AERO-TRAIN Summer School

DAY 3 – HUMAN-ROBOT INTERACTION

Day Introduction

- **Welcome!! :)**

Sessions

Cloud-enabled Remote Control

Presented by

- Gerasimos Damigos
- Viswa Narayanan Sankaranarayanan
- Achilleas Santi Seisa

Haptics and Teleoperation

Presented by

- Manuel Fernandez Gonzalez
- Julien Mellet

Agenda

Presentation

- Network control systems – Cloud-enabled remote control

Tutorial 1

- Task 1 – UAV position control

Coffee Break

- Task 2 - Teleoperation Interface
- Task 3 – Delay compensation

Lunch

Presentation

- Haptics – How to interface operator with aerial manipulators

Tutorial 2

Final remarks

Tutorial 1

Please follow the instructions and download and run the bash script files of the GitHub repository!

https://github.com/AERO-TRAIN/exercises_summer_school_hri_day

```
chmod +x run_gazebo.sh
./run_gazebo.sh
```

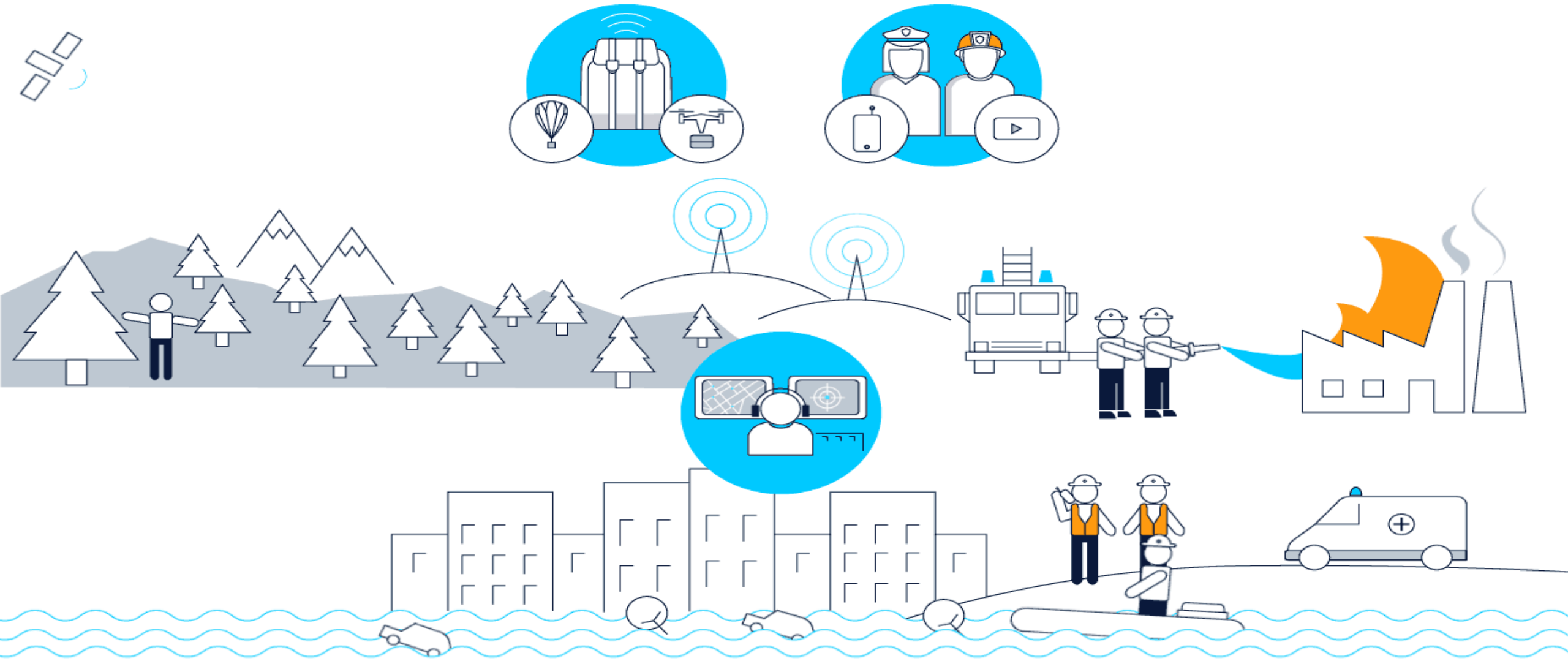
```
chmod +x run_controller.sh
./run_controller.sh
```

ROBOTICS TEAM
at Luleå University of Technology

ERICSSON

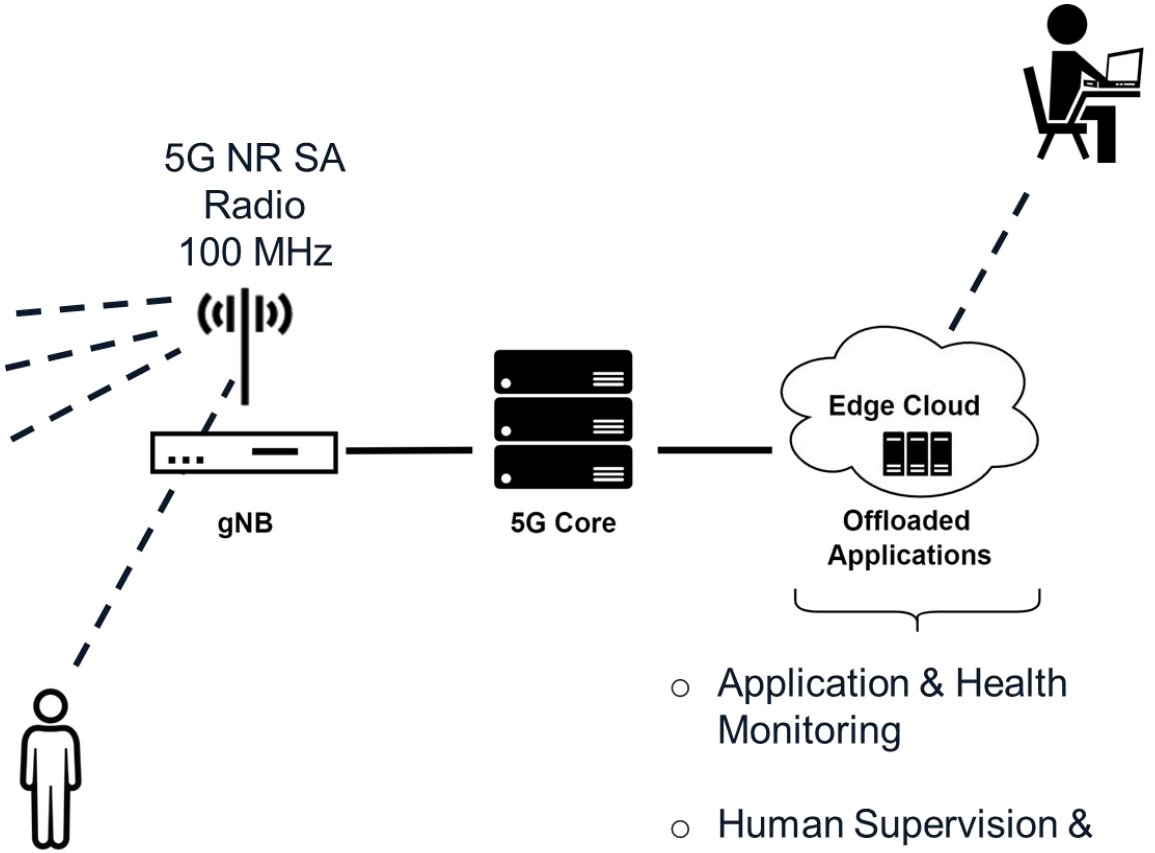
AERO TRAIN

Cloud-enabled Remote Control



Motivation

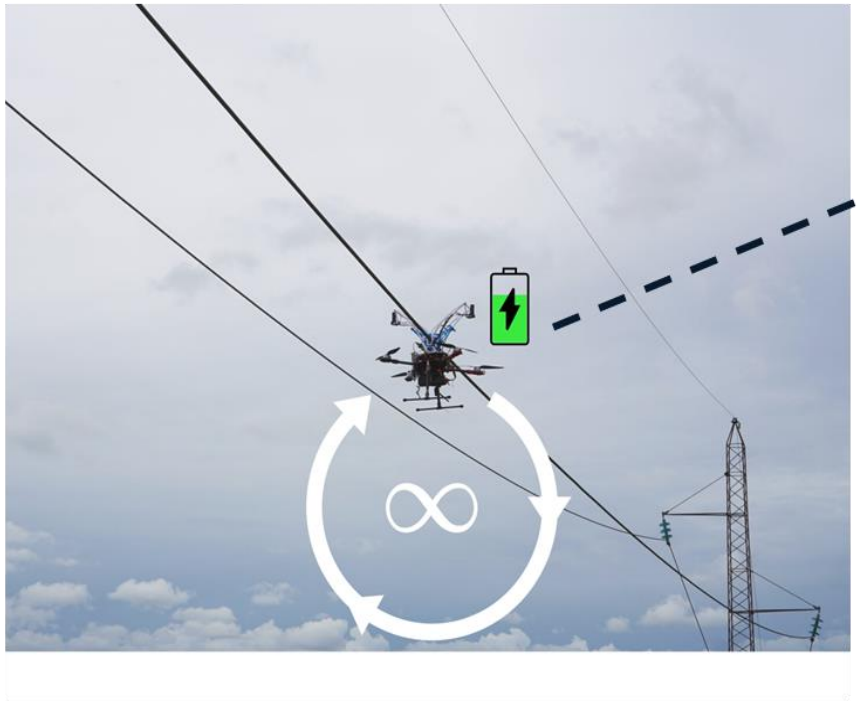
Image source: Boliden

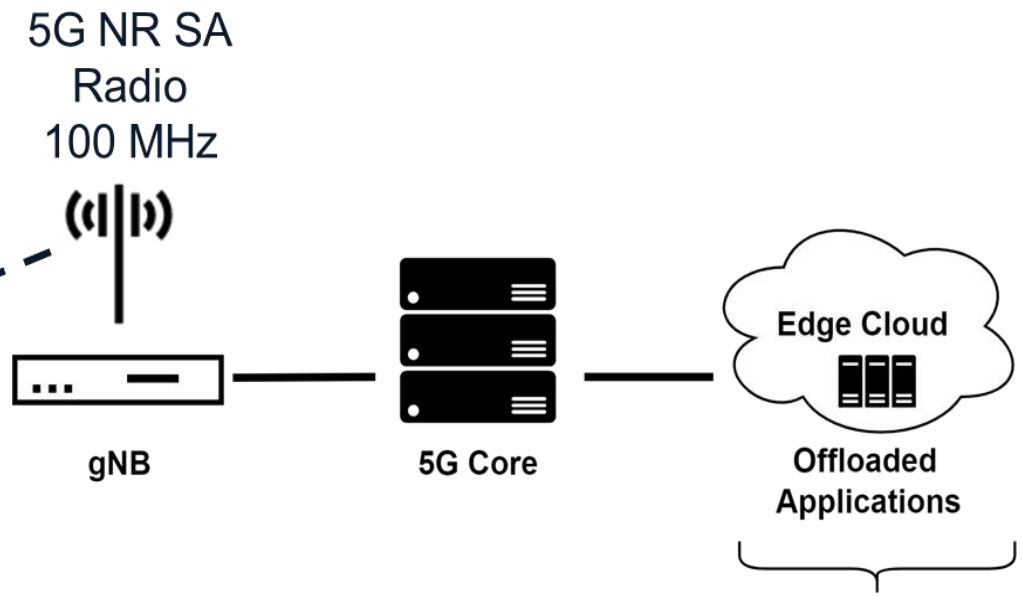


- Application & Health Monitoring
- Human Supervision & Guidance
- Information Sharing & Centralized applications.

Ref: ”A case study on automation in mining”.

Motivation

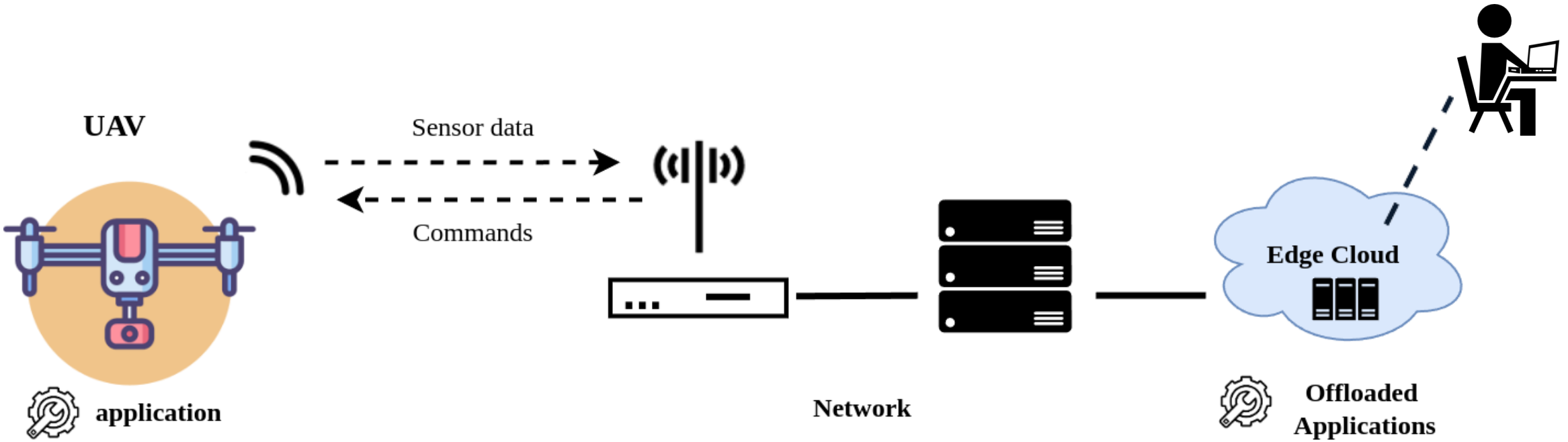




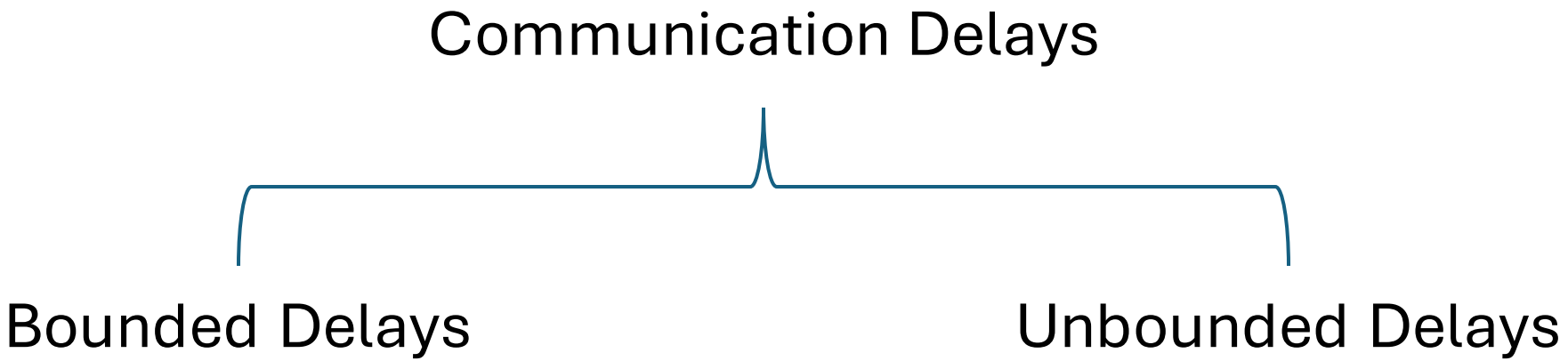
- Offloaded perception & autonomy components.
- e.g., object detection and path planning offloaded into the cloud.
- Data logging.

Ref: Hoang, Viet Duong, et al. "Autonomous Overhead Powerline Recharging for Uninterrupted Drone Operations." *arXiv preprint arXiv:2403.06533* (2024).

Architecture



Challenges



Challenges

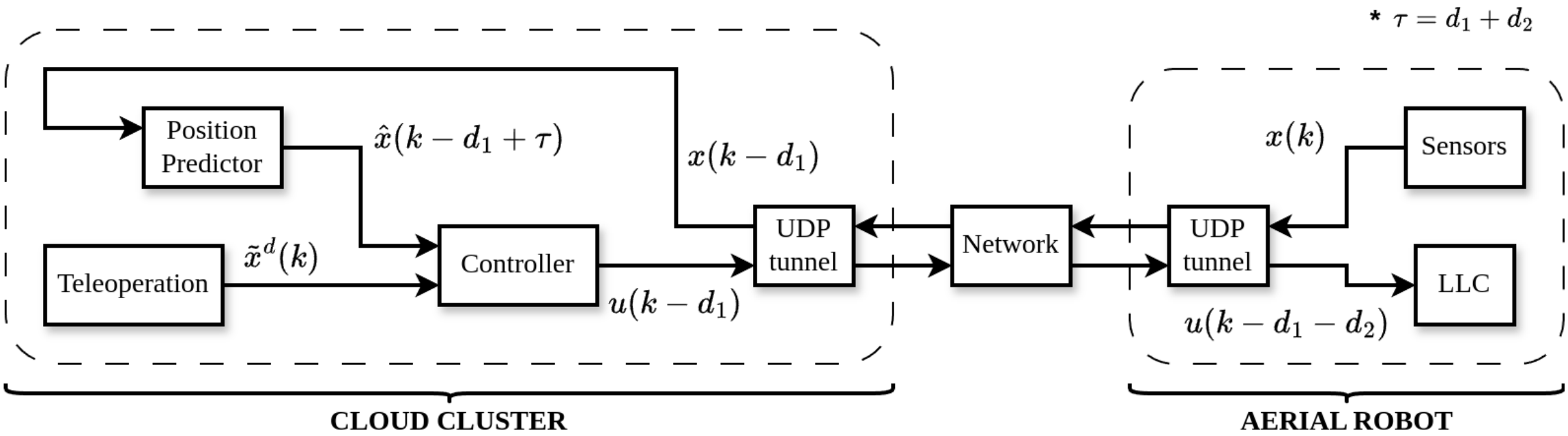
Communication Delays

Bounded Delays

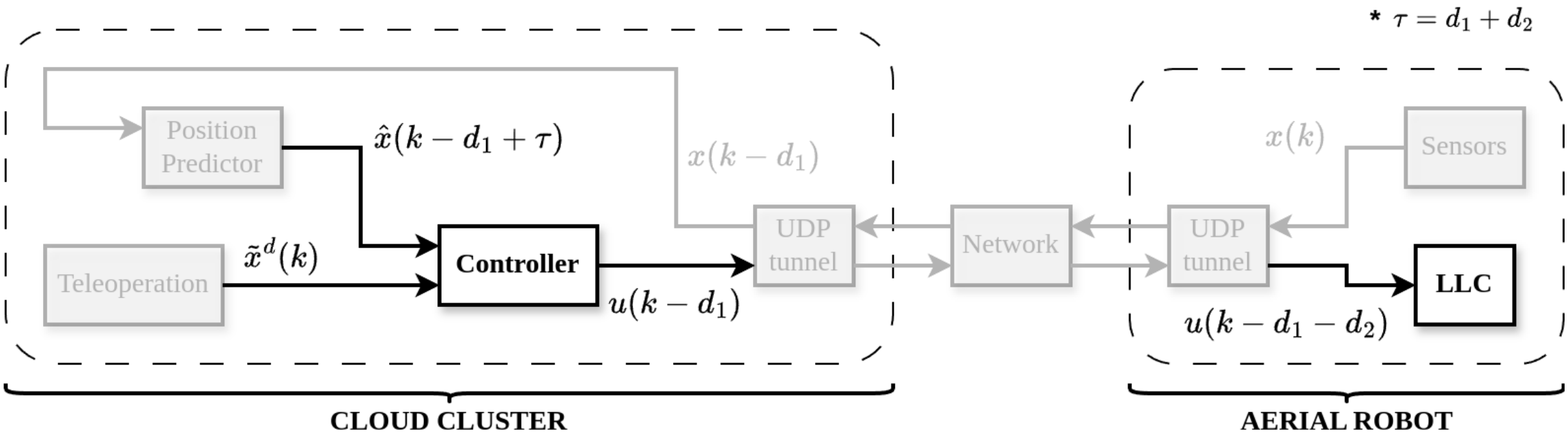
Compensate delay effect in robot's sensor data

Unbounded Delays

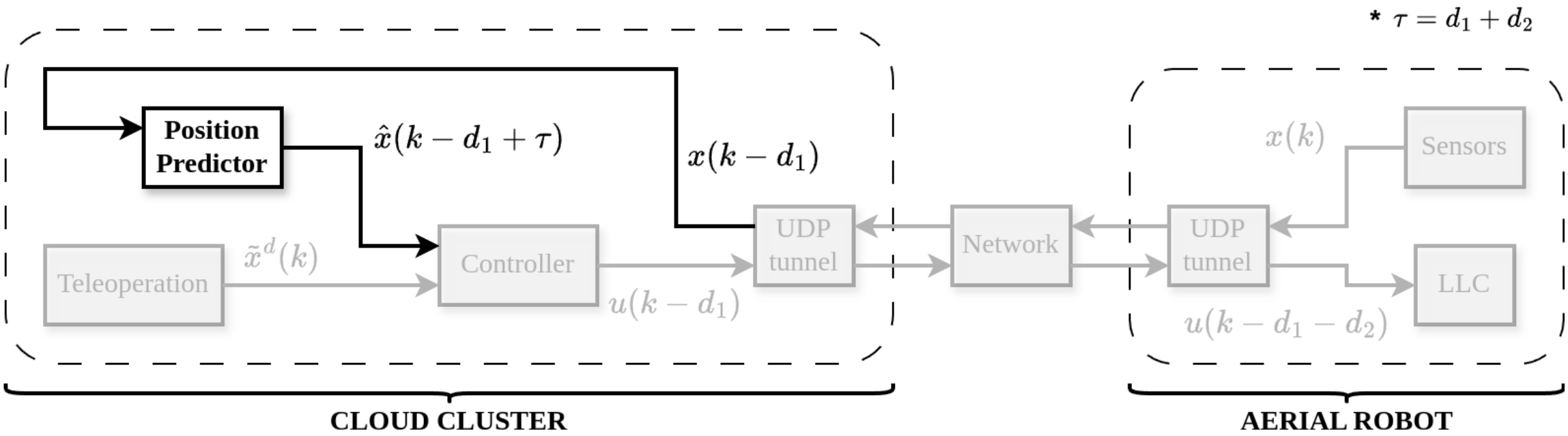
Overall Architecture



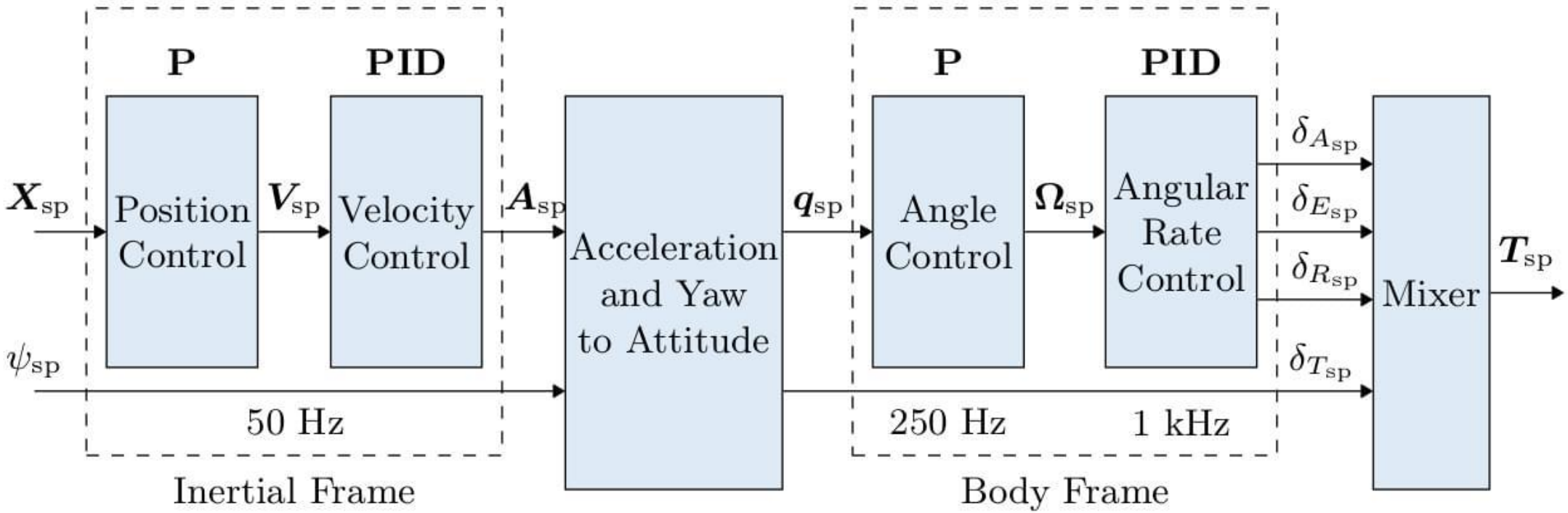
Controller


* $\tau = d_1 + d_2$
Position Predictor
$\hat{x}(k - d_1 + \tau)$
$x(k - d_1)$
Teleoperation
$\tilde{x}^d(k)$
Controller
$u(k - d_1)$
UDP tunnel
Network
UDP tunnel
$x(k)$
Sensors
LLC
$u(k - d_1 - d_2)$
CLOUD CLUSTER
AERIAL ROBOT

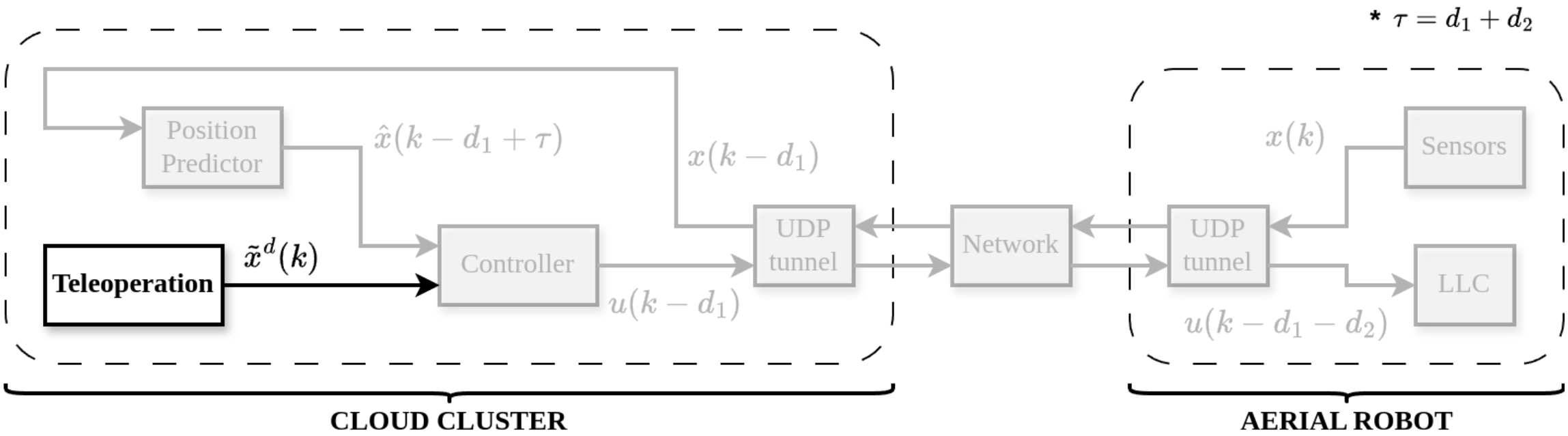
Position Predictor



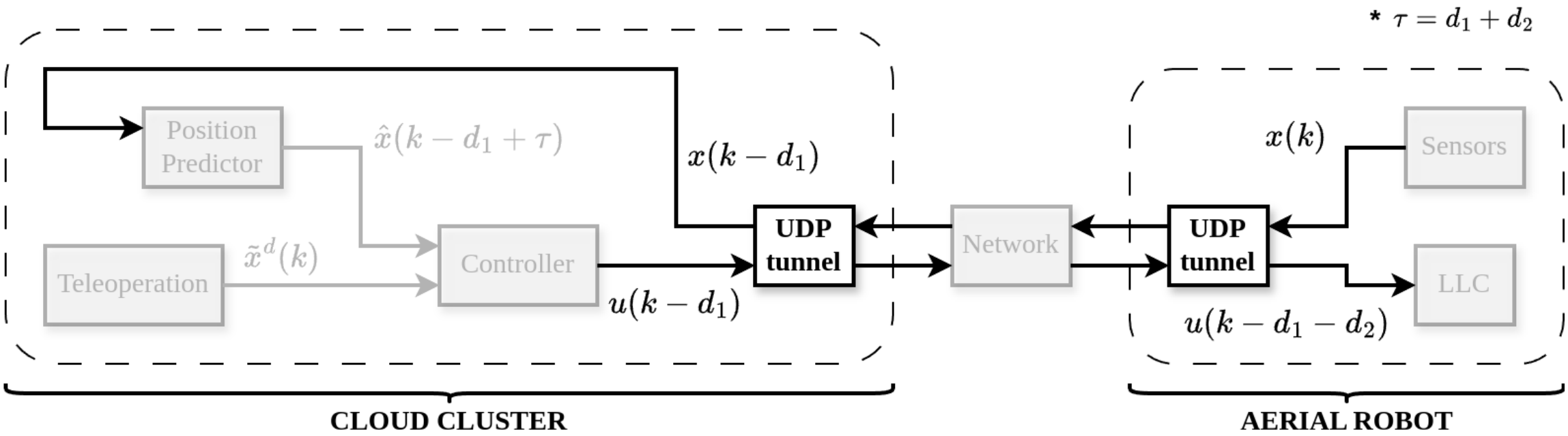
PX4 Controller Example



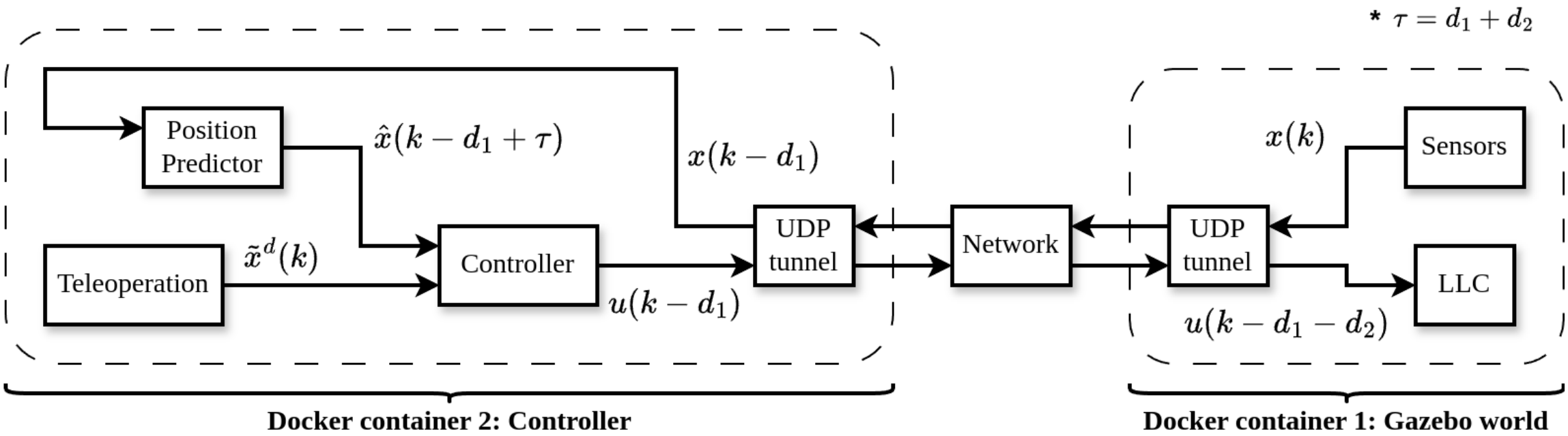
Teleoperation



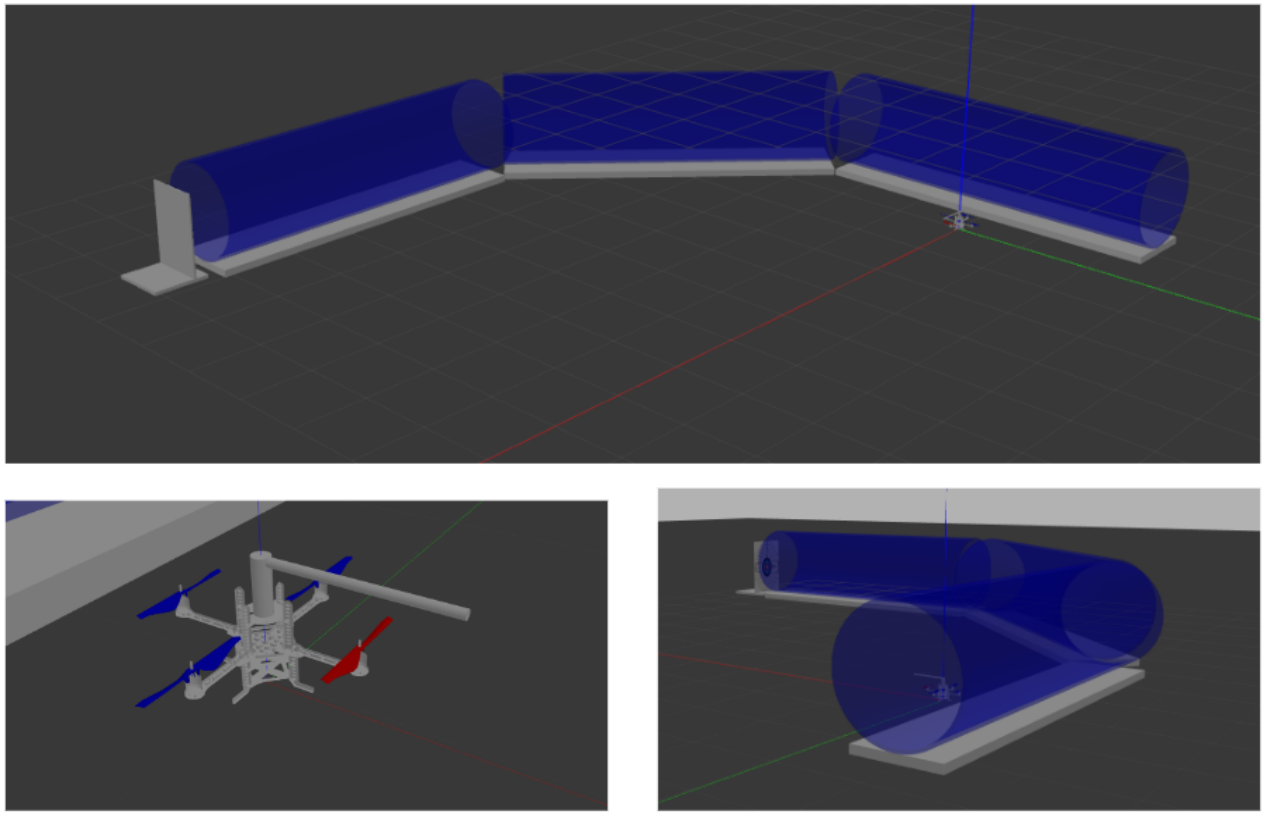
UDP Tunnels


* $\tau = d_1 + d_2$
Position Predictor
$\hat{x}(k - d_1 + \tau)$
$x(k - d_1)$
Teleoperation
$\tilde{x}^d(k)$
Controller
UDP tunnel
$u(k - d_1)$
Network
UDP tunnel
$x(k)$
Sensors
LLC
$u(k - d_1 - d_2)$
CLOUD CLUSTER
AERIAL ROBOT

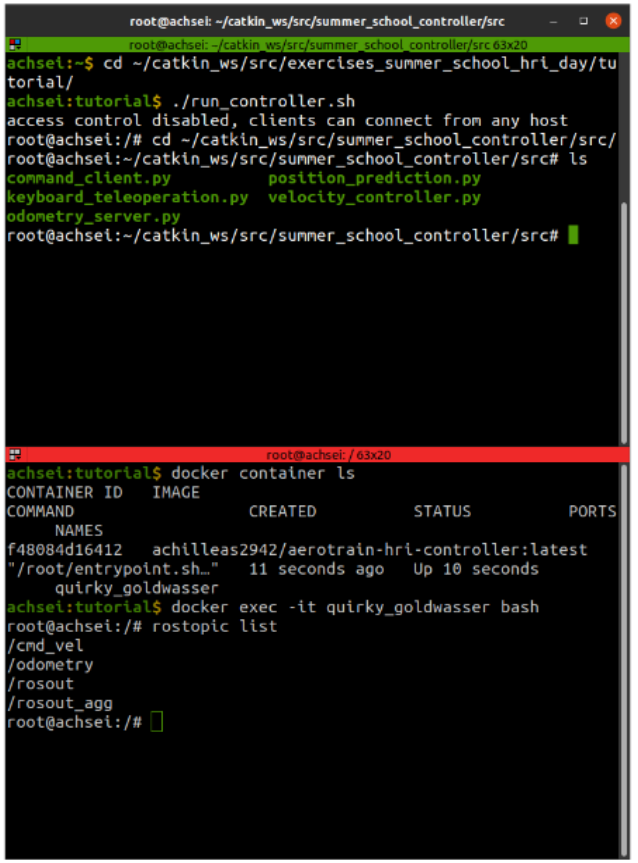
Simulation


* $\tau = d_1 + d_2$
Position Predictor
$\hat{x}(k - d_1 + \tau)$
$x(k - d_1)$
Teleoperation
$\tilde{x}^d(k)$
Controller
$u(k - d_1)$
UDP tunnel
Network
UDP tunnel
$x(k)$
Sensors
LLC
$u(k - d_1 - d_2)$
Docker container 2: Controller
Docker container 1: Gazebo world

Simulation



Docker container 1: Gazebo world



Docker container 2: Controller

Gazebo World Container

Navigate to the /exercises_summer_school_hri_day/tutorial directory.

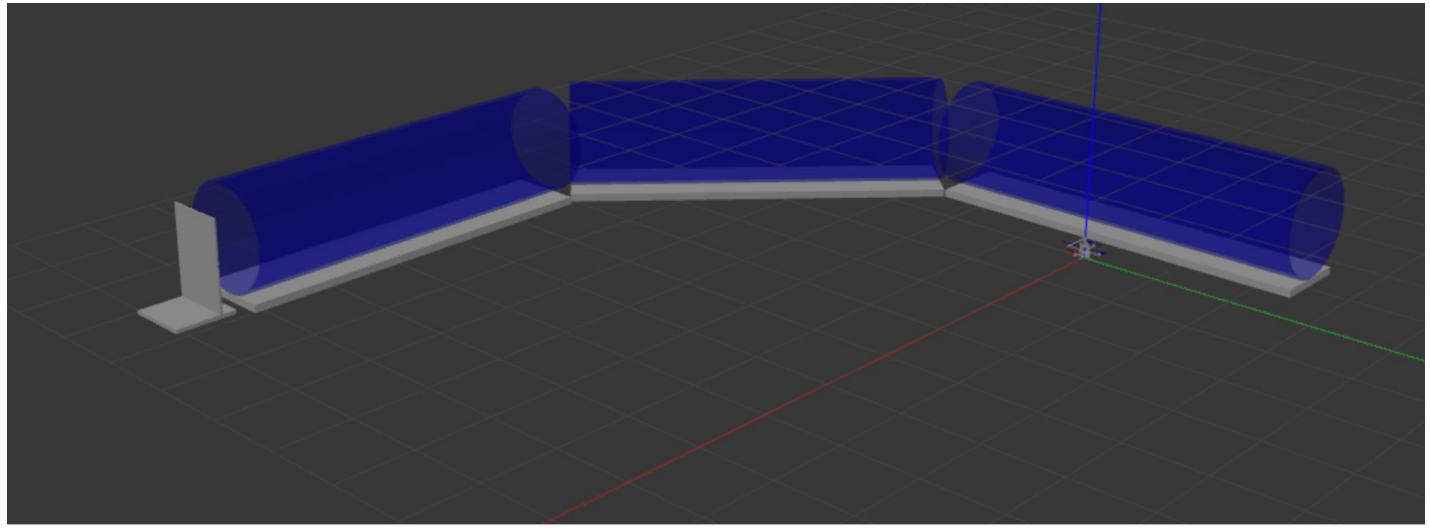
Make the run_gazebo.sh file executable.

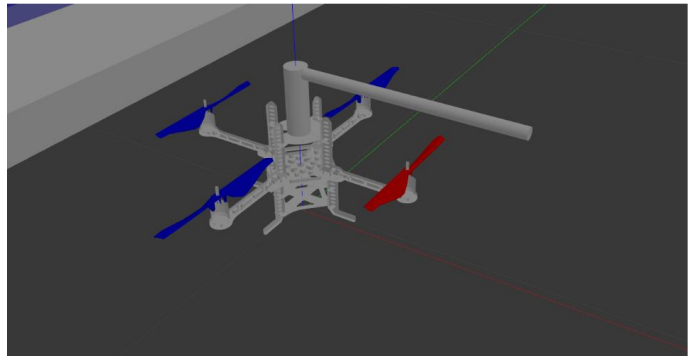
```
chmod +x run_gazebo.sh
```

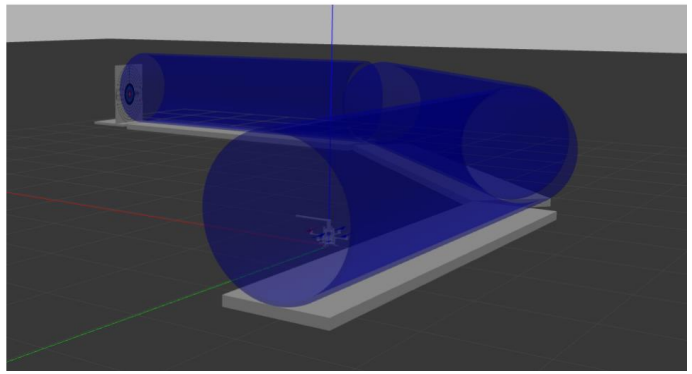
Launch the Gazebo world.

```
./run_gazebo.sh
```

When the environment is built, the simulation shown in the figure appears on your screen.







NOTE! The Gazebo simulation is paused by default.

Controller Container

Navigate to the /exercises_summer_school_hri_day/tutorial directory.

Make the run_controller.sh file executable.

```
chmod +x run_controller.sh
```

Launch the controller environment.

```
./run_controller.sh
```

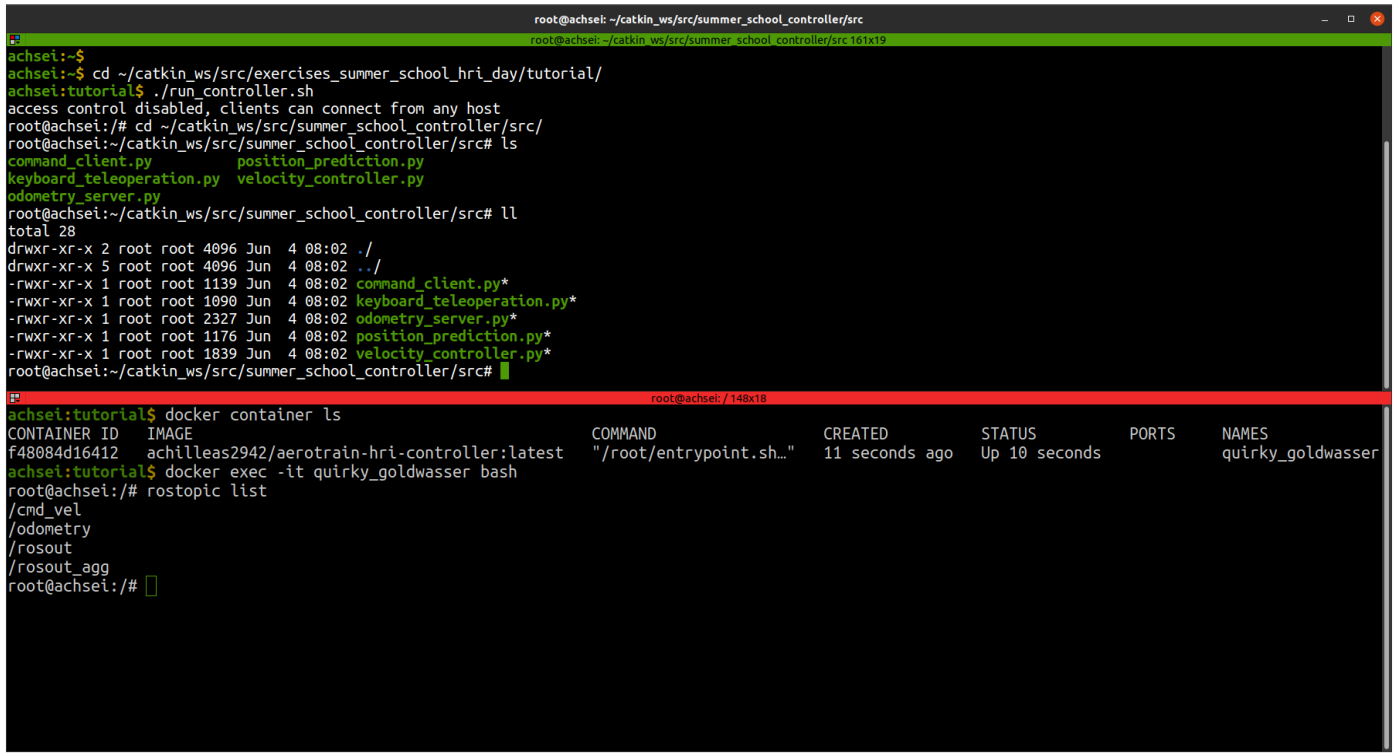
Extra controller container terminals.

```
docker container ls

docker exec –it <container> bash
```

For the exercise navigate to

```
cd ~/catkin_ws/src/summer_school_controller/src
```



Tutorial

Aim of the tutorial

The aim of this workshop is to design a basic teleoperation interface to navigate a quadrotor UAV to a desired position.

Goal of the tasks

The goal is to navigate the UAV through the three pipes to establish smooth contact with the target.

Task 1 – UAV Position Control

Design a position controller for the UAV.

- Open the *velocity_controller.py* file with your preferred editor (*gedit, nano, vim*).
- Under the *velocity_publisher()* method in the *VelocityController* class, please insert your code to send velocity inputs to track the position setpoints.
- You may publish waypoints on the /setpoints_position topic to test the tracking performance of the controller.

Task 1 – UAV Position Control (solution)

Indicative solutions can be found in the solutions branch

```
git checkout solutions
nano ~/catkin_ws/src/summer_school_controller/src/velocity_controller.py
```

Task 2 – Teleoperation Interface

Enable a Teleoperation Interface.

- Please open the *keyboard_teleoperation.py* with your preferred editor (*gedit, nano, vim*).
- Under the *on_press()* method in the *DroneTeleoperator* class, please insert your code to read keyboard inputs and publish position setpoints.
- Test it with the *velocity_controller.py* to navigate the UAV to different positions.

Task 2 – Teleoperation Interface (solution)

Indicative solutions can be found in the solutions branch

```
git checkout solutions
nano ~/catkin_ws/src/summer_school_controller/src/keyboard_teleoperation.py
```

Enable Delays

Delete netem

```
tc qdisc del dev lo root
```

Add a root qdisc

```
tc qdisc add dev lo root handle 1: htb default 10
```

Add a class

```
tc class add dev lo parent 1: classid 1:1 htb rate 100mbit
```

Add a delay

```
tc qdisc add dev lo parent 1:1 handle 10: netem delay 50ms
```

Apply the netem

```
tc filter add dev lo protocol ip parent 1:0 prio 1 u32
match ip protocol 17 0xff flowid 1:1
```

Change the netem

```
tc qdisc change dev lo parent 1:1 handle 10: netem delay
40ms 5ms
```

Task 3 – Delay Compensation

Compensate for the Communication Delays.

- Please open the *position_prediction.py* with your preferred editor (*gedit, nano, vim*).
- Under the comment *#estimate_delay* comment in the *PositionPredictor* class, please insert your code to estimate the average delay from the ROS messages.
- Under the *#predict_position* comment in the *PositionPredictor* class, please insert your code to estimate the current position of the UAV from the available knowledge of the delays.
- Test it with the *velocity_controller.py* and *keyboard_teleoperation.py* to navigate the UAV through the pipes.

Task 3 – Delay Compensation (solution)

Indicative solutions can be found in the solutions branch

```
git checkout solutions
nano ~/catkin_ws/src/summer_school_controller/src/position_prediction.py
```

Available Material

Gazebo World

Controller Environment

Thank you :)

Gerasimos Damigos
gerasimos.damigos@ericsson.com

Viswa Narayanan Sankaranarayanan
viswa.narayanan.sankaranarayanan@ltu.se

Achilleas Santi Seisa
achilleas.seisa@ltu.se

AERO TRAIN