MINISTRY OF EDUCATION AND SCIENTIFIC RESEARCH

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

DISTRIBUTED SYSTEMS

Assignment 2

Asynchronous Communication

Sensor Monitoring System and Real-Time Notification

Tudor Cioara

Marcel Antal
Cristina Bianca Pop

Claudia Daniela Antal

2021

Contents

1. Requirements

The clients of the energy distributor have installed smart meters for each device registered to measure its energy consumption. Each sensor sends data to a server periodically, in the form *(timestamp, sensor_id, measurement_value)*, where *timestamp* is the time instance when the measurement was made and *measurement_value* is the value of the energy counter measuring the total energy consumed by the device in kWh since the sensor was installed.
Implement a system based on a message broker middleware that gathers data from the sensors and pre-processes them before storing them in the database. If the queue consumer application that preprocesses the data detects a measurement power peak that exceeds the sensor maximum threshold (i.e. sensor *maximum value* measure in kW defined in Assignment 1) it notifies asynchronously the client on its web interface. To compute a power peak, the instantaneous power in a measurement interval is computed by averaging the energy consumption and dividing the value to the time interval.

$$P_{peak}(t_1, t_2) = \frac{measurement_{value}(t_2) - measurement_{value}(t_1)}{t_2 - t_1} < MAX_{value}$$

A Sensor Simulator will simulate a sensor that reads data from files (sensor.csv), one value at every 10 minutes. The module will contain a timer synchronized with the local clock. The module sends data in the form < *timestamp, sensor_id, measurement_value* > to the message broker. The timestamp is taken from the local timer, the *measurement_value* is read from the file at the corresponding index, representing the energy measured in kWh, and the *sensor_id* is unique to each instance of the Sensor Simulator and corresponds to the sensor ID associated to a device of a client from the Energy Database.

The sensor simulator should be developed as a standalone application (i.e. desktop application) to read the sensor monitored activities from the file *sensor.csv,* configured as a message producer and send the monitored sample data to the queue defined. The file *sensor.csv* can be downloaded from https://dsrl.eu/courses/sd/materials/sensor.csv. The measurements are sent to the queue using the following JSON format:

```
{
    “timestamp" : 1570654800000,
```

"sensor_id": "5c2494a3-1140-4c7a-991a-a1a2561c6bc2"
"measurement_value" : 0.1,
}

1.1. Functional requirements:

- The message-oriented middleware allows the sensor system to send data tuples in a JSON format
- The message consumer component of the system processes each message and notifies asynchronously using WebSockets the client application

1.2. Implementation technologies:

- Use the following technologies: RabbitMQ, WebSockets.

2. Deliverables

- A solution description document (about 4 pages, Times New Roman, 10pt, Single Spacing) containing:
  a) Conceptual architecture of the distributed system.
  b) UML Deployment diagram.
  c) Readme file containing build and execution considerations.
- Source files. The source files and the database dump will be uploaded on the personal *gitlab* account created at the *Lab resources* laboratory work, following the steps:
  - Create a repository on *gitlab* with the exact name: *DS2021_Group_LastName_FirstName_Assignment_Number*
  - Push the source code and the documentation (push the code not an archive with the code or war files)
  - Share the repository with the user *utcn_dsrl*

3. Evaluation

3.1. Assignment Related Basic Questions:

During project evaluation and grading you will be asked details about the following topics:

- Message Oriented Middleware types
- Queue vs Topic
- Point-to-Point vs Publish Subscribe communication
- Server pushing data to clients: Sockets, WebSockets, Long Polling

3.2. Grading

The assignment will be graded as follows:

Points	Requirements
5 p	**Minimum to pass** • Implement application with 3 modules: message producer, message broker and message consumer

	• Display messages extracted from the queue • Documentation • Correct answers to 3.1 questions
2 p	Check if measurements exceed sensor limits when processing messages. Create WebSocket and push notifications to clients
2 p	Integration with assignment 1: register a client, create a device and a sensor. Set the sensor ID for the sensor simulator using a configuration file. Start the sensor simulator. View the data in the client page. Receive notifications in the client page for sensor exceeding maximum value.
1 p	Run at least two sensor simulators simultaneously and view the measurements on two client pages by opening the application in two browsers.

***NOTES:**

1. For the project component, you need to deploy on Docker or Heroku only the server part (database, REST API application and message broker – RabbitMQ). The sensor simulator should be run as a desktop application from your PC.
2. The sensor application should have a configuration file where you can set the sensor ID for the sensor associated to the client for which you test the application.

4. Bibliography

1. http://www.coned.utcluj.ro/~salomie/DS_Lic/
2. Lab Book: I. Salomie, T. Cioara, I. Anghel, T.Salomie, *Distributed Computing and Systems: A practical approach*, Albastra, Publish House, 2008, ISBN 978-973-650-234-7
3. Lab Book: M. Antal, C. Pop, D. Moldovan, T. Petrican, C. Stan, I. Salomie, T. Cioara, I. Anghel, Distributed Systems – Laboratory Guide, Editura UTPRESS Cluj-Napoca, 2018 ISBN 978-606-737-329-5, 2018, https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/329-5.pdf
4. https://spring.io/guides/gs/messaging-stomp-websocket/
5. https://www.rabbitmq.com/documentation.html