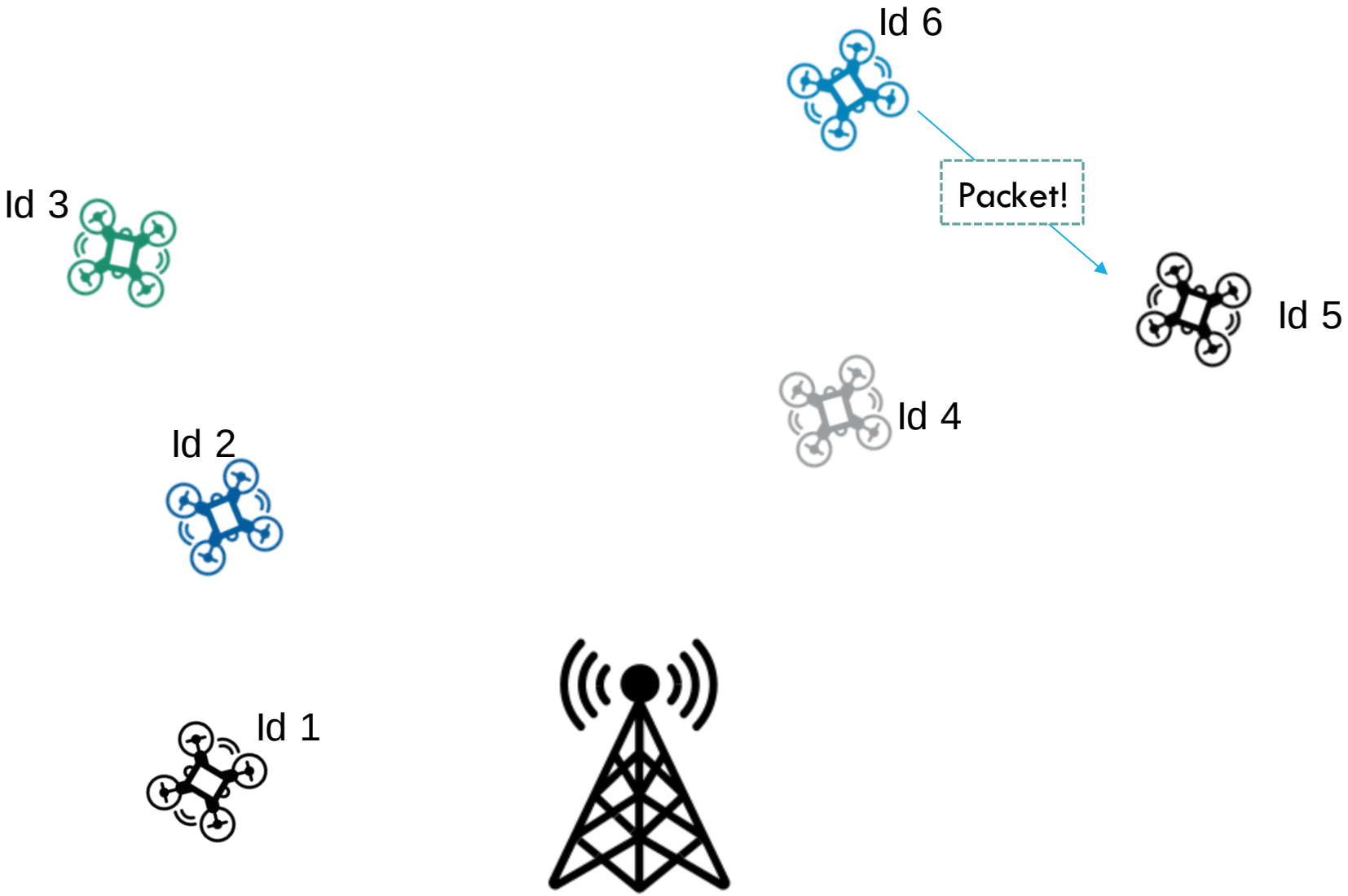
HOMEWORK 2 - ROUTING



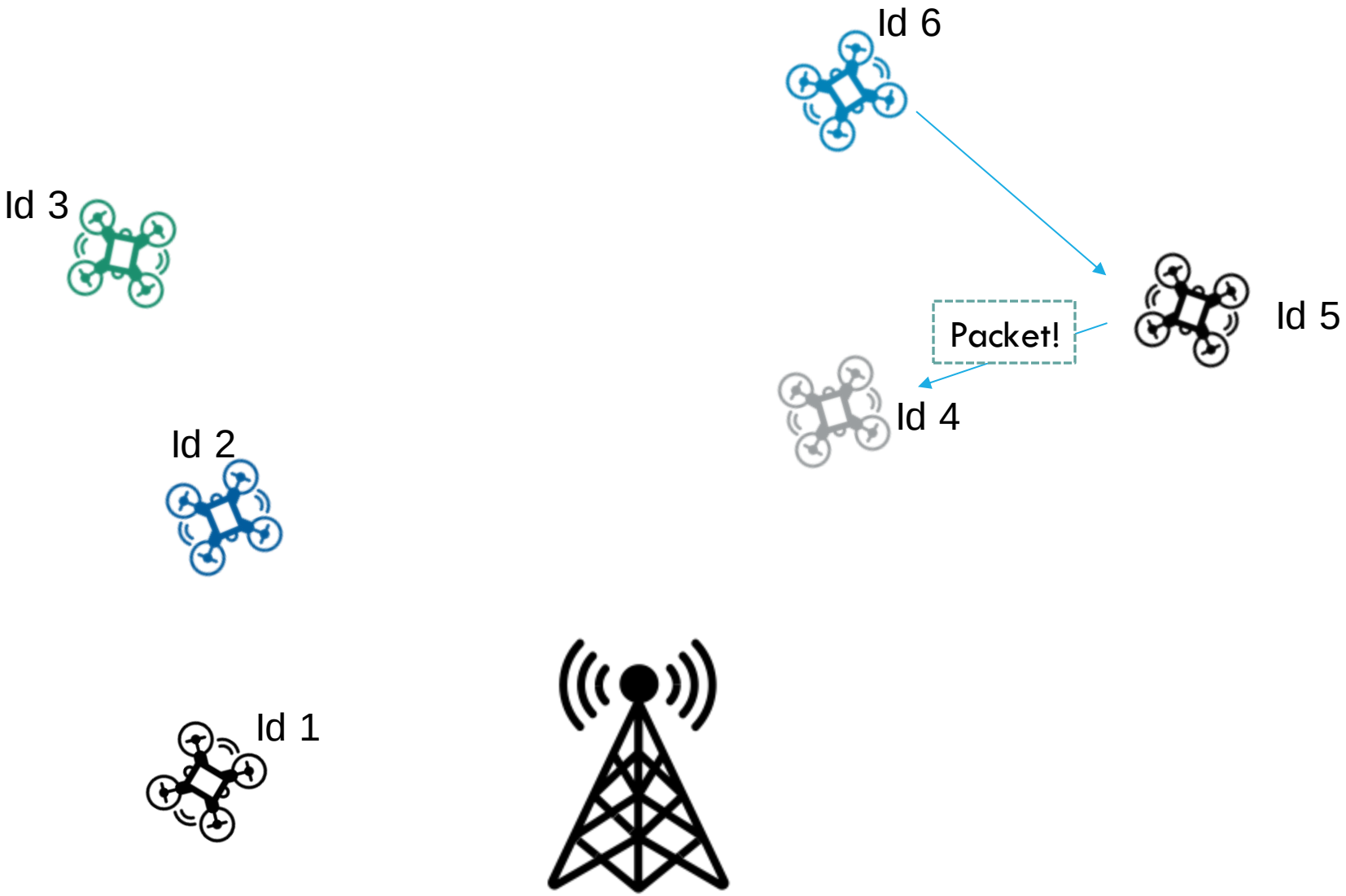
HOMEWORK 2 - ROUTING



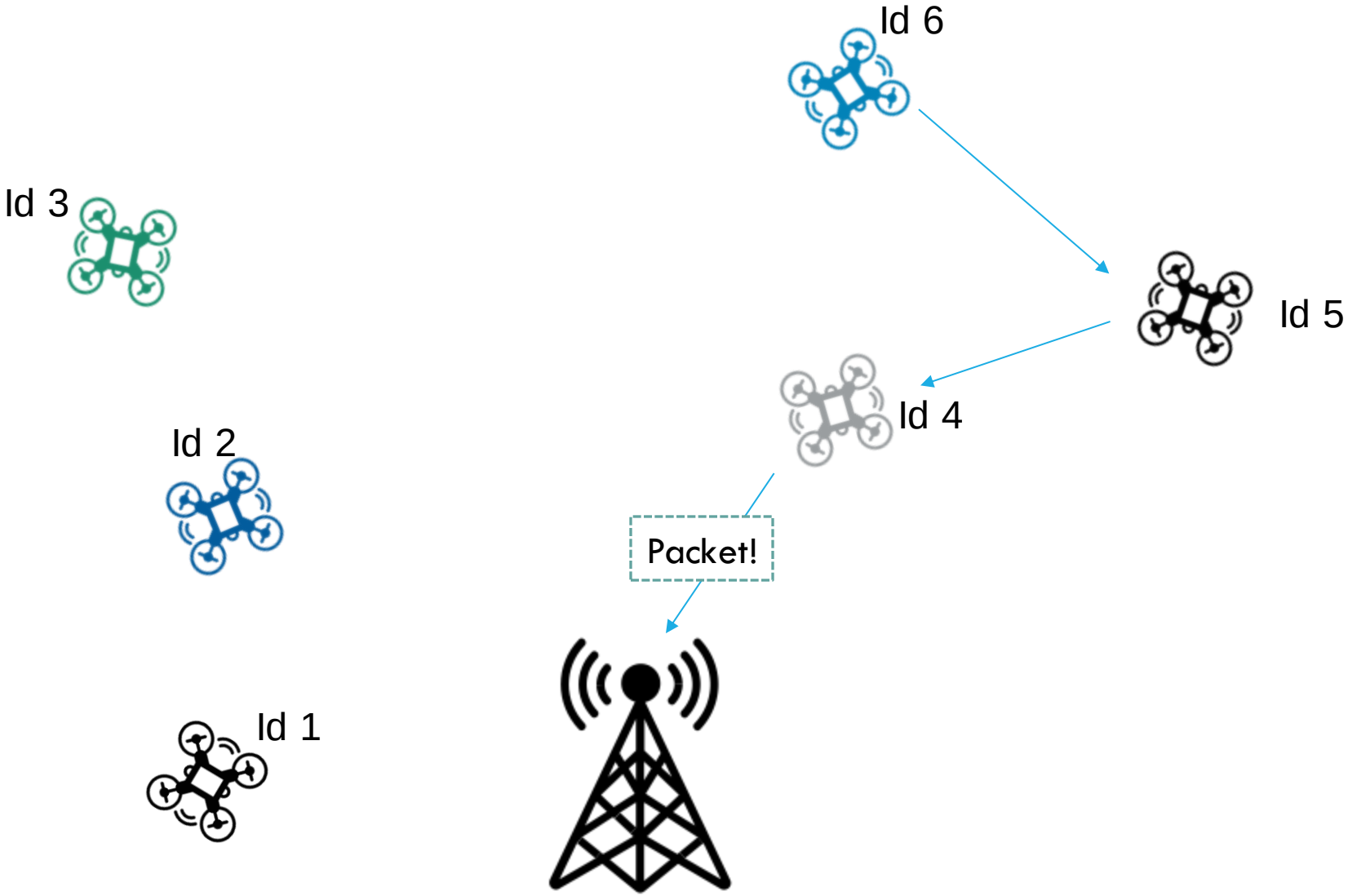
HOMEWORK 2 - ROUTING



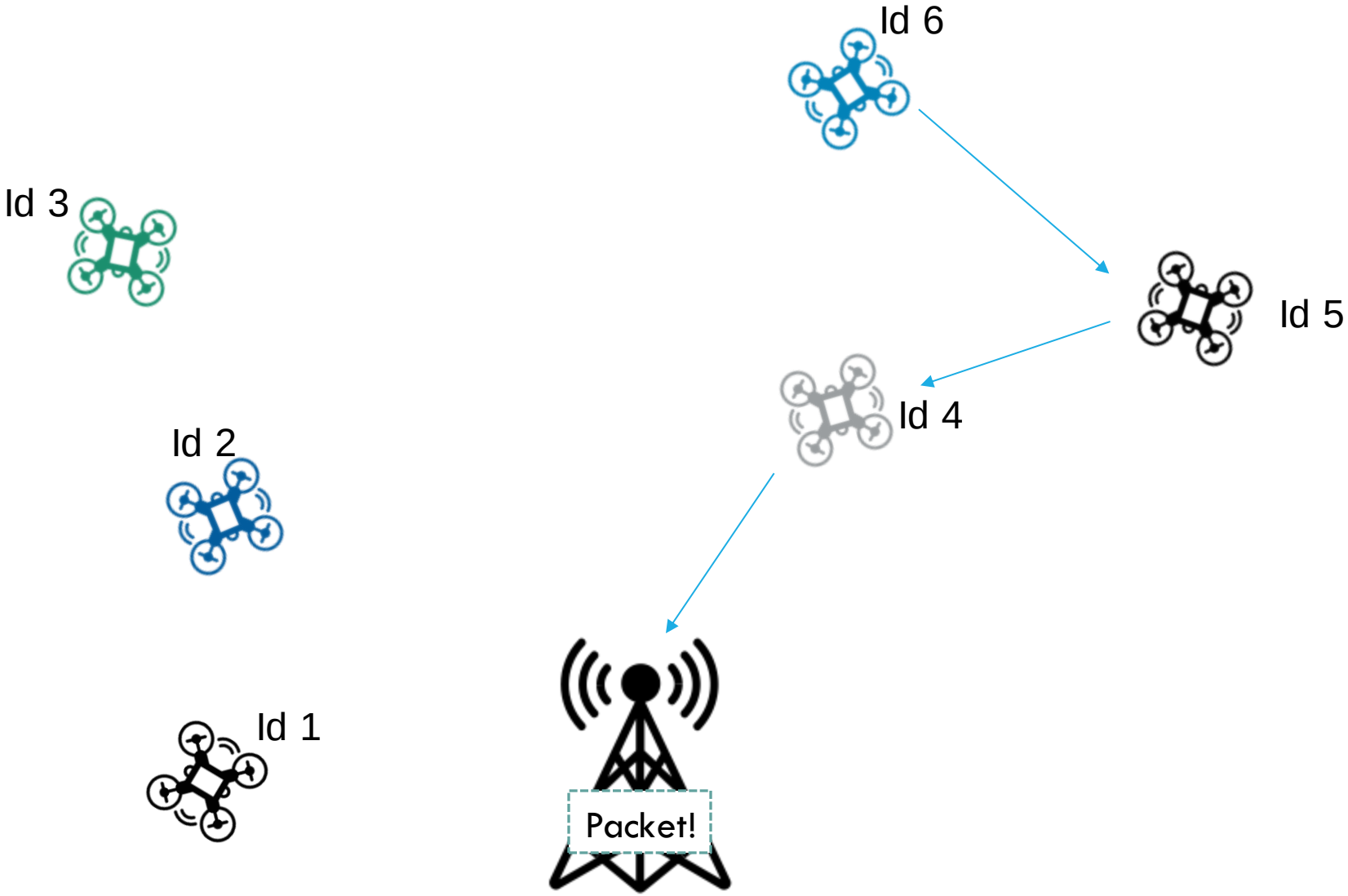
HOMEWORK 2 - ROUTING



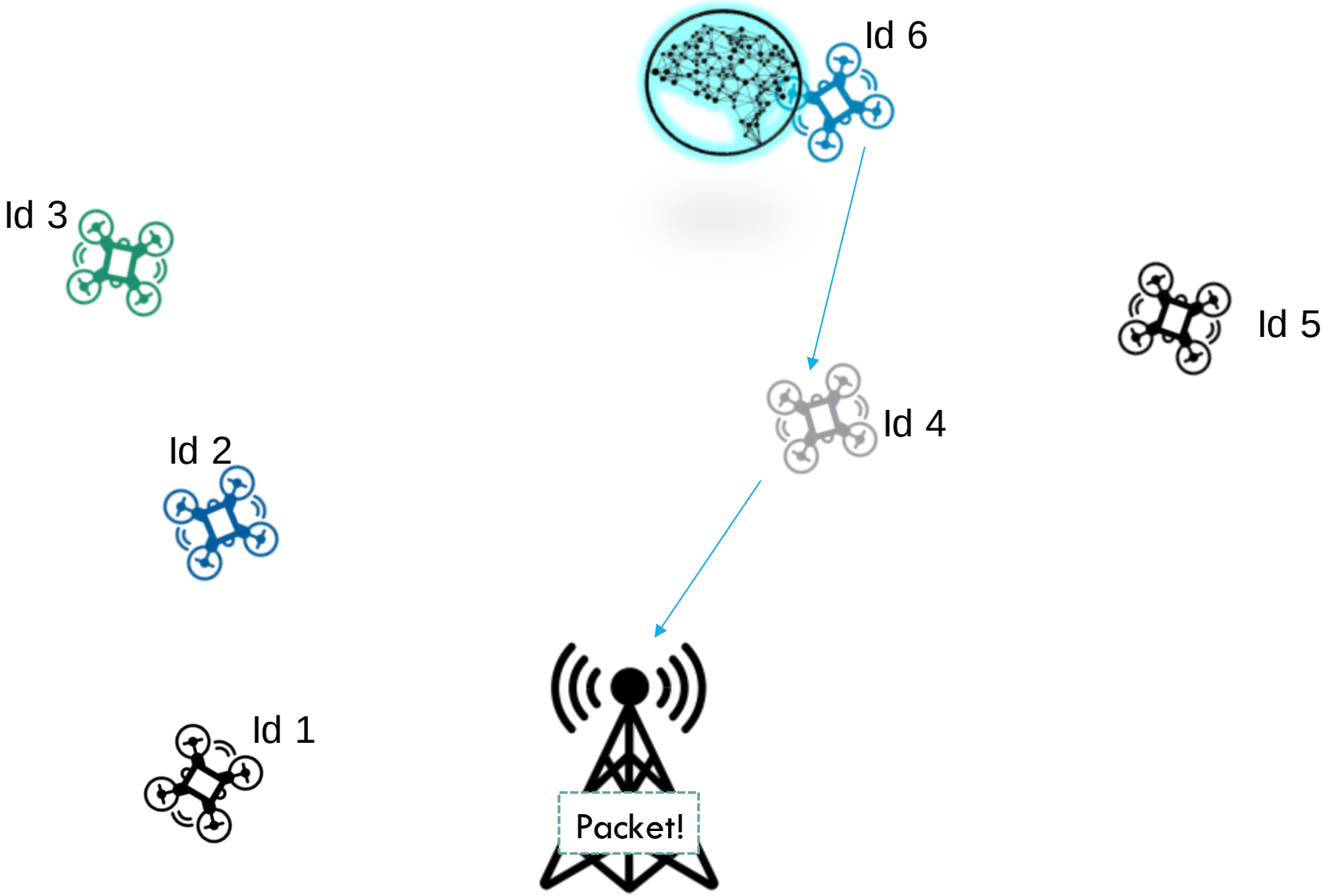
HOMEWORK 2 - ROUTING



HOMEWORK 2 - ROUTING

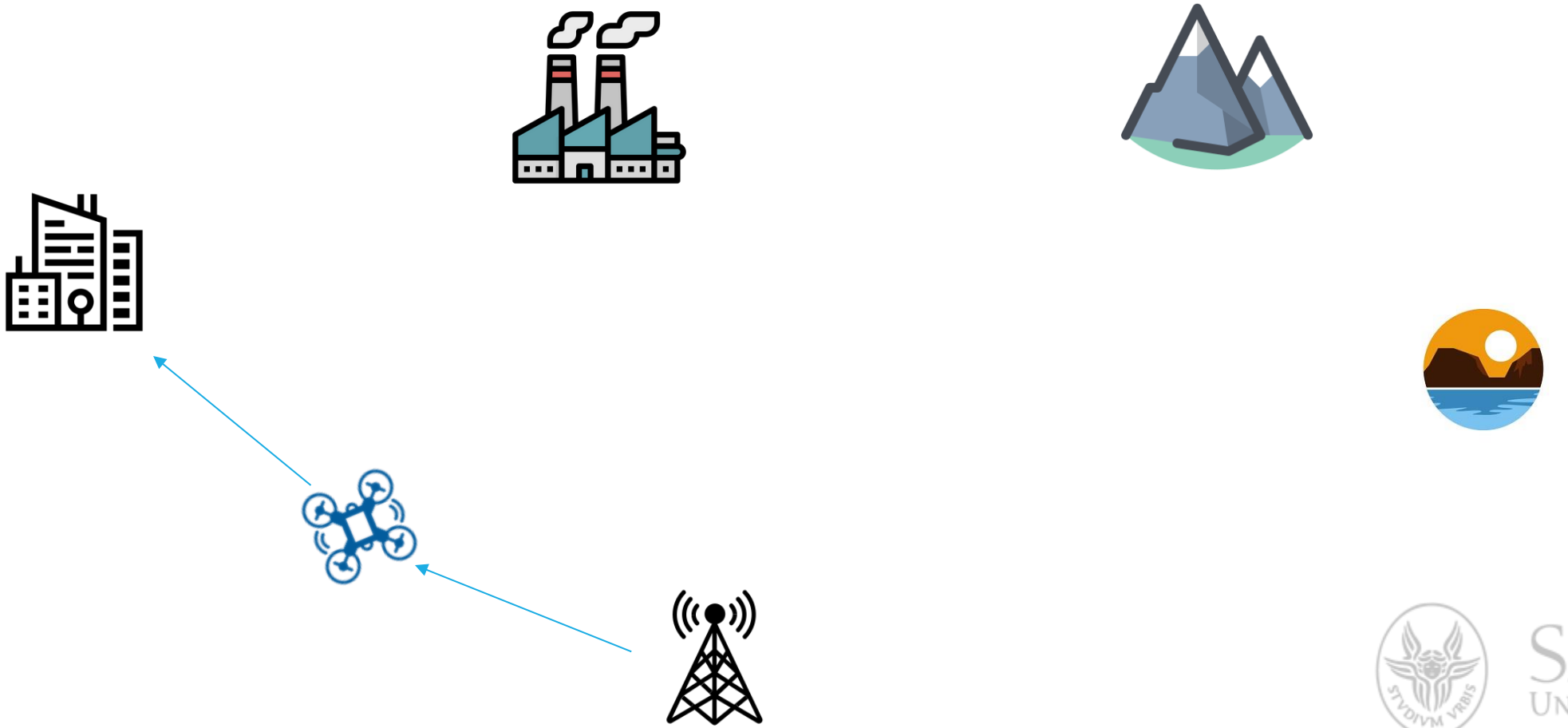


HOMEWORK 2 – REINFORCEMENT LEARNING ROUTING



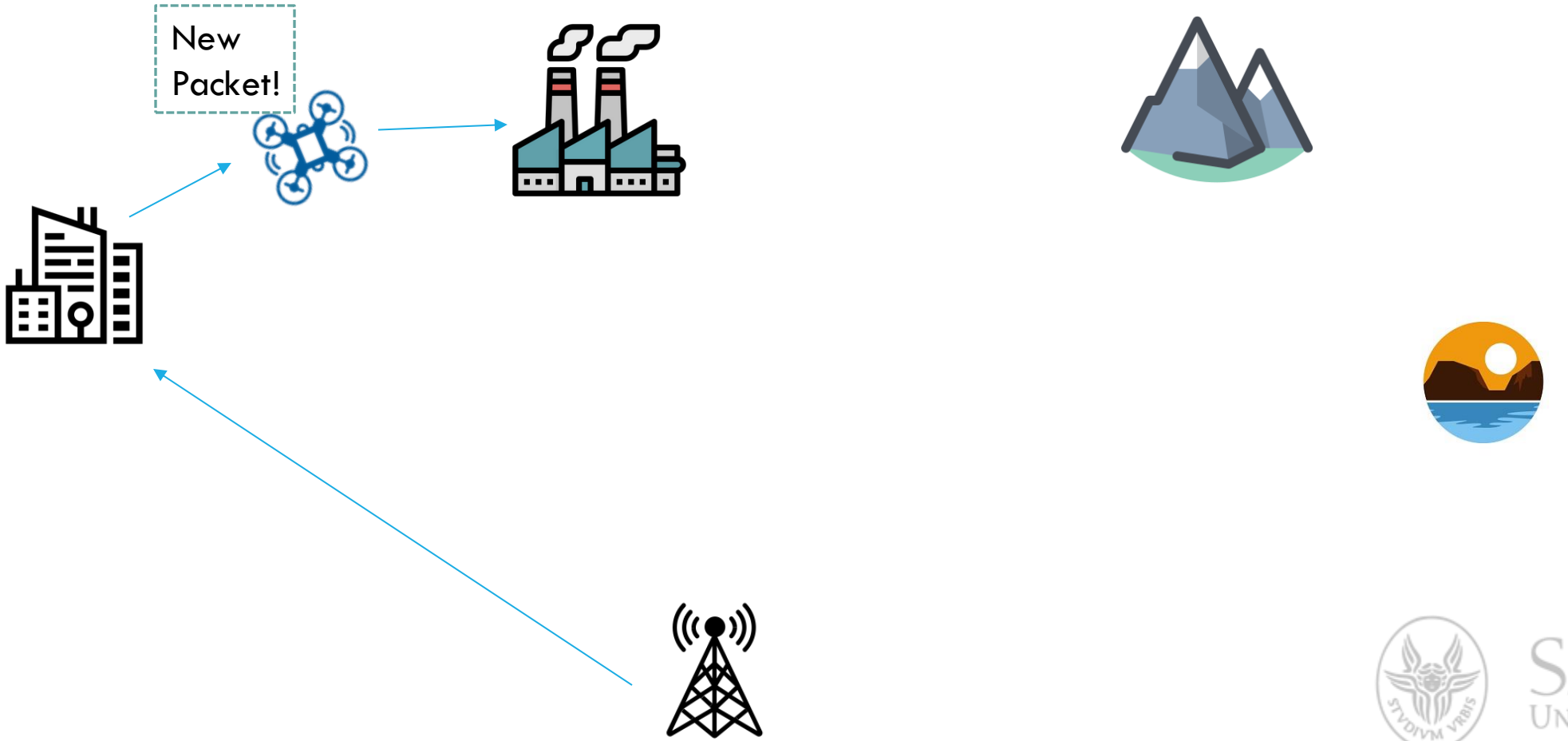
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



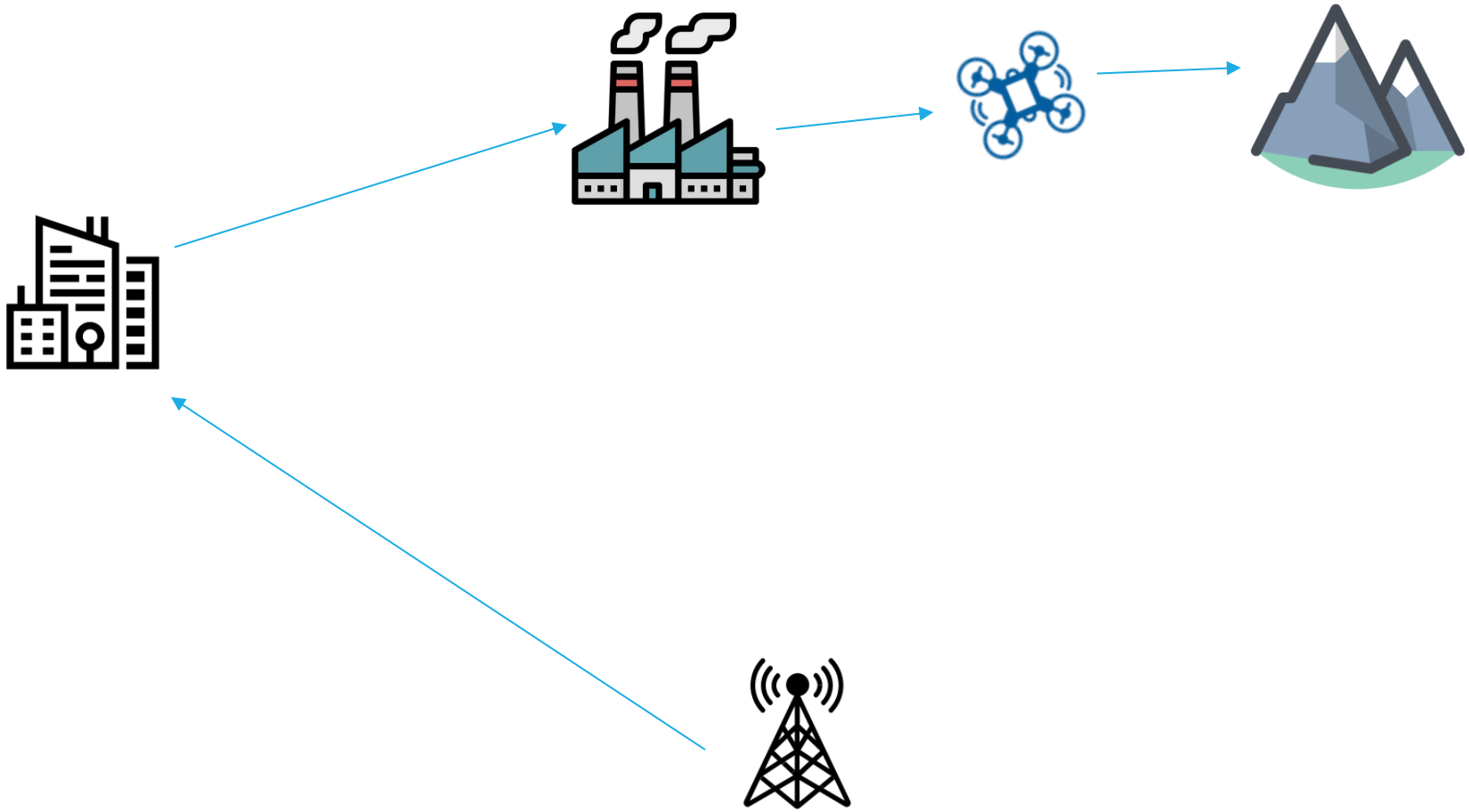
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



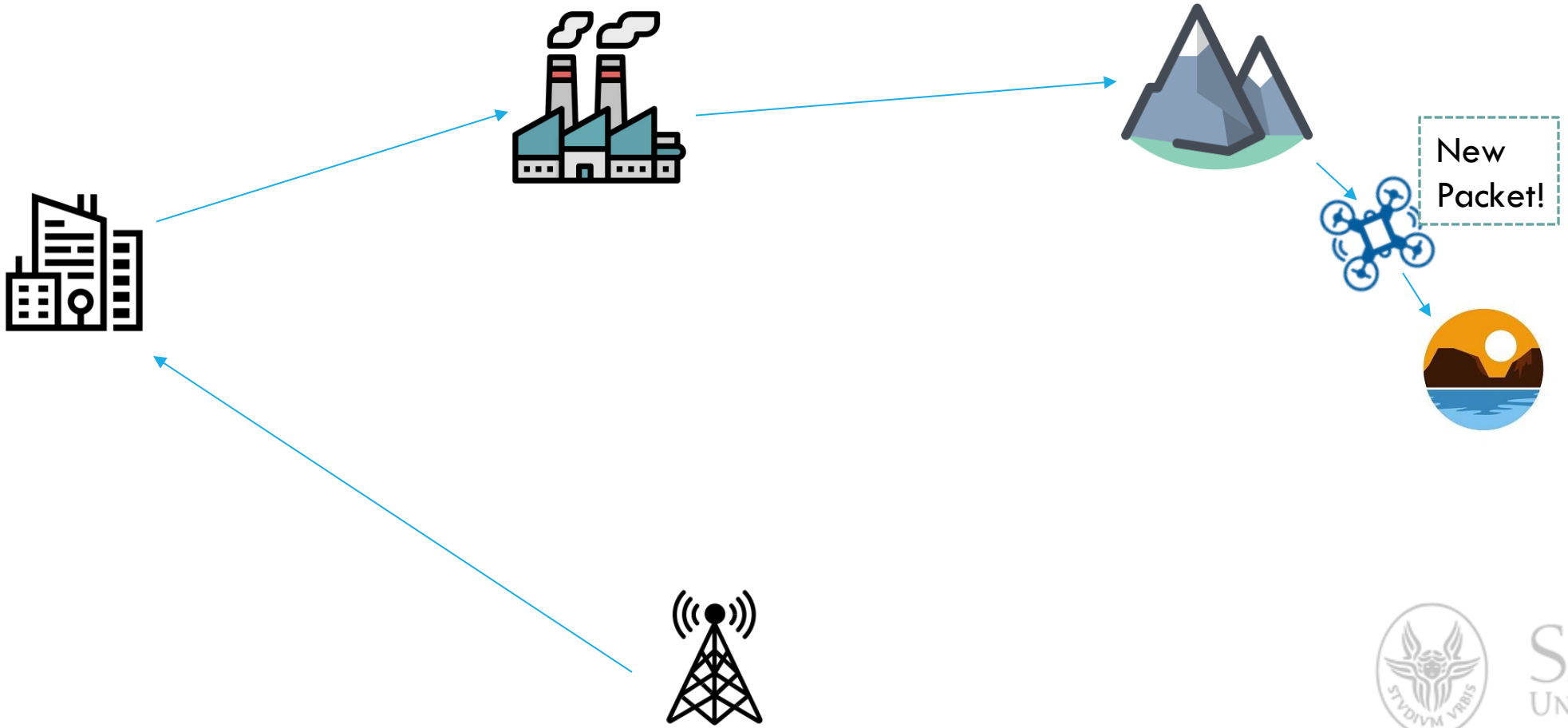
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



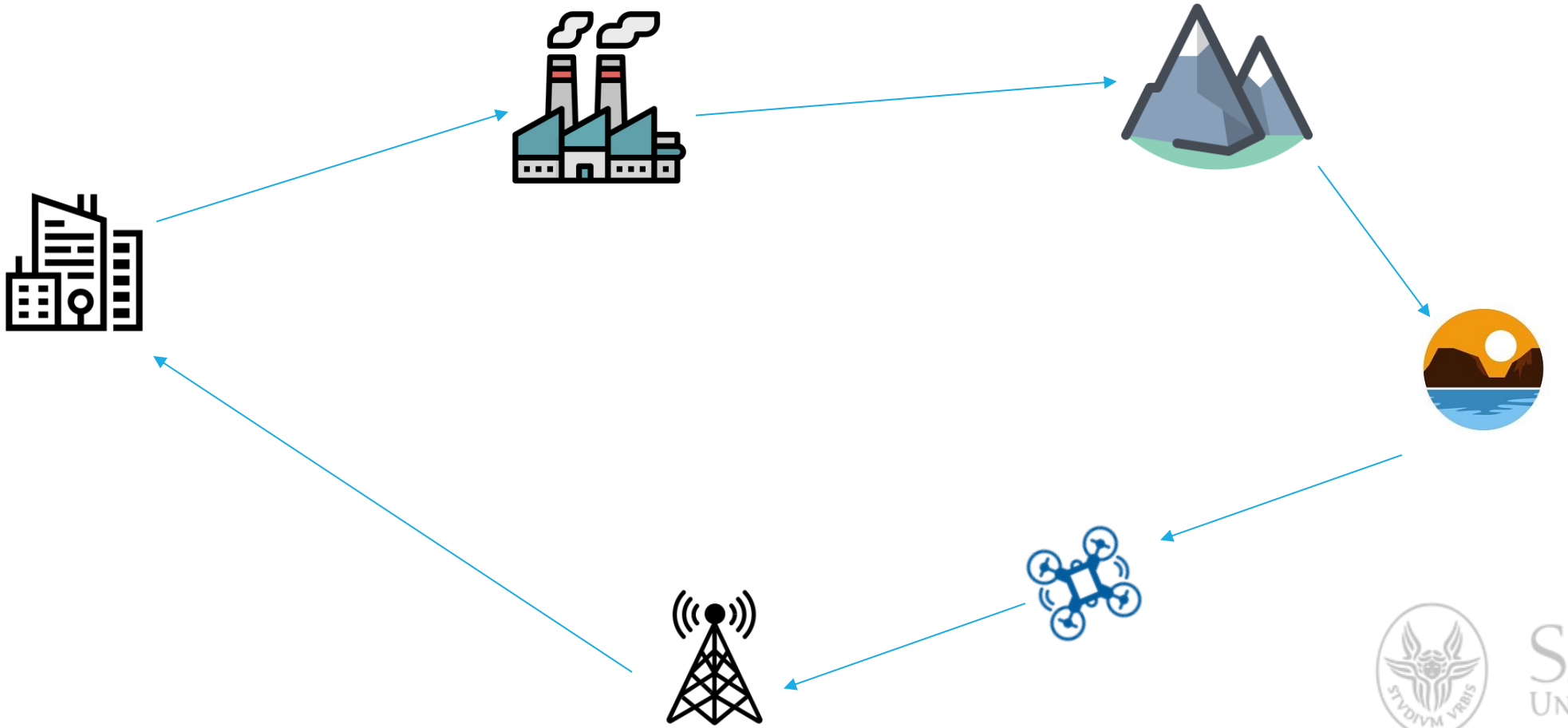
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



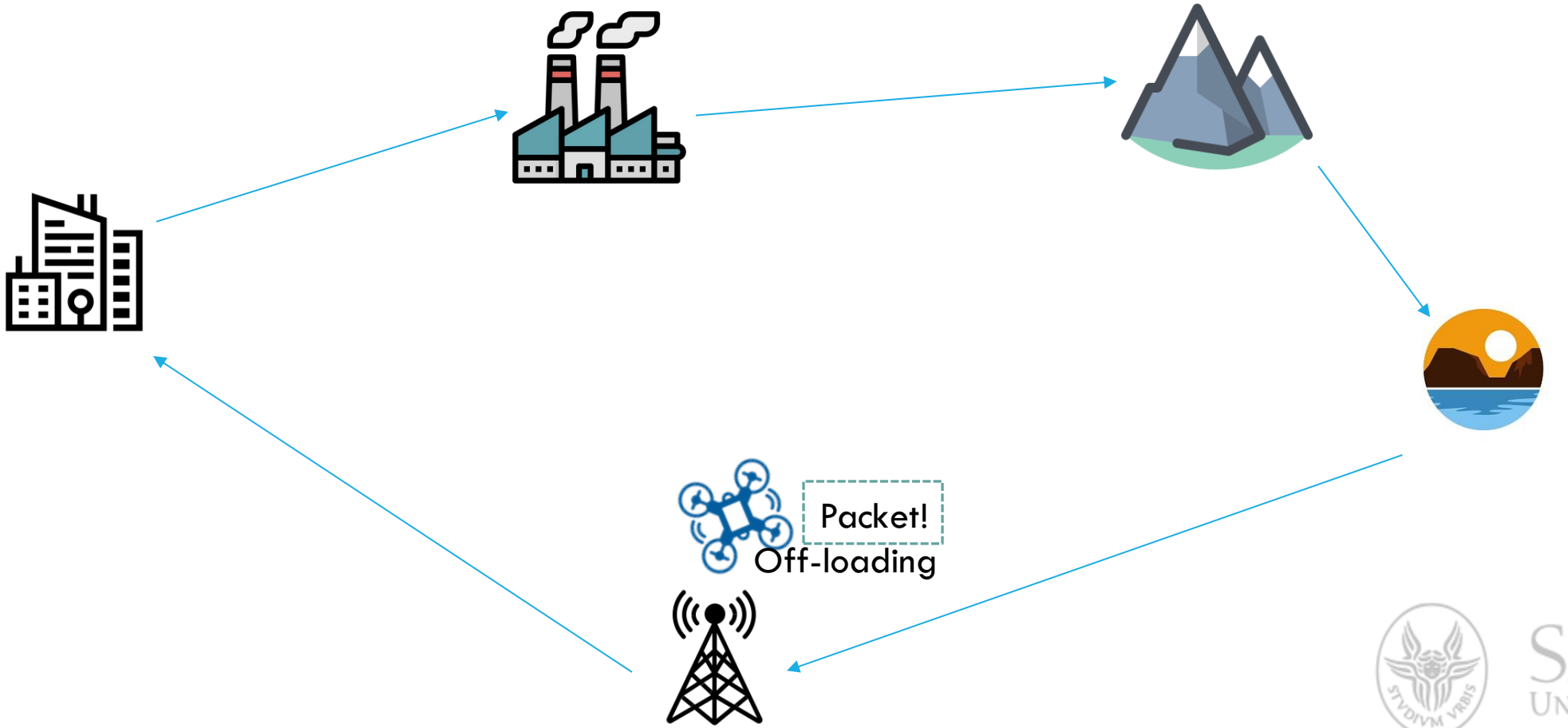
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



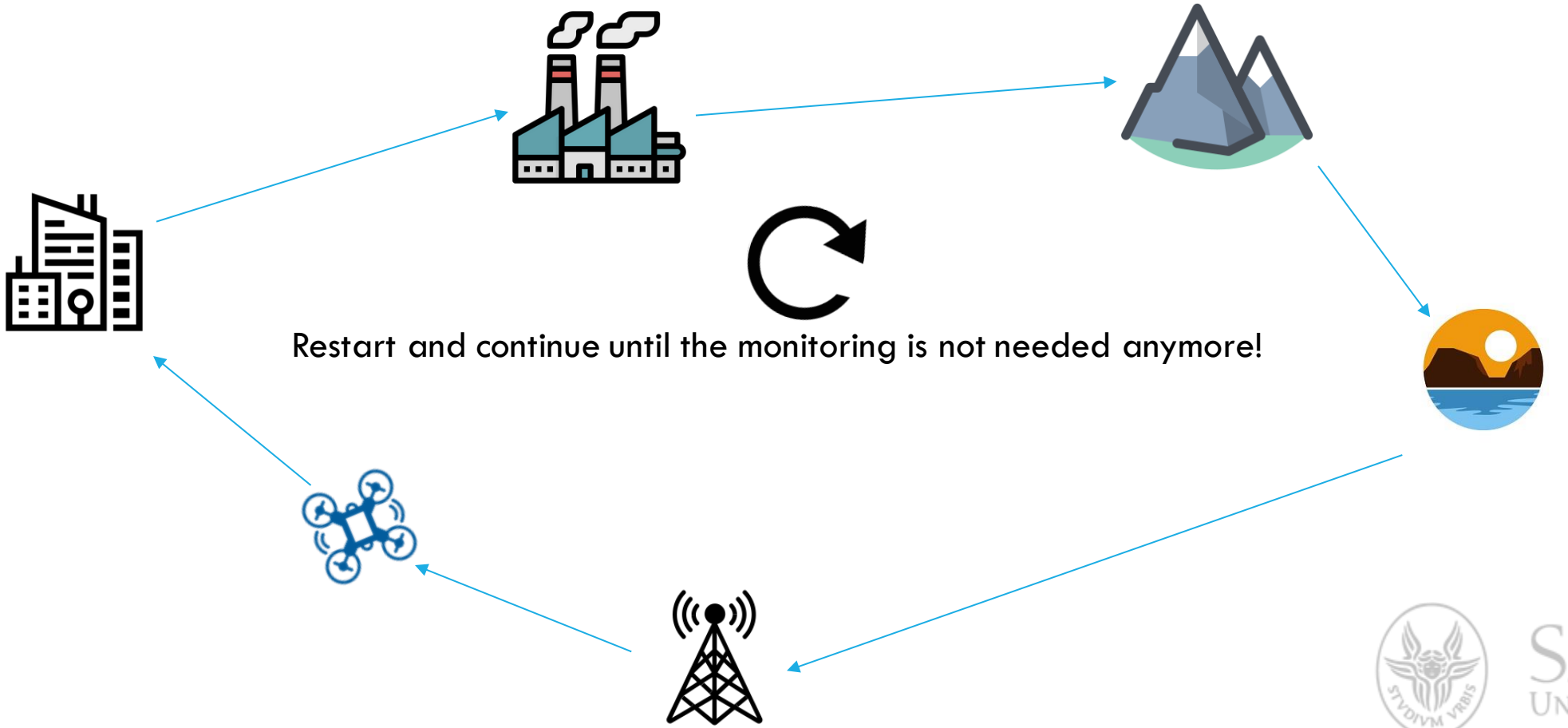
HOMEWORK 2 – PATROLLING SCENARIO

Drones continuously explore an area of interest to detect and monitor the area.



HOMEWORK 2 – PATROLLING SCENARIO

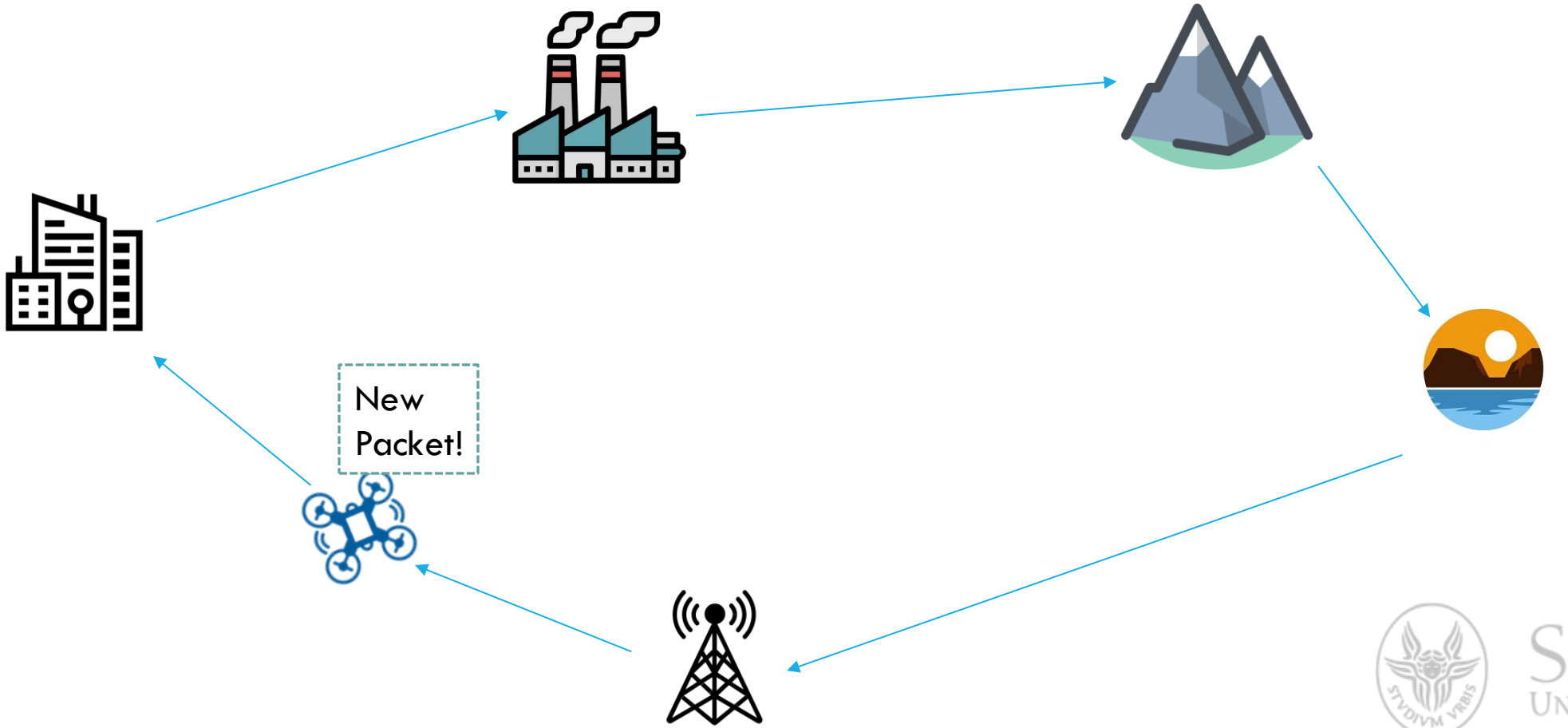
Drones continuously explore an area of interest to detect and monitor the area.



HOMEWORK 2 – PATROLLING SCENARIO

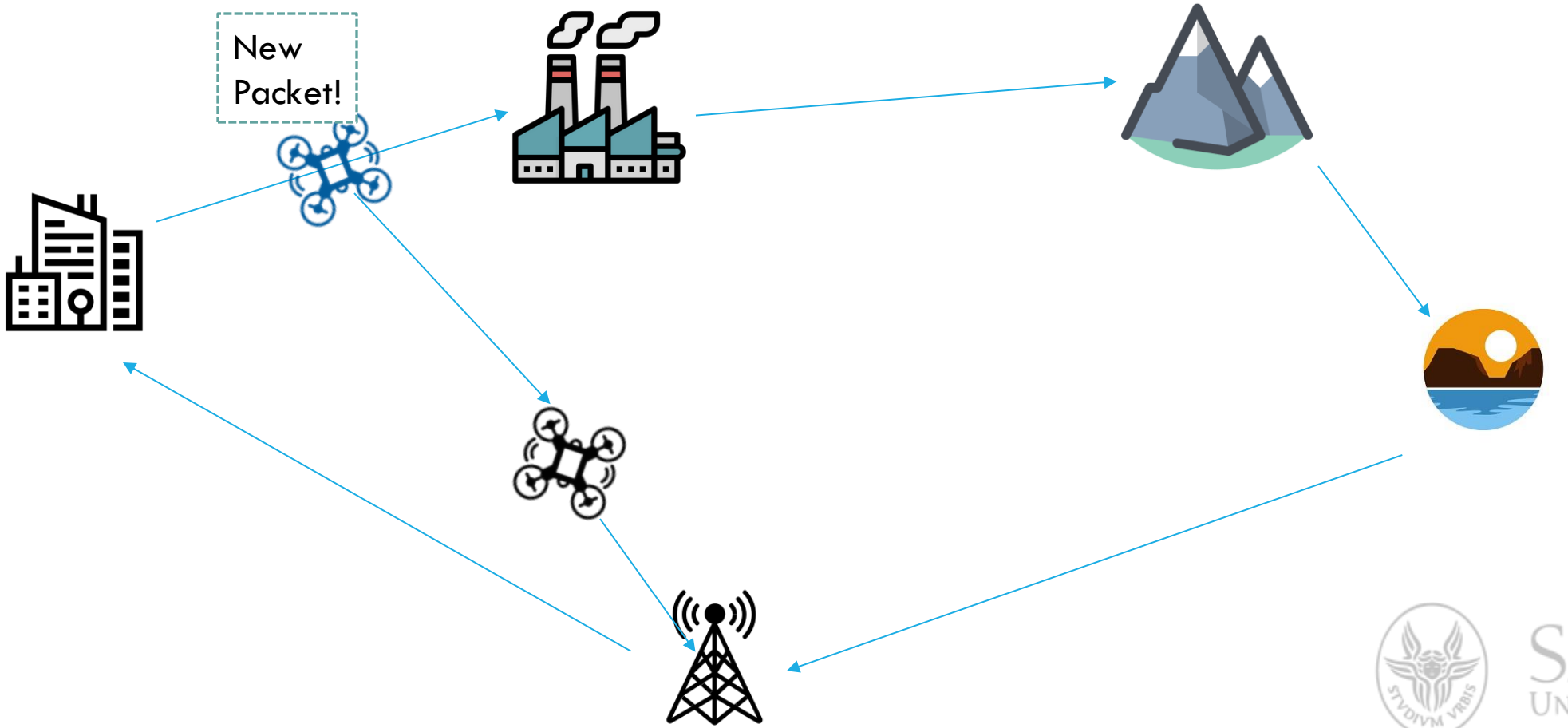
We have problems with this approach?

A packet has to wait until the drone arrives at the depot!!



HOMEWORK 2 – PATROLLING SCENARIO

Idea: Using a squad of drones and a routing protocol we can improve the delivery!!!

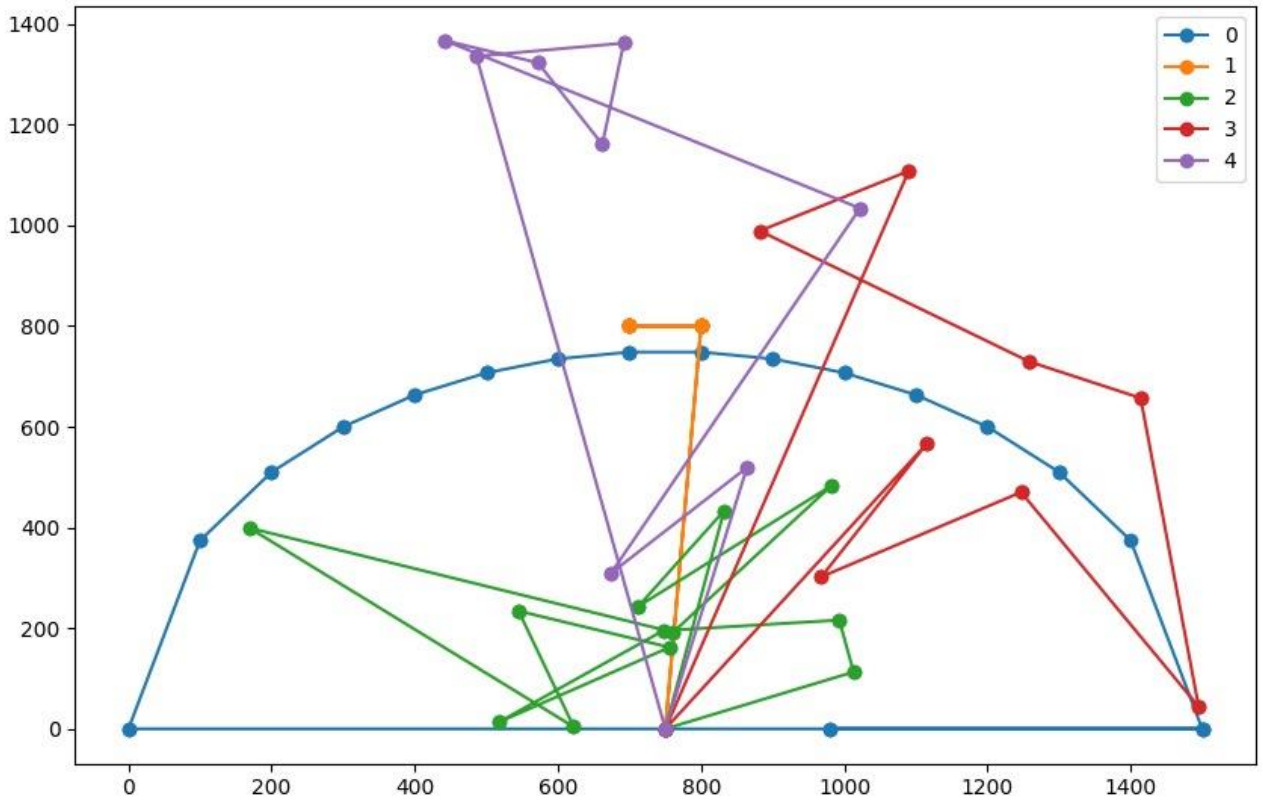


HOMEWORK 2 – PATROLLING SCENARIO

Idea: Using a squad of drones and a routing protocol we can improve the delivery!!!

We have a squad of N-drones:

- 1 Drone monitors the area (drone 0) and collect data!
- N-1 drones are performing other tasks, but they can be used to offload packets and reduce data latency at the depot.



Drone 0 is the only one which collect data!

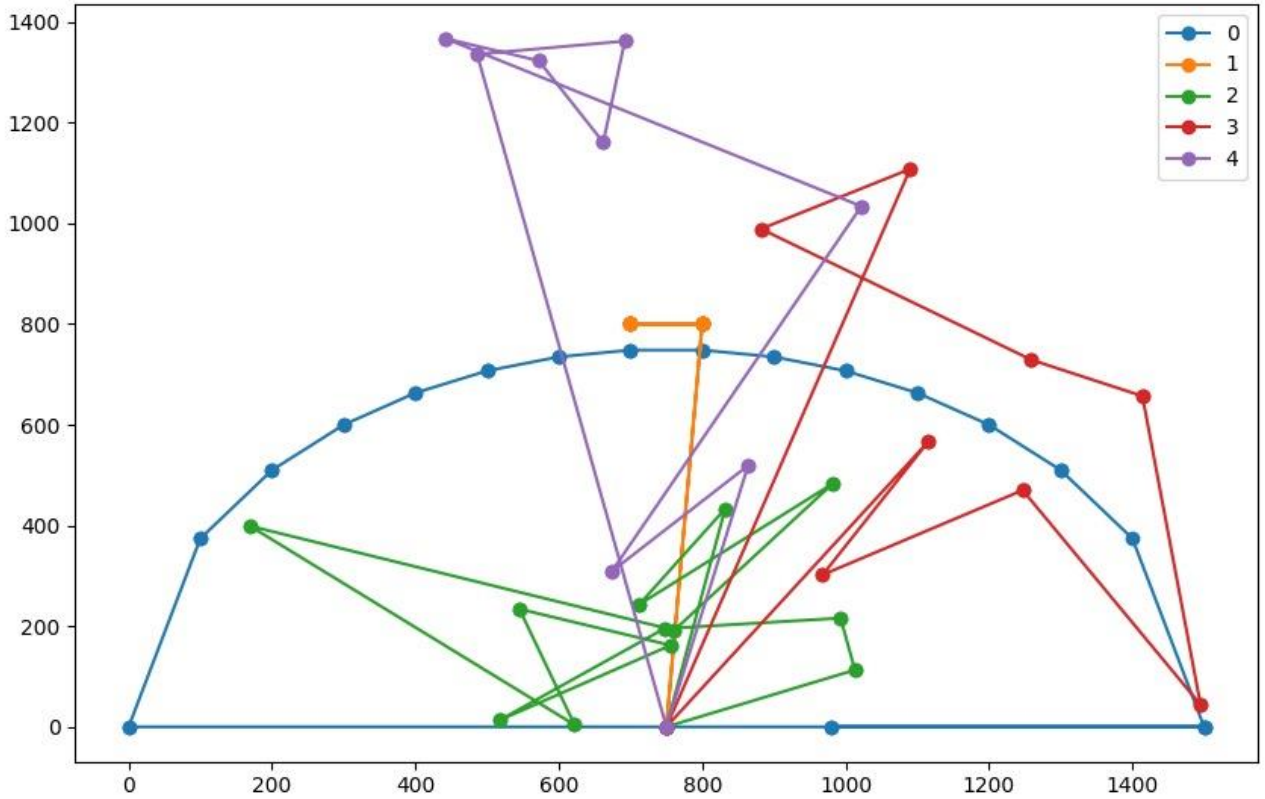
It can:

- Store the data and wait to arrive at the depot
- Send the data to a neighbor drone, which may arrive at the depot before him.

HOMEWORK 2 – YOUR TASK

Idea: Using a squad of drones and a routing protocol we can improve the delivery!!!

Only drone 0 can take decision: keep or transmit the packet!
The other drones can receive a packet and carry that packet to the depot **(THEY DO NOT COMMUNICATE)**



GOAL:

Create a **Reinforcement Learning** Routing Protocol for drone 0.

To decide whether keep or send the packet.

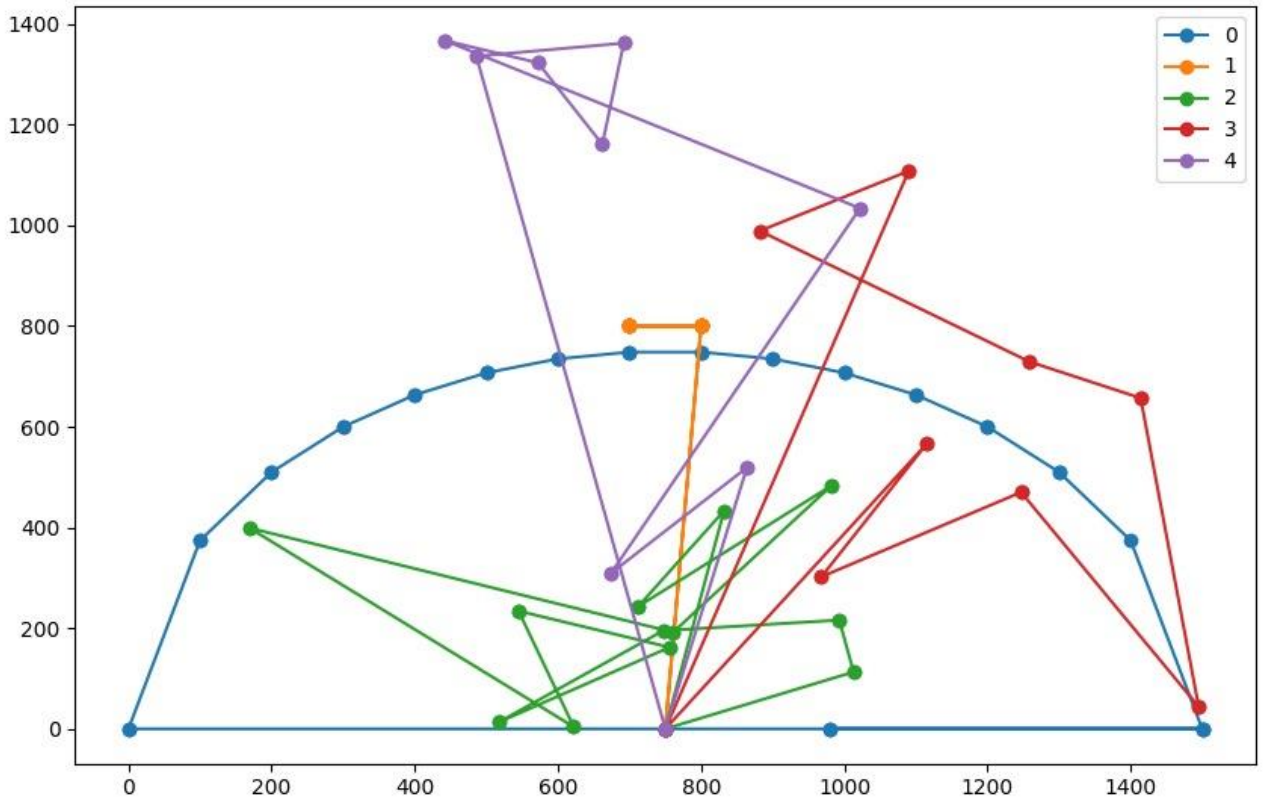
HOMEWORK 2 – ASSUMPTIONS

Idea: Using a squad of drones and a routing protocol we can improve the delivery!!!

The patrolling mission repeat with the same trajectories until the simulation ends.

Note: You can assume to know your trajectory (drone 0) but you do not the trajectories of the other drones!!!

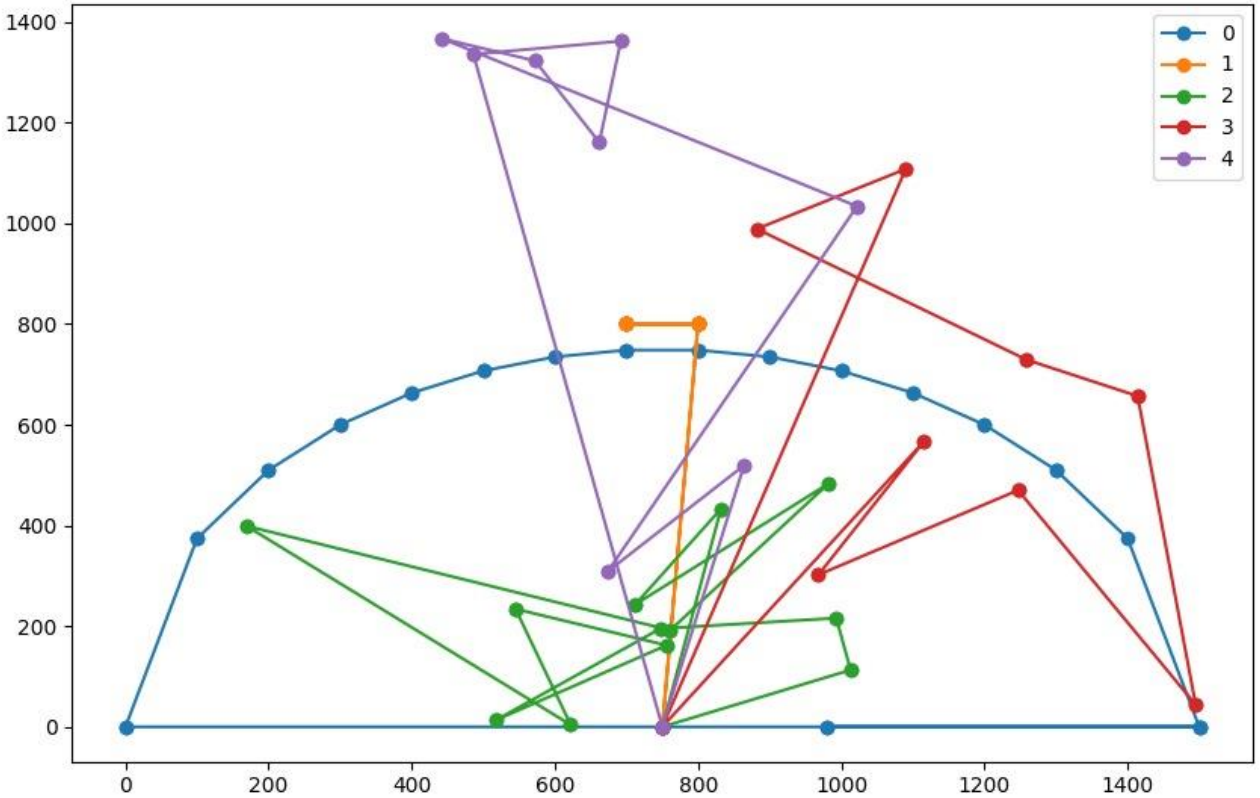
The packets have a limited time to live!



HOMEWORK 2 – GOAL

You have to delivery as much as possible packets to the depot as primary task.
As secondary task you have to reduce the latency of the packets!

The final score/goal of your approach is to minimize: $1.5 \cdot |expired\_packets| \cdot ttl + \sum_{pck \in delivered} delivery\_time$



ttl -> the maximum time to live of a packet in seconds

HOMEWORK 2 – HOW

You can implement the Class below, using a Reinforcement Learning approach.
We suggest you start with a Q-learning based approach.

```
class AIRouting(BASE_routing):
    def __init__(self, drone, simulator):
        BASE_routing.__init__(self, drone, simulator)
        # random generator
        self.rnd_for_routing_ai = np.random.RandomState(self.simulator.seed)
        self.taken_actions = {}  #id_event : action_taken

    def feedback(self, drone, id_event, delay, event):...

    def relay_selection(self, opt_neighbors, pkd):...

    def print(self):...
```

HOMEWORK 2 – HOW

The main method to implement is always the same:

```python
def relay_selection(self, opt_neighbors, pkd):
    """ arg min score  -> geographical approach, take the drone closest to the depot """
    state, action = None, None

    # Store your current action --- you can add several stuff if needed to take a reward later
    self.taken_actions[pkd.event_ref.identifier] = (state, action)

    return None  # here you should return a drone object!
```

Notice that, you can store the state and the action (or what you prefer) in the self.taken_action which is useful to update your model according the feedback!

The feedback is delayed (when the packet arrives at the depot or expires) you will be notified.

HOMEWORK 2 – HOW

The feedback is notified using a control channel (long-range radio with low bit rate) to drone 0.

```
def feedback(self, drone, id_event, delay, outcome):
    """ return a possible feedback, if the destination drone has received the packet """
    # Packets that we delivered and still need a feedback
    print(self.taken_actions)

    # outcome == -1 if the packet/event expired; 0 if the packets has been delivered to the depot
    # Feedback from a delivered or expired packet
    print(drone, id_event, delay, outcome)

    # remove the entry, the action has received the feedback
    # Be aware, due to network errors we can give the same event to multiple drones and receive multiple feedback for the same packet!!
    if id_event in self.taken_actions:
        state, action = self.taken_actions[id_event]
        del self.taken_actions[id_event]
        # reward or update using the old state and the selected action at that time
        # do something or train the model (?)
```

It contains: the drone that delivered the packet, the identifier of the event; the delay of the packet (time elapsed from the beginning of the event); the outcome.

The outcome is 0 if the event has been delivered to the depot; -1 if the event has expired before.

HOMEWORK 2 – HOW

The feedback is notified using a control channel (long-range radio with low bit rate) to drone 0.

```
def feedback(self, drone, id_event, delay, outcome):
    """ return a possible feedback, if the destination drone has received the packet """
    # Packets that we delivered and still need a feedback
    print(self.taken_actions)

    # outcome == -1 if the packet/event expired; 0 if the packets has been delivered to the depot
    # Feedback from a delivered or expired packet
    print(drone, id_event, delay, outcome)

    # remove the entry, the action has received the feedback
    # Be aware, due to network errors we can give the same event to multiple drones and receive multiple feedback for the same packet!!
    if id_event in self.taken_actions:
        state, action = self.taken_actions[id_event]
        del self.taken_actions[id_event]
        # reward or update using the old state and the selected action at that time
        # do something or train the model (?)
```

Using the **id_event** and **self.taken_actions** you can understand which action and state have caused that result!

Notice that, due to communication errors, an event can be delivered multiple times!!

HOMEWORK 2 – HOW

Example of a feedback:

Drone, the identifier of the event, the delay of the packet, the outcome (expired).

```
Drone 3 140600041281616 2000 -1
```

Drone, the identifier of the event, the delay of the packet, the outcome (delivered).

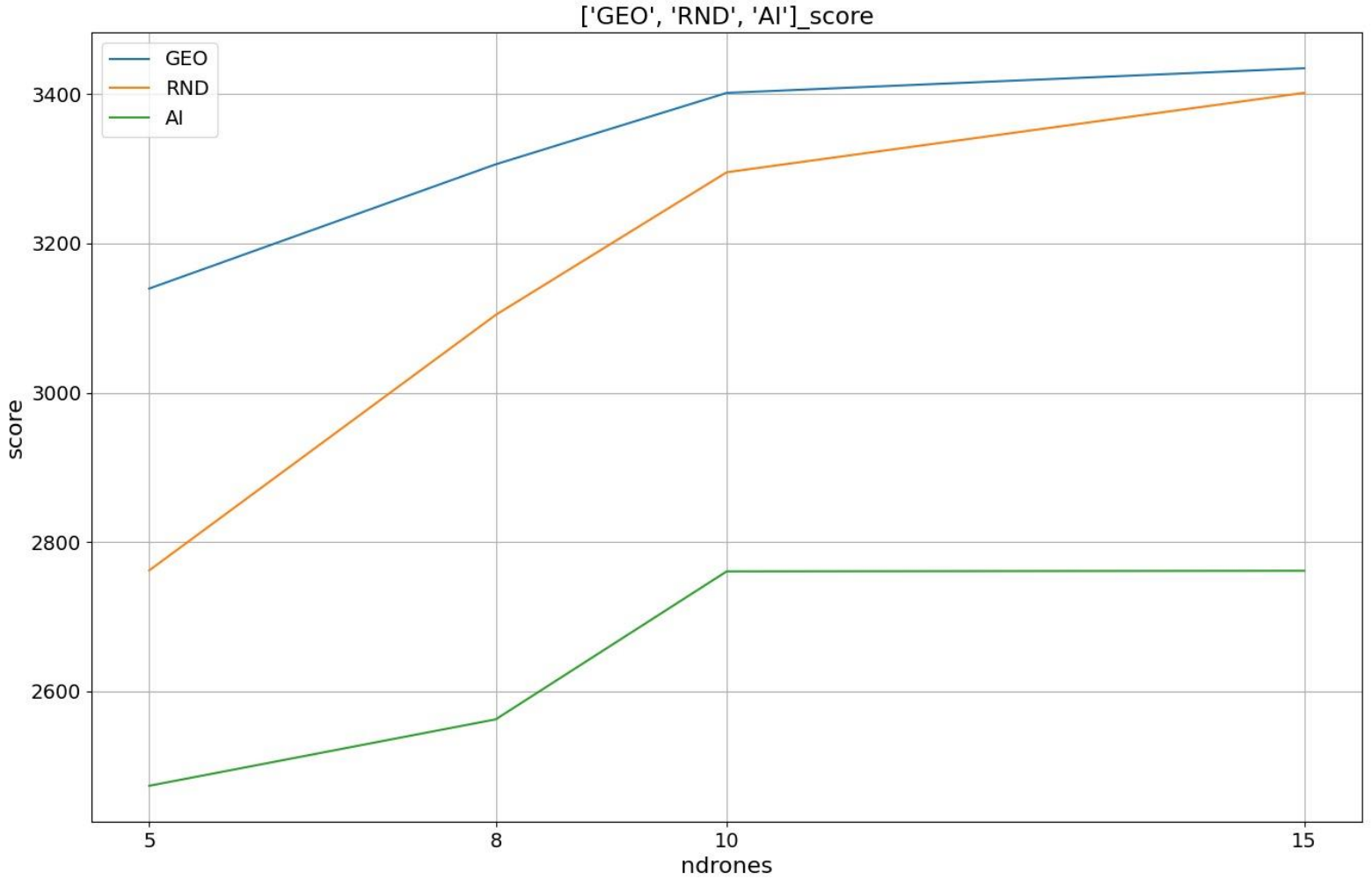
```
Drone 2 140600041301712 1402 1
```

HOMEWORK 2 – RUN MULTIPLE SIMULATIONS

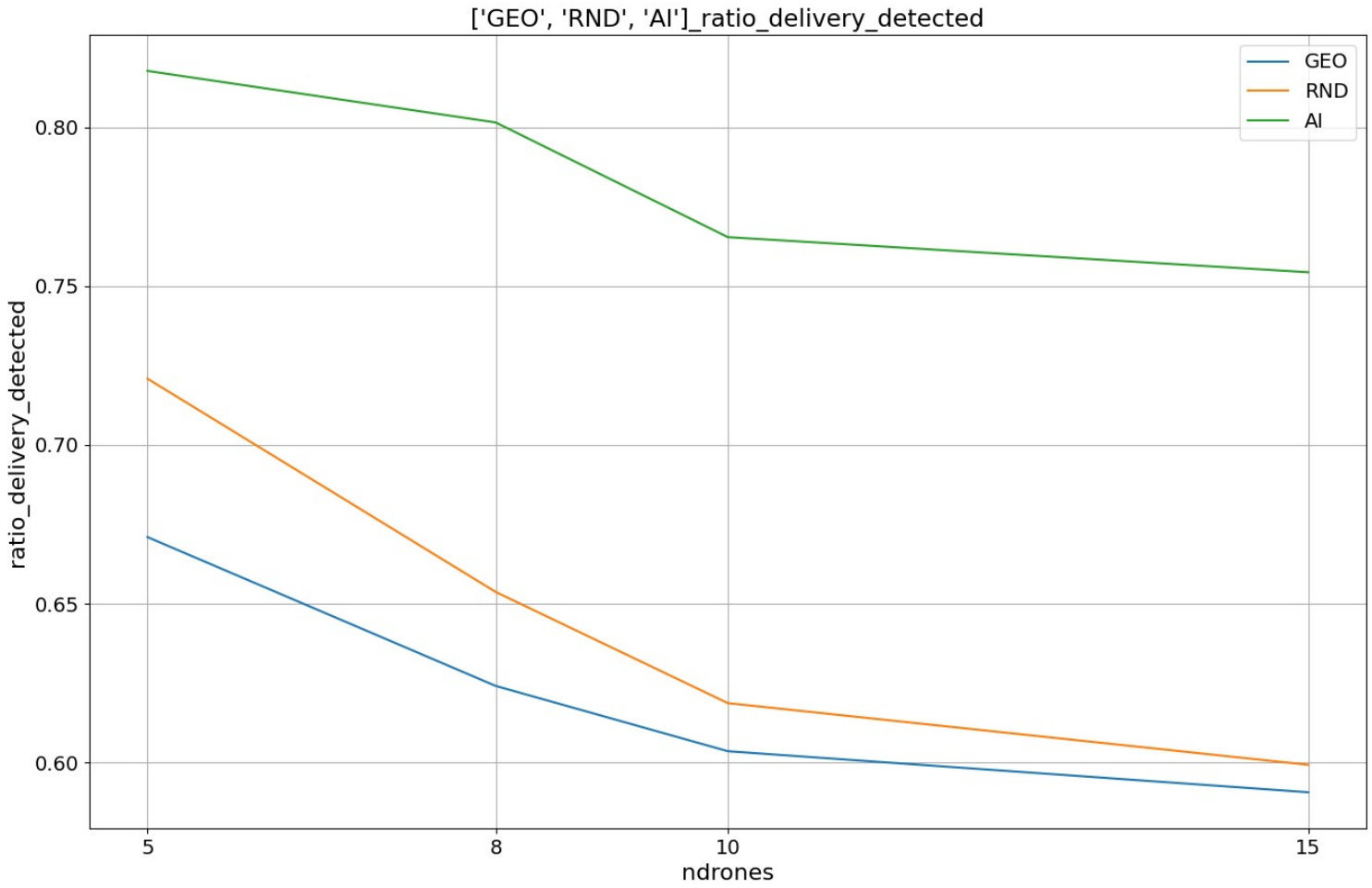
As for the first homework:

```
#-------------------------------------------------------------#
#          _  _ ___  ___  ___  _  _ ___ ___                   #
#         | \| |   \| _ \/ _ \| \| | __/ __|                  #
#         | .` | |) |   / (_) | .` | _|\__ \                  #
#         |_|\_|___/|_|_\\___/|_|\_|___|___/                  #
#                                                             #
#-------------------------------------------------------------#

#test baselines
for nd in "5" "10" "15";
do
    for alg in "GEO" "RND" "AI";
    do
        echo "run: ${alg} - ndrones ${nd} "
        python3 -m src.experiments.experiment_ndrones -nd ${nd} -i_s 1 -e_s 3 -alg ${alg} &
        #python3 -m src.experiments.experiment_ndrones -nd ${nd} -i_s 10 -e_s 20 -alg ${alg} &
        #python3 -m src.experiments.experiment_ndrones -nd ${nd} -i_s 20 -e_s 30 -alg ${alg} &
    done;
done;
wait

python3 -m src.experiments.json_and_plot -nd 5 -nd 10 -nd 15 -i_s 1 -e_s 3 -exp_suffix GEO -exp_suffix RND -exp_suffix AI
```

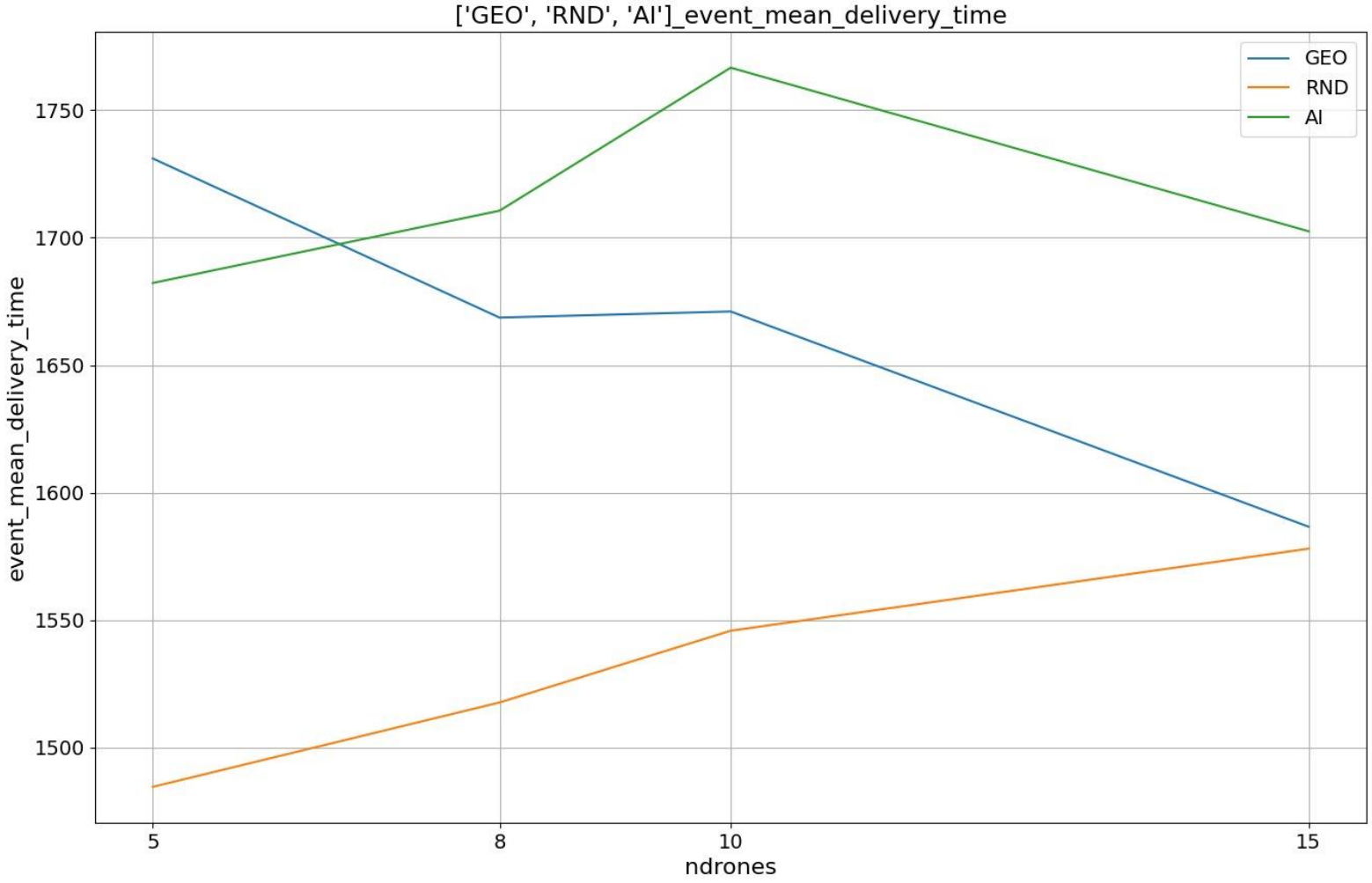
HOMEWORK 2 – RESULTS



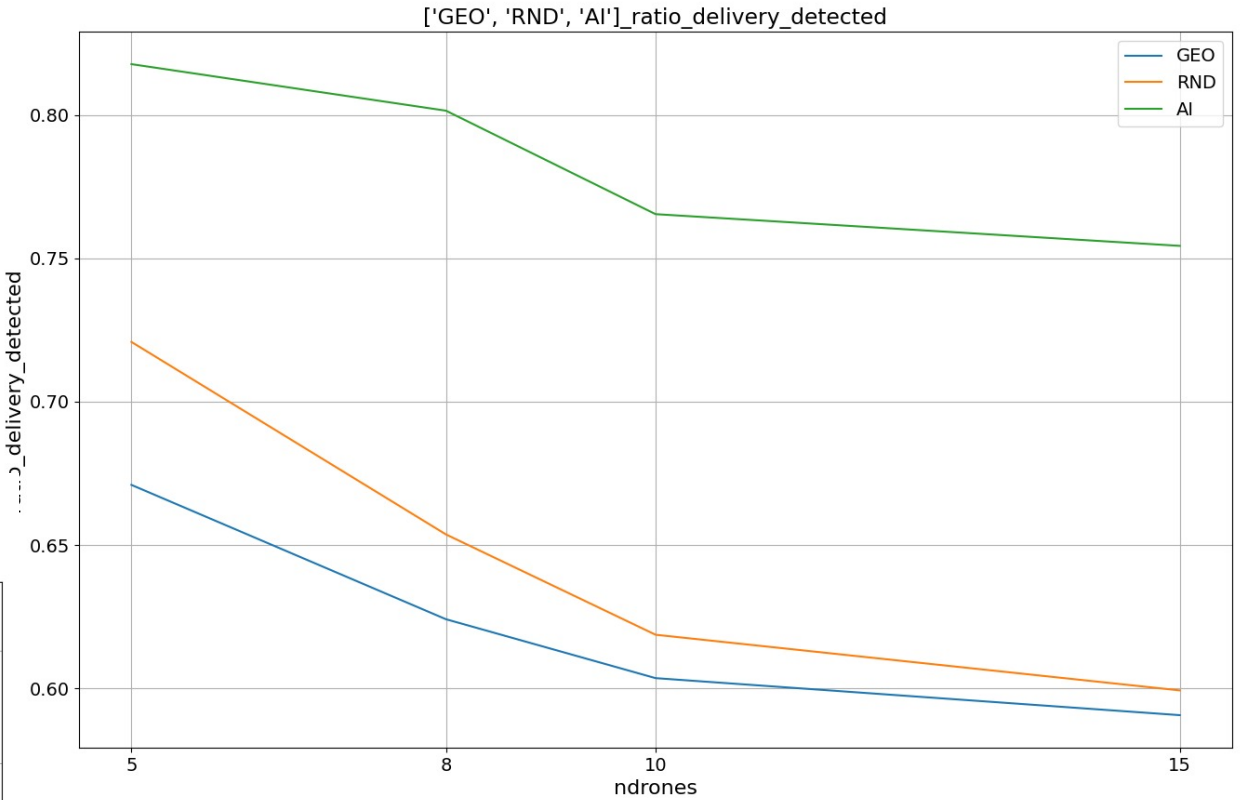
HOMEWORK 2 – RESULTS

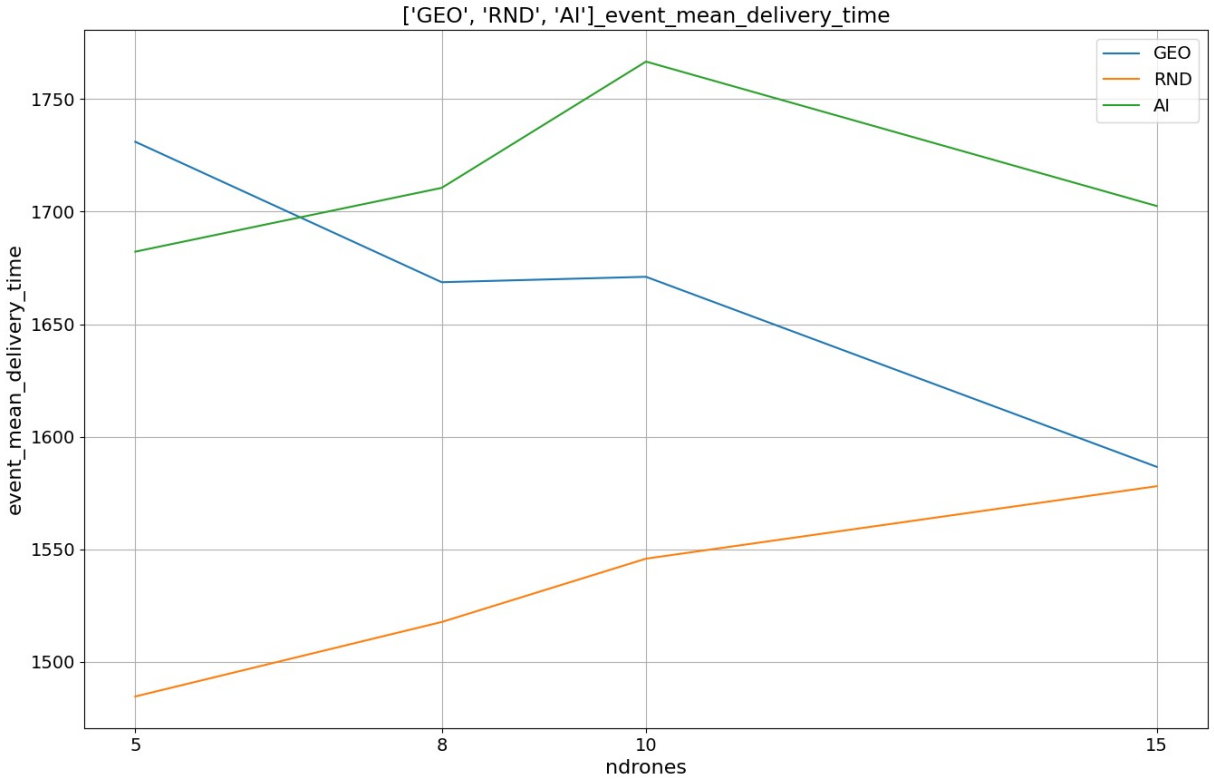


HOMEWORK 2 – RESULTS



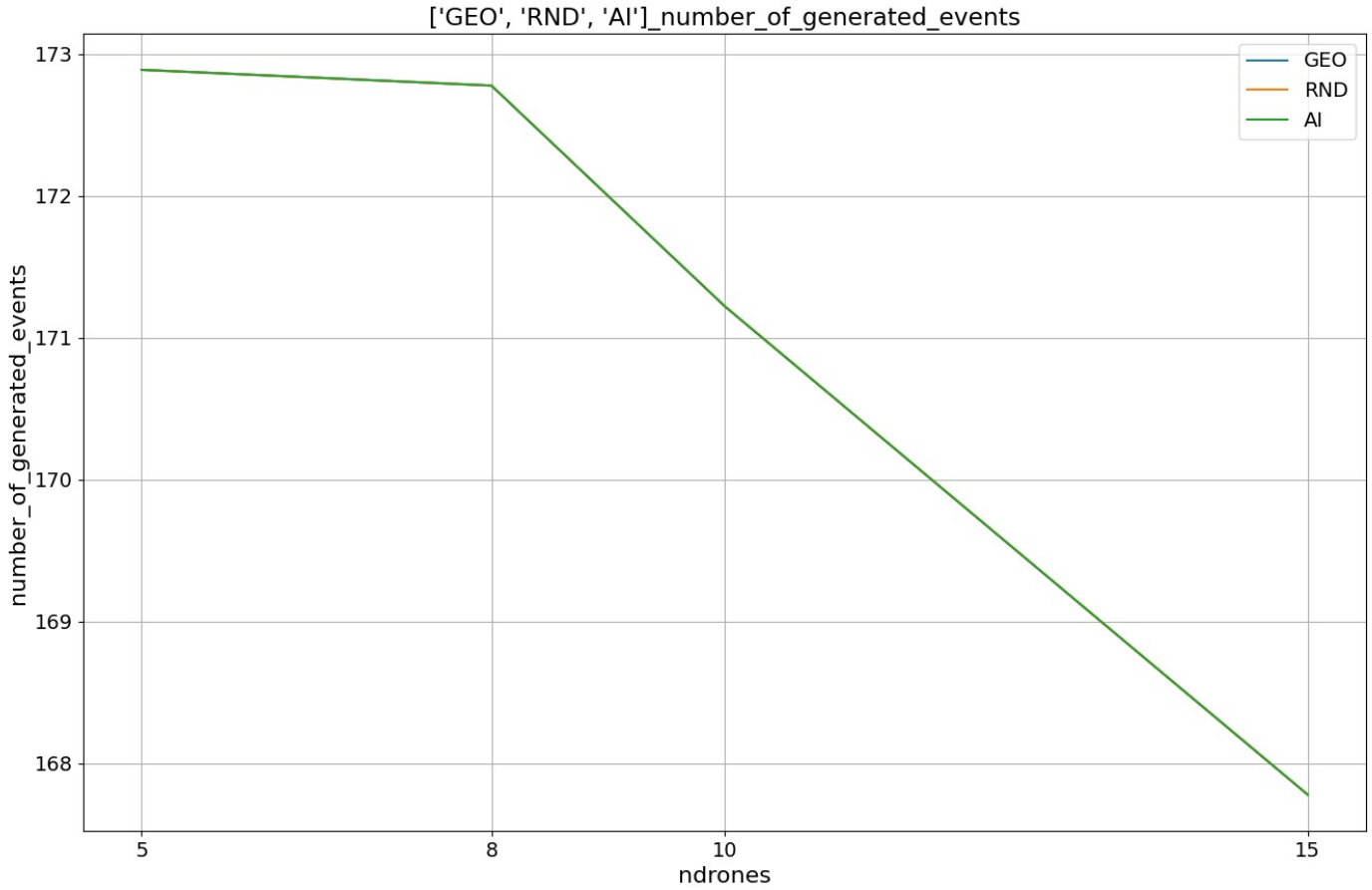
HOMEWORK 2 – RESULT





HOMEWORK 2 – RESULTS

Only for check, this chart!



HOMEWORK 2!!

Deadline: 23:59:59 - 18/12/2020

How : in group of **1** to **3**

Submission: report + code

Score: up to 31 (30 cum laude)

Interaction lecture: 10:00 - 11/12/2020

FAQ: Common questions will be answered on classroom, above the homework post

Evaluation: approach; report; algorithm score (% of delivered packets).

Challenge: rank of top 10 algorithms.

Submission: Same of Homework 1.

Note: Convergence time is really important here!

CONTACTS

Andrea Coletta:

coletta@di.uniroma1.it