INF142 Computer Networks

Mandatory assignment
Deadline: 09.04.2021

1 Introduction

The Fictional Meteorological Institute (FMI) is modernizing its infrastructure. Currently, the institute counts on three, rather remote, weather stations for collecting data about temperature and precipitation. A decision has been made to fully automate the data collection process and to deploy two intermediate servers for storage, see Figure 1.

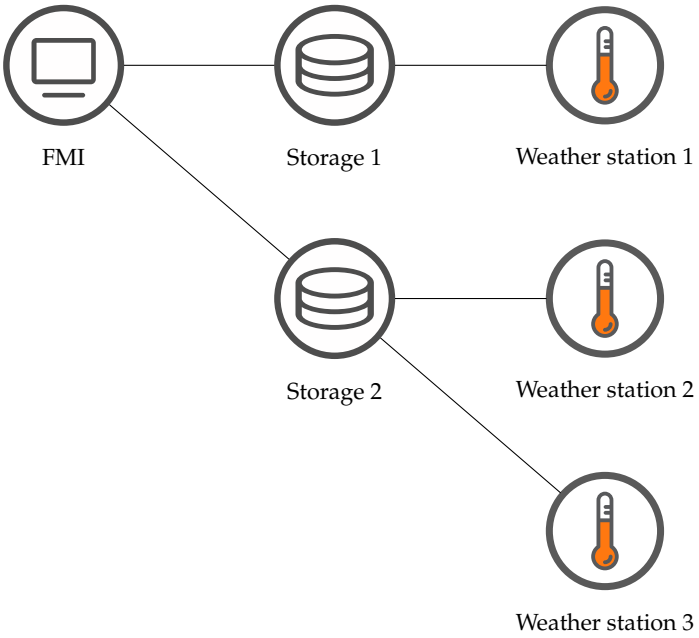

Figure 1: The future of FMI.

A team of engineers is responsible for the deployment of the necessary hardware, whilst your team develops the corresponding distributed application. To save power and resources, it is agreed that data *shall not* persist in the devices connected to the weather sensors in weather stations. Instead, these devices must only allow for either remote readings of the data or its periodical transmission. Therefore, storage servers must either read the data on a regular basis from the weather stations or to wait for new data. At FMI, a host must be always running a user agent to allow users to retrieve data from all the storage servers. At the moment, your team is just prototyping the app.

You will count on the extremely valuable support of two tech leads (your mentor TAs) and, from time to time, you can also request the service of an external consultant (the course instructor). The leadership of FMI expects a minimum viable product (MVP) of your app in Python no later than 09.04.2021.

2 Tasks

As a preliminary step, note that the architecture in Figure 1 can be simplified as in Figure 2, where FMI, Storage and Weather station are processes instead of end systems.

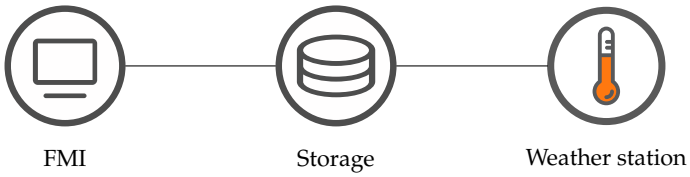

Figure 2: Simplified model.

Your application will be considered a MVP if it fulfills all of the following requirements:

- It consists of at least three Python scripts, one for each of the aforementioned processes.
- At least one TCP socket is used.
- At least one UDP socket is used.
- The provided script `station.py` is used by the *weather station process* to simulate the readings of weather sensors. Note that you only need to import a class from this script. You do not need to understand the code in it.
- The *storage server process* periodically stores data in a file or database.
- The *storage server process* provides remote access to the stored data.
- The *FMI process* (user agent) runs in a CLI and, upon request, displays *all* the data available in the storage server.

The three processes can run, without any isolation, in the same device by using localhost. Note that this is the minimum you have to accomplish. Of course, your tech leads encourage to do more and to find inspiration in the tasks for the group sessions #4, #7 and #8.

3 Adding some extras

If you wish, you can try to impress the FMI leadership by delivering more than just the MPV. However, it is of utmost importance that you meet the deadline and that adding features to your app will not demand too much from you. These extras should be added only if at least some of your teammates are comfortable with their usage and/or implementations. Do not add more than two.

As a byproduct of adding these features, your application architecture can differ from that in Figure 2 as long as yours uses more than three processes. For instance, by using Flask and Docker, the external consultant implemented a distributed application whose architecture is illustrated in Figure 3.

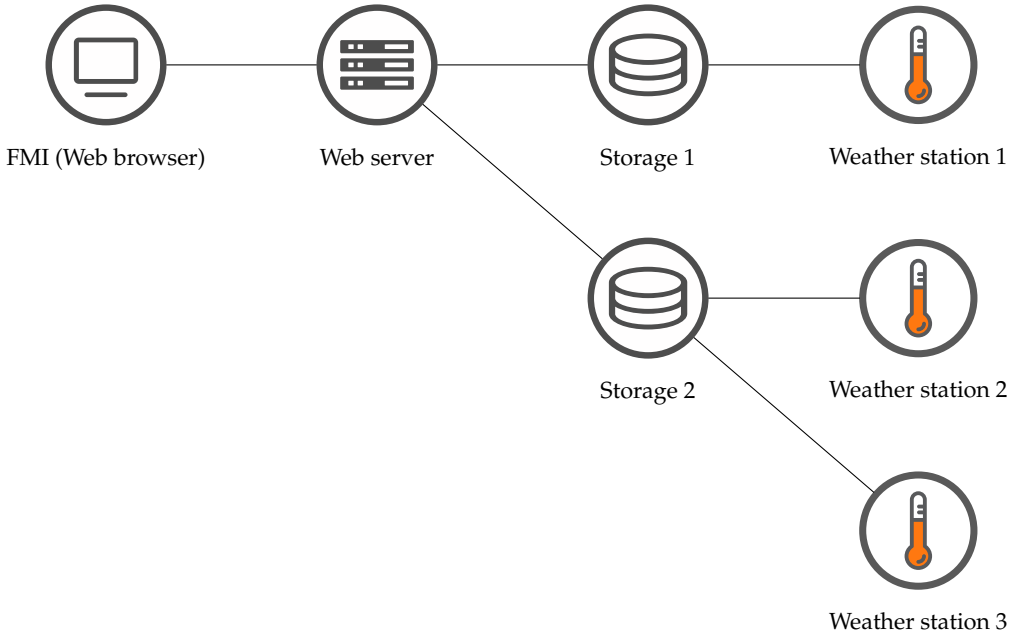

Figure 3: External consultant's app.

These are some features that you can add or tools that you can use:

- Requests.
- Flask.
- Docker (and Docker Compose).
- SQL.
- MongoDB.
- A GUI.

Talk to your tech leads if you want to add something else.

What follows is sensitive information. Therefore, the role game is unfortunately over. We hope you have fun and learn something from this mandatory assignment.

4 Rules against cheating

- Cooperation between teams is encouraged. You can exchange ideas, suggestions, etc. but not code. If we find two codes that are partially the same, both teams will get zero points.
- If a team cannot explain the code at all, the team will get zero points.
- If a teammate stops contributing, the team must report this to the mentors. If this is not reported, the team can risk failing the assignment. The way in which a team distributes the workload is up to the team. However, all its members must be able to explain every line of code. A couple of meetings, discussing who did what, should be enough for every teammate to understand what others did.

5 Your grade

The grade for the mandatory assignment will consist of the following components:

- **7 pt.** Handing in three simple Python scripts that function as required in the assignment.
- **5 pt.** Adding some extra features or tools.
- **8 pt.** The answer to our questions about your code, the extra features (if there are some) and the concepts in computer networks related to them.

After you have handed in your assignment, we will schedule a meeting with each team where each teammate will be asked to explain parts of the code. The way in which you hand in the assignment is up to your mentor TAs. The mandatory assignment will be graded by the mentors and the instructor.

Our strategy to deal with the current situation consists in giving you more than enough time and support for the assignment. Therefore, extensions will be granted only to students in rather exceptional situations. In case of sickness you must present a certificate from your GP.