Middleware Technologies for Distributed Systems

Politecnico di Milano
Luca Mottola and Alessandro Margara

Exam Projects 2021/2022
v.0.2

General Rules and Guidelines

1. Projects and demos must be developed in groups of **exactly three students**. Any exception to this rule must be explicitly approved by Luca Mottola, or it won't be possible to take the exam.
2. Those who are part of groups of only two people or did not find a group to join should contact Luca Mottola <luca.mottola@polimi.it> by **December 1st, 2021**. This applies regardless of when in the academic year you plan to work on the projects and demos.
3. The projects and demos described below are valid for this academic year only. This means that they must be submitted **before the last official exam session of this academic year**.
4. You are to select a combination of projects and demos that, taken together, **covers all technologies** discussed in the course, excluding aerial drones. Work with the latter is described in the next bullet. You can only cover one technology with a demo. For example:
    - It is **acceptable** if you use Contiki-NG, MPI, and Spark for project #1, Contiki-NG, Node-red, and Kafka for project #2, and implement a demo for Akka.
    - It is **not acceptable** if you use Contiki-NG, MPI, and Spark for project #1, Contiki-NG, and Node-red for project #2, and implement a demo for Akka, because you are not using Kafka anywhere.
    - It is **not acceptable** if you use Contiki-NG and Spark for project #1, Contiki-NG and Kafka for project #2, and implement a demo for Akka and Node-red, because you cannot implement more than one demo.
5. Projects **in mobile robotics** are going to be defined as small research projects and be carried on the FlyZone testbed at NESLab <www.neslab.it>. The number of such projects is limited;

groups interested in these projects should contact Luca Mottola <luca.mottola@polimi.it> as soon as possible. These projects are valid as a wildcard to cover **any three other technologies**.

6. Any question or clarification is to be posted on WeBeep. We may update this document over time in case further information is needed.
7. The possibility remains for students interested in carrying out their thesis work with either instructor, to skip the project corresponding to their thesis topic. This option must be discussed with the relevant instructor beforehand.

Rules and Guidelines for Projects

1. You are expected to demonstrate your project implementations working in an **actual distributed setting**; for example, using
    - two or more laptops or virtual machines or...;
    - cloud deployments;
    - network simulators whenever applicable.
2. The design and implementation must be thought as if the system was actually **deployed in real-world conditions**. Simplifications that only apply to the specific demonstration shown for the exam are not acceptable.
3. You are expected to present the software architecture and the algorithms / protocols implemented in the projects. You can use a few slides to support their presentation. Preparing slides is, however, **not mandatory**, as long you are well prepared to present your work.
4. Each student in a group is expected to answer **questions on any part of any project** developed by the group. This requires knowledge of design principles, implementation choices, and testing procedures of any project developed by the group by any of its members.
5. Discussion of the projects is to be carried out by scheduling an appointment with the relevant instructor. Please drop an email to Luca Mottola <luca.mottola@polimi.it> or Alessandro Margara <alessandro.margara@polimi.it> for this.
6. When asking for an appointment, you also need to **attach the complete source code of all projects and a short design document (~4 pages) for each project**, prepared according to the template distributed by the lecturers.

Rules and Guidelines for Demos

1. Demos are **not** to be demonstrated; sending the code with a short README file in TXT format is sufficient as long as the teacher can run the demo on his machine.

2. There will be **no discussion** for demos, but they must be submitted before the project discussion takes place, so the teacher may ask you questions if required.

Project #1: Simulation and Analysis of Noise Level

You are to design and implement a system that studies the level of noise in a country. The country is divided into regions: some of them provide open access to noise data from sensors, while others do not. The system uses sensor data in regions where they are available, and data generated by a computer simulation otherwise. In the design of your data model, you can take inspiration (or even reuse data) from open datasets such as sensor.community (https://sensor.community/en/).

Sensor data

Sensors attached to mobile IoT devices (e.g., smart watches) measure the level of noise every T=10 sec. A sliding window is applied that computes the average of the last six readings. The readings are reported to the backend on the regular Internet, through static IoT devices acting as IPv6 border routers. Should the value of the average exceed a certain threshold K, the raw readings are reported instead of the average obtained from the sliding window.

Noise levels

In regions where sensor data is not available, the system relies on computer simulations and/or virtual software-emulated sensors.

Simulations are based on population dynamics. The simulation considers the noise produced by people and vehicles. It takes in input the following parameters:

- P = number of people
- V = number of vehicles
- W, L = width and length of the (rectangular) region where individuals move (in meters)
- Np = level of noise (dB) produced by each person
- Nv = level of noise (dB) produced by each vehicle
- Dp = distance (side of the square area, in meters) affected by the presence of a person
- Dv = distance (side of the square area, in meters) affected by the presence of a vehicle
- Vp = moving speed for an individual
- Vv = moving speed for a vehicle
- t = time step (in seconds): the simulation recomputes the position of people and vehicle, and the level of noise of each square meter in the region with a temporal granularity of t (simulated) seconds

Virtual sensors, by contrast, emulate the behavior of IoT devices. They implement the exact same application logic, but the sensor data comes from a sensor trace on a file that is either generated a priori with random data, or from an existing open dataset.

Data cleaning and enrichment (optional)

The backend receives noise level data from real IoT sensors, computer simulations, and/or virtual sensors. Each noise level data is annotated with the geographic area where it was measured. The backend performs a pre-processing step before storing the data:

1. it discards invalid/incomplete measurements, where either the level of noise or the geographic area are missing or invalid (e.g., a noise level below zero) due to malfunctioning sensors;
2. it associates each reading with the name of the nearest point of interest (e.g., the name of a place, road, building); you can assume that the names and locations of points of interest are stored in a (relatively small) static dataset.

Data analysis (optional)

The backend periodically computes the following metrics:

1. hourly, daily, and weekly moving average of noise level, for each point of interest;
2. top 10 points of interest with the highest level of noise over the last hour;
3. point of interest with the longest streak of good noise level;
    - given a maximum threshold T, a streak is the time span (in seconds) since a point of interest has last measured a value higher than T;
    - by definition, a point of interest with a value above T has a streak of 0;
    - at any given point in time, the query should compute the point (or points) of interest with the longest stream.

Note: in case both "Data cleaning and enrichment" and "Data analysis" are not implemented, sensor data coming from the IoT network and/or simulated must at least be dumped on a CSV file on a central machine.

Practical considerations

- Sensor data may be faked to show specific executions of interest; for example, a case where the average of noise readings grows above K.

Project #2: Smart Buildings and Neighborhoods

You are to design and implement a system that manages sensors and actuators deployed in several buildings of possibly different neighborhoods. The system should orchestrate simple control loops within the same room of a building, while collecting and analyzing long-term environmental data from the available sensors. In the design of your data model, you can take inspiration (or even reuse data) from open datasets such as sensor.community (https://sensor.community/en/).

Control loops

Sensors deployed in a room periodically monitor environmental data, such as temperature and humidity, and use this data to trigger commands on nearby actuators. For example, if either temperature or humidity fall outside an interval considered to provide sufficient comfort to the inhabitants, an HVAC controller in the same room is turned on until the sensor data is back within the comfort ranges. Sensors are IoT devices, whereas actuators are more powerful devices able to run a full-fledged OS. The control logic may be deployed on the IoT devices, on the actuators, or on both.

Data collection and analysis (optional)

Sensors periodically funnel environmental data to the back-end, where it is stored and eventually used to compute the following statistics:

1. hourly, daily, and weekly moving average of each environmental quantity, at room-, building, and neighborhood-scale;
2. daily night-day temperature difference at room- and building-level; you can assume the night temperature to be the average temperature between 8pm and 8am, and the day temperature to be the average temperature between 8am and 8pm;
3. month of the year with the highest average night-day temperature difference, for each building.

Practical considerations

- Sensor data may be faked to show specific executions of interest; for example, a case where the HVAC is first turned on and then turned off.
- The identities or addresses of sensors and actuators in the same room cannot be hardcoded or used statically in the code.
- The operation of the HVAC may be emulated by printing something out on the console.

Project #3: Compute Infrastructure

You are to design and implement a system that accepts compute tasks from clients and executes them on a pool of processes. Each request from a client includes the name of the task to be executed (e.g., image compression), a payload with the parameters to be used in the computation (e.g., a set of images and a compression ratio), and the name of a directory where to store results.

Task scheduling

Clients submit compute tasks to a front-end, which can be implemented as a simple command-line application or a basic REST/Web app and get notified when they complete. Tasks are scheduled for execution onto a set of processes. Each process can handle one task at a time, so tasks may need to wait until some process is available. Processes may fail at any time and be restarted, but clients should not be aware of failures, that is, they need to be notified once and only once when the task completes and the results have been successfully stored on the disk.

Logging and analysis (optional)

The system logs information about tasks, which is used analyzed to extract the following statistics:

1. number of completed tasks for each month, week, day, hour (updated every minute);
2. number of pending tasks (updated every 5 seconds);
3. average number of times each task starts executing (updated every minute); Tasks may start executing more than once in the case of failures.

Practical considerations

- You can assume a finite set of possible tasks, each identified by a unique name (e.g., image compression, text formatting, ...). You can simulate heavy tasks using sleep primitives instead of implementing complete algorithms.

Demos

Akka

Make Exercise 5 (stream processor) fault-tolerant. You need to make sure that all operators at the same stage of the pipeline are resumed in a consistent way, such that each data item is processed at least once. This means that duplicate data items or multiple applications of the same operator to the same data item are acceptable. The demo must emulate a failure and show that consistency is achieved this way, so no data item is lost.

Node-red

Solve Exercise 5 (fictitious smart home) as stated in the lecture slides. Assume that a topic exists via MQTT called /home1/heating where published messages are delivered to the heating system. The message payload only includes a single boolean value, which holds true in case the heating must be increased, or false if it needs to be decreased. The control loop should run periodically, every 5 minutes.

MPI

Modify Exercise 7 (character count) to count words instead of characters

- The program still mimics the MapReduce programming model
- First, each process reads a (different) input file and computes the number of occurrences of each word in that file
- Second, words are partitioned across processes, and each process reduces the partial results for the words it is responsible for
- Third, the final results are delivered to a single node, which writes them to a single file
- You can assume the maximum length of words to be known at compile time
- You can write your program in C++ and use the strings and collections data types of the standard library

FAQ

Q: In project 1, is it possible to emulate different regions with different technologies, say three regions with three different technologies?
A: Yes, that is totally fine. You can emulate as many regions as you want using any technology you feel appropriate. Design decisions must be motivated, like for everything else.

Q: If we implement the projects using a combination of technologies that already cover them all, do we need to implement a demo anyways?
A: No, you are done. You must cover all technologies, using at most (not exactly) one demo.

Q: If we use the same technology in different projects, do we get a higher grade?
A: Not necessarily. The mere fact of using the same technology multiple times is not an added value. Design decisions must be motivated and justified, that is what matters.