AY 2021/2022

Politecnico di Milano

Middleware Technologies for Distributed Systems

Project report of

Smart Buildings and Neighborhoods

Students
Matteo Beltrante Marco Bendinelli Simone Berasi

Supervisors
Luca Mottola Alessandro Margara

1 Introduction

1.1 General description

The project is about the development of a system that manages sensors and actuators, in the context of several buildings of possibly different neighborhoods. The system needs to be able to orchestrate simple control loops within the same room, while collecting long-term data from all available sensors.

1.2 Control Loops

Sensors periodically monitor data such as temperature and humidity, and use this data to trigger commands on nearby actuators. The HVAC controller should be turned on/off if values fall outside a comfort interval. Sensors are IoT devices, while actuators are able to run a full-fledged OS.

2 Architecture

We decided to use Kafka for the inter-rooms communication, while the sensor-actuator messaging system is implemented with MQTT.

2.1 IoT device - Sensors

As in project 1, the IoT devices are implemented with Contiki-NG os and they use the MQTT protocol to send data to the actuators. The `mqtt_client_process` consists on the implementation of a Finite State Machine that is periodically triggered by a timer: it handles the connection to the local Mosquitto broker and all the MQTT events. Moreover, it retrieves the readings by a virtual sensor that is included in the IoT device. The virtual sensor is a program that can generate different type of values within a default bound. The messages have topic = *iot/-houseID/roomID/sensorType*, where ID is an identifier for house and room, and sensorType is one between "temperature" and "humidity". These messages are published to a local broker that uses the bridge mode to connect with a remote broker.

2.2 Actuators

We assume to have a single sensor for every actuator in the room and we took into consideration two possible kind of actuators: one for temperature and one for humidity management. This system can be seen both as producer and consumer:

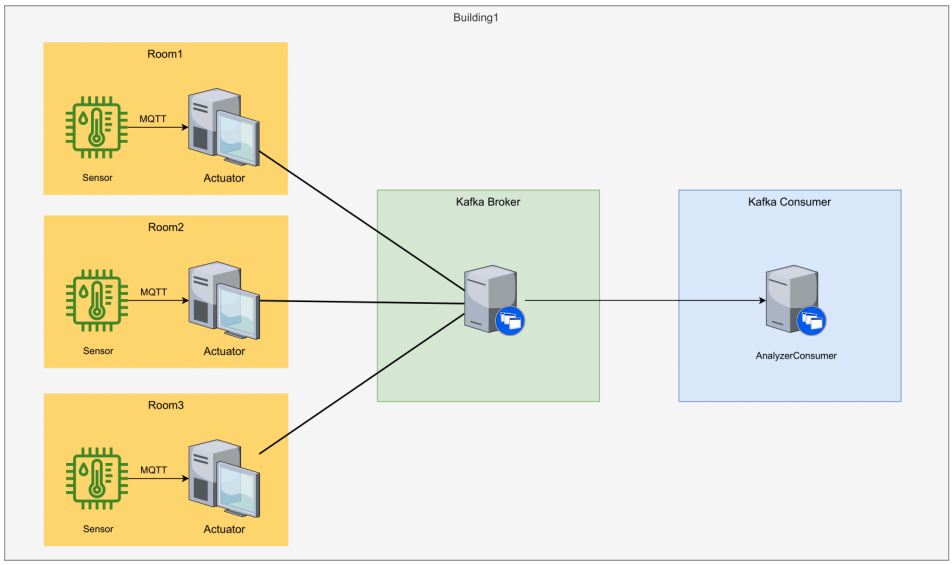

Figure 1: Simple architecture example for one building

it consumes the MQTT messages sent by the sensor and produces a Kafka event to be collected by the Analazyer.

2.3 Analyzer

This system is a Kafka consumer as it receives every message of every room of the whole system and stores it for long-term analysis.

2.4 Overview of the system

Starting from a value read by the IoT sensor (randomly generated in this case), the system does the following:

- An MQTT message is created and sent, with the following signature: *type:value:"datetime":datetime*
- The actuator system receives it and:
  - checks if the actuator is to be turned on/off;
  - builds a Kafka message, where the topic is one between *temperature* and *humidity*, the key tracks buildings and rooms and the value has the same signature as the MQTT one above.

- The analyzer receives the message produced above and save data in a file for future analysis.

3 Design Choices

- Being the sensors not able to run a full-fledged operating system, we decided to use MQTT to implement the communication between sensors and actuators;
- On the other hand, we chose Kafka to communicate between actuators and a central server collecting all the data. In the proposed version only one broker and one partition are used, this can be seen as a central processing device for every building. In case of massive data the project can be easily extended to more brokers and computing units.
- Overall we decided to use MQTT and Kafka because it is well known they work great together and where one fail the other excels.