Middleware Technologies for Distributed Systems

Politecnico di Milano
Luca Mottola and Alessandro Margara

Exam Projects 2020/2021
v.0.1

General Rules and Guidelines

1. Projects must be developed in groups of **exactly three students**. Any exception to this rule must be explicitly approved by Luca Mottola.
2. Students who are currently part of groups of only two people or did not find a group to join should contact Luca Mottola <luca.mottola@polimi.it> by January 22th, 2021. This applies regardless of when in the academic year you plan to work on and/or present the projects.
3. If you are not part of a group of three students and did get approval that you can work in a group of two or alone, **you cannot take the exam**. Again this applies regardless of when in the academic year you plan to work on and/or present the projects.
4. The projects described below are valid for this academic year, only. This means that they must be presented and discussed with the course instructors **before the last official exam session of this academic year**.
5. Students are expected to demonstrate their projects working in an **actual distributed setting**; for example, using
    - two or more laptops (or virtual machines);
    - cloud deployments;
    - network simulators whenever applicable.
6. Students are expected to present the software architecture and the algorithms / protocols implemented in the projects. They can use a few slides to support their presentation. Preparing slides is, however, **not mandatory**, as long as students are well prepared to present their work.
7. You are to select a combination of projects that, taken together, **covers all 6 technologies** discussed in the course.

8. 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.
9. 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.
10. 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**, illustrating the main design and implementation choices.
11. Any question or clarification required on the project description below is to be posted to the public Beep forum of the course. We may update this document over time in case further information is needed.
12. 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.

Project #1: Contact Tracing with Body-worn IoT Devices (Luca Mottola)

Description

People roaming in a given location carry IoT devices. The devices use the radio as a proximity sensor. Every time two such devices are within the same broadcast domain, that is, at 1-hop distance from each other, the two people wearing the devices are considered to be in contact. The contacts between people's devices are periodically reported to the backend on the regular Internet. Whenever one device signals an event of interest, every other device that was in contact with the former must be informed.

Assumptions and Guidelines

1. The IoT devices may be assumed to be constantly reachable, possibly across multiple hops, from a single static IoT device that acts as a IPv6 border router, that is, you don't need to consider cases of network partitions.
2. The IoT part may be developed and tested entirely using the COOJA simulator. To simulate mobility, you may simply move around nodes manually.
3. Notification at a target device may be accomplished in simple ways, for example, by turning on a LED or printing something out on the serial console.

Technologies

ContikiNG + one other technology of choice for the backend processing

Project #2: Distributed Node-red Flows (Ref. Luca Mottola)

Description

You are to implement an architecture that allows Node-red flows to span multiple devices. Normally, a Node-red flow executes locally to the machine where it is installed. Instead, consider multiple Node-red installations that i) register to a central repository that maintains information on all running installations and ii) can exchange messages among them by logically connecting the output of a node in one installation to the input of another node in a different installation. Addressing of Node-red installations must be content-based, that is, the target Node-red installations that receive the messages cannot be determined based on their IP address or some other form of machine-level identifier. You need to demonstrate that a flow developed to be executed on a single Node-red installation may be split across two or multiple Node-red machines with a limited set of modifications to the flow itself.

Assumptions and Guidelines

1. The project may be accomplished without creating custom nodes, that is, creating custom nodes is welcome but not mandatory.
2. The set of running Node-red installations is dynamic, that is, your implementation must be able to manage Node-red installations coming and going while the application executes.
3. Your solution must be agnostic to the content of messages, that is, it must be usable regardless of the format of messages or the nature of nodes that produce/consume them.

Technologies

Node-red + one other technology of choice for the coordination among Node-red installations

Project #3: A Simple Smart Home (Ref. Luca Mottola)

Description

In a smart home, a control panel provides a user interface to coordinate the operation of several home appliances, such as HVAC systems, kitchen machines, and in-house entertainment, based on environmental conditions gathered through sensors. Users input to the control panel their preferences, *e.g.*, the desired room temperature. Based on information returned by every appliance, the control panel offers information on the instantaneous energy consumption over the Internet. You are to implement the software layer for the smart home using the actor model. Every appliance may come and go depending on their operational times. The processes in charge of managing every appliance may also crash unpredictably. You need to demonstrate at least three example executions that include i) the user inputting some preferences, ii) sensors producing some (dummy) values, and iii) appliances changing their behavior based on i) and ii). The three example executions must be able to operate in parallel in two or more rooms of the same smart home and tolerate process crashes.

Assumptions and Guidelines

1. Although processes can crash, you can assume that channels are reliable.
2. You are allowed to use any Akka facility, including the clustering/membership service.
3. Sensors, appliances, and the control panel may be emulated via software with dummy data.

Technologies

Akka

Project #4: A Simple Model for Virus Spreading (Ref. Alessandro Margara)

Description

Scientists increasingly use computer simulations to study complex phenomena. In this project, you have to implement a program that simulates how a virus spreads over time in a population of individuals. The program considers N individuals that move in a rectangular area with linear motion and velocity v (each individual following a different direction). Some individuals are initially infected. If an individual remains close to (at least one) infected individual for more than 10 minutes, it becomes infected. After 10 days, an infected individual recovers and becomes immune. Immune individuals do not become infected and do not infect others. An immune individual becomes susceptible again (i.e., it can be infected) after 3 months.

The overall area is split into smaller rectangular sub-areas representing countries. The program outputs, at the end of each simulated day, the overall number of susceptible, infected, and immune individuals in each country. An individual belongs to a country if it is in that country at the end of the day.

A performance analysis of the proposed solution is appreciated (but not mandatory). In particular, we are interested in studies that evaluate (1) how the execution time changes when increasing the number of individuals and/or the number of countries in the simulation; (2) how the execution time decreases when adding more processing cores/hosts.

Assumptions and Guidelines

1. The program takes in input the following parameters
    - N = number of individuals
    - I = number of individuals that are initially infected
    - W, L = width and length of the rectangular area where individuals move (in meters)
    - w, l = width and length of each country (in meters)
    - v = moving speed for an individual
    - d = maximum spreading distance (in meters): a susceptible individual that remains closer than d to at least one infected individual becomes infected
    - t = time step (in seconds): the simulation recomputes the position and status (susceptible, infected, immune) of each individual with a temporal granularity of t (simulated) seconds
2. You can make any assumptions on the behavior of individuals when they reach the boundaries of the area (for instance, they can change direction to guarantee that they remain in the area)

Technologies

MPI or Apache Spark

Project #5: Analysis of COVID-19 Data (Ref. Alessandro Margara)

Description

In this project, you have to implement a program that analyzes open datasets to study the evolution of the COVID-19 situation worldwide. The program starts from the dataset of new reported cases for each country daily and computes the following queries:

1. Seven days moving average of new reported cases, for each county and for each day
2. Percentage increase (with respect to the day before) of the seven days moving average, for each country and for each day
3. Top 10 countries with the highest percentage increase of the seven days moving average, for each day

You can either use real open datasets[1] or synthetic data generated with the simulator developed for Project #4.

A performance analysis of the proposed solution is appreciated (but not mandatory). In particular, we are interested in studies that evaluate (1) how the execution time changes when increasing the size of the dataset and/or number of countries; (2) how the execution time decreases when adding more processing cores/hosts.

Assumptions and Guidelines

1. When using a real dataset, for countries that provide weekly reports, you can assume that the weekly increment is evenly spread across the day of the week.

Technologies

Apache Spark or MPI

---

1 https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide

Project #6: Food Delivery Application (Ref. Alessandro Margara)

Description

In this project, you have to implement a food delivery application. The application consists of a frontend that accepts requests from users and a backend that processes them. There are three types of users interacting with the service: (1) normal *customers* register, place orders, and check the status of orders; (2) *admins* can change the availability of items; (3) *delivery men* notify successful deliveries. The backend is decomposed in three services, following the microservices paradigm: (1) the *users* service manages personal data of registered users; (2) the *orders* service processes and validates orders; (3) the *shipping* service handles shipping of valid orders. Upon receiving a new order, the order service checks if all requested items are available and, if so, it sends the order to the *shipping* service. The *shipping* service checks the address of the user who placed the order and prepares the delivery.

Assumptions and Guidelines

1. Services do not share state, but only communicate by exchanging messages/events over Kafka topics
    - They adopt an event-driven architecture: you can read chapter 5 of the book "Designing Event-Driven Systems"[2] to get some design principles for your project
2. Services can crash at any time and lose their state
    - You need to implement a fault recovery procedure to resume a valid state of the services
3. You can assume that Kafka topics with a replication factor >= 2 cannot be lost
4. You can use any technology to implement the frontend (e.g., simple command-line application or basic REST/Web app)

Technologies

Apache Kafka

[2] Book available for free at: https://www.confluent.io/designing-event-driven-systems/