Every Ware Lab

UNIVERSITÀ DEGLI STUDI DI MILANO

Project Development and Design Tips

Lesson 2

Luca Arrotta and Riccardo Presotto

EWLab – Università degli studi di Milano

Professor: Claudio Bettini

These slides are based on previous versions created by Letizia Berolaja, Sergio Mascetti, Dario Freni, Claudio Bettini and Gabriele Civitarese

Copyright

Some slides for this course are partly adapted from the ones distributed by the publisher of the reference book for this course (Distributed Systems: Principles and Paradigms, A. S. Tanenbaum, M. Van Steen, Prentice Hall, 2007).

Every Ware Lab

The next steps

- Four lessons that aim at assisting the development of the project:
  - You can take advantage of these lessons to develop your project
  - You can clarify doubts or ask for advice

- The lessons have the usual timetable and the usual duration
- The first part consists of a short lecture
- Then, it is possible to work on your project

- It is mandatory that you work individually
  - This does not mean that you can't talk with your colleagues…
  - Two copied projects are EVIDENT

Every Ware Lab

Disclaimer

- The content of these slides has to be considered indicative
  - Development and design suggestions will be provided.
  - Each student can make different choices

- The directives on how to carry out the project are reported in the text of the project on the course website.

Every Ware Lab

Recommended Development flow I

- First step (today's lesson): REST server and SETA development
  - Design of the *Administrator Server* (resources and methods)
  - Synchronization problems analysis
  - Testing of the REST server with dedicated tools
  - *Administrator Client* Development
  - Implementation of SETA and the MQTT protocol to publish and receive orders

- Second step (second lesson): **Development of the taxis' network**
  - Architecture and protocols design of the peer-to-peer network of taxis
  - Insertion of a taxi in the peer-to-peer network
  - Rides management via a distributed and decentralized algorithm
  - Removal of a drone from the peer-to-peer network

Recommended Development flow II

- Third Step (Third lesson): Sensor data collection and local statistics
  - Implementation of the sensors data collection.
  - Computation and communication of the local statistics

- It is **crucial** to carefully consider both the internal synchronization and distributed synchronization problems

Every Ware Lab

The Taxis Network

- It is a **P2P** network
  - Each Taxi application is both client and server
- Each **Taxi** must be simulated through a **single process** (not a thread!)
- Each Taxi locally handles a representation of the network topology
  - It must be consistent among all the taxis
- It is mandatory to use **GRPC** to handle communications among taxis

Taxi Structure

- GRPC methods to handle messages
  - Is it better to keep open a **single bidirectional stream** or to provide different GRPC methods?
  - Is it better to **use synchronous or asynchronous** methods?

- A **thread** to handle the **standard input**
  - In this way, the taxi can leave the network in a controlled way

- A module to handle the **network topology** and the communication via **GRPC** with the other drones
- A module to handle the communication via **MQTT** with the Administrator server
- A module to handle the **PM10 sensor**
- A module to generate the **rides' statistics**

Every Ware Lab

Insertion of a taxi in the Network

- Once you start a taxi, it requests the *Administrator* Server to join the network

- If the registration ends successfully, the taxi receives its **position** in the smart-city, **the list of the other taxis**, and then:
  - it starts acquiring data from the **pollution sensor**
  - it has to **present itself** to the other taxis by sending them its position in the smart-city
  - It has to **subscribe to the MQTT topics** on which it is interested in

Every Ware Lab

Distributed synchronization #1

- Taxis must use a **distributed and decentralized algorithm** to decide who **will take charge** of each **ride**
  - Each ride will be handled by the Taxi (located in the same district of the ride's starting position) which meets the criteria indicated in the project
- The distributed and decentralized algorithm must be based on the ones introduced in the **theory lessons**
- Some options ….

Distributed synchronization #2

Option 1)

- **Ricart and Agrawala-based** algorithm
  - The disputed resource consists of a specific ride request
    - It changes every time a new ride request arrives
    - Every time a taxi acquires the “permission” to accomplish the ride, the related resource has to be discarded

Option 2)

- **Election-based** algorithm
  - A new master has to be elected every time a new ride request arrives
  - The elected master takes charge of accomplishing the disputed ride
    - The master loses its master role once it acquires the “permission” to accomplish the ride

Rides Management

“If in a district there is no Taxi that is available to take charge of a ride, such a ride must not be discarded “

- How to properly handle these rides?
  - Additional MQTT topics could be useful?

Every Ware Lab

Rides Management

"If in a district there is no Taxi that is available to take charge of a ride, such a ride must not be discarded "

- How to properly handle these rides?
  - Additional MQTT topics could be useful?
    - An MQTT topic to communicate to SETA the availability of a taxi in a specific district
    - An MQTT topic to communicate to SETA that a ride in a specific district has been accomplished
    - …..?

Every Ware Lab

Leaving the network

- We assume that taxis terminate only in a controlled way
  - When the message ”quit” is inserted by the command line
- To leave the system each taxi has to:
  - complete the possible ride it is involved in, and send to the Administrator Server the statistics described in the project
  - disconnect from the MQTT broker
  - complete any battery recharge
  - notify the other taxis of the smart city
  - request the Administrator Server to leave the smart city

Good Job!

Every
Ware
Lab