Dept. of Computer Science
Télécom SudParis
IP Paris, France

Projet de Fin d'Etude ASR 2021 - 2022

PlanMosquitto: enhancing the Eclipse Mosquitto message broker with AI planning capabilities

September 17, 2021

Context: Smart Buildings with IoT devices

With the advent of the Internet of Things (IoT), buildings, homes, and spaces are becoming smarter and interconnected. The developed applications are often composed from different Quality of Service requirements (timeliness, accuracy, energy, etc.) to support application types such as: (i) *IoT analytics*, based on sensor readings for behavior analysis; (ii) *real-time*, for data acquisition within a limited latency bound; (iii) *transactional*, for request-response related applications; and (iv) *video streaming*, for bandwidth and data-intensive applications. IoT applications classified to the above categories exchange data with IoT devices via well known IoT protocols. MQTT is an OASIS standard messaging protocol for the IoT. It provides a lightweight method of carrying out messaging that follows the publish/subscribe interaction paradigm. To decouple IoT devices and applications, message brokers are often used for time and space decoupling. Eclipse Mosquitto is an open source MQTT-based message broker that is widely used for IoT data collection and dissemination. While such message brokers facilitate data exchange between devices and applications, it is not trivial to simultaneously support the aforementioned categories of IoT applications while meeting their QoS requirements.

We are currently designing an intelligent scheduler for transmitting IoT data flows to applications deployed in IoT-enhanced buildings while satisfying their QoS requirements. Messages from sensors are grouped into *IoT flows* based on the category of recipient applications and an *AI planner* computes an AI plan, i.e. a schedule that satisfies the QoS requirements of these applications. For instance, in the following AI plan (Fig. 1), at instant t_0, it is scheduled a flow for transaction to a printing machine and the data must be transmitted within 0.4ms; at instant $t_0 + 0.4$ms, the plan switches to a flow coming from a room occupancy device and its data must be transmitted within 0.10ms; at $t_0 + 0.5$ms, the plan switches to a flow coming from a robot and its data must be transmitted within 26.3ms, etc.

```
0.00000: (SCHEDULE_FLOW_TRANSACTIONAL PM1  TRANSACTIONAL_APP PRINTINGMACHINE_DEVICE)
0.00000: (TRANSMIT_FLOW_TRANSACTIONAL PM1  TRANSACTIONAL_APP PRINTINGMACHINE_DEVICE)
0.00040: (SCHEDULE_FLOW_TRANSACTIONAL RO1 TRANSACTIONAL_APP ROOMOCCUPANCY_DEVICE)
0.00040: (TRANSMIT_FLOW_TRANSACTIONAL RO1 TRANSACTIONAL_APP ROOMOCCUPANCY_DEVICE)
0.00050: (SCHEDULE_FLOW_REALTIME ROBOT1 REALTIME_APP ROBOT_DEVICE)
0.00050: (TRANSMIT_FLOW_REALTIME ROBOT1 REALTIME_APP ROBOT_DEVICE)
0.02680: (SCHEDULE_FLOW_REALTIME ROBOT4 REALTIME_APP ROBOT_DEVICE)
0.02680: (TRANSMIT_FLOW_REALTIME ROBOT4 REALTIME_APP ROBOT_DEVICE)
[...]
```

Figure 1: Example of an AI plan.

The **ultimate objective** of this project is to parse and execute an AI plan for **enabling IoT flow scheduling in the MQTT-based Mosquitto broker**.

Project Tasks

The selected team will have to work on the following tasks:

- Get familiar with the open source Mosquitto message broker (FYI, in Teaching Unit CSC5002, we study the AMQP protocol and the RabbitMQ broker, which are similar in principle);
- Get acquainted with the approach by reading a research paper that explains how the flows are grouped by category of recipient applications and how an AI planner computes AI plans;
- Model the internal micro-architecture of Mosquitto in order to explain how the subscription is managed, how a publication is managed, etc.;
- Design and implement a scheduler into Mosquitto using AI planning to control scheduling and transmission of IoT flows.

The most interesting and challenging tasks are the modeling, and the design and implementation.

Expected deliverables

1. Model of the internal of Mosquitto
2. Model of the added scheduler
3. Corresponding C code

Since Mosquitto is programmed in C language, the models are data structures, and in pseudo-code or temporal diagrams. This documentation is very important because the study will begin with this project. The C code to be written during the project is going to be licensed under a free license, a priori under the Eclipse Public License, which is the one chosen for Mosquitto.

Supervising Team

Georgios Bouloukakis, `georgios.bouloukakis AT telecom-sudparis.eu`

Denis Conan, `denis.conan AT telecom-sudparis.eu`

Eric Lallet, `eric.lallet AT telecom-sudparis.eu`