MBP: Not just an IoT Platform

Ana Cristina Franco da Silva
IPVS, University of Stuttgart
Stuttgart, Germany
franco-da-silva@ipvs.uni-stuttgart.de

Pascal Hirmer
IPVS, University of Stuttgart
Stuttgart, Germany
hirmer@ipvs.uni-stuttgart.de

Jan Schneider
IPVS, University of Stuttgart
Stuttgart, Germany
st117301@stud.uni-stuttgart.de

Seda Ulusal
IPVS, University of Stuttgart
Stuttgart, Germany
st116928@stud.uni-stuttgart.de

Matheus Tavares Frigo
IPVS, University of Stuttgart
Stuttgart, Germany
st165592@stud.uni-stuttgart.de

***Abstract*—In this demonstration paper, we introduce the Multi-purpose Binding and Provisioning Platform (MBP), an open-source IoT platform developed for easy binding, provisioning, and management of IoT environments. Furthermore, the MBP enables the simple realization of IoT applications, such as heating, ventilation, air conditioning (HVAC) systems, by allowing users to create rules for the IoT environment, in a straightforward and event-condition-action fashion. The efficient and timely data processing of IoT environments are assured through underlying complex event processing technologies.**

***Index Terms*—Internet of Things, Sensor Integration, IoT environments, Complex Event Processing**

I. Introduction

Since the popularization of the Internet of Things [1], many commercial and non-commercial IoT platforms were developed to help non-expert users with the management of IoT objects within their IoT environments. Examples of such IoT objects are sensors, actuators, or devices. However, the binding and provisioning of IoT objects to these IoT platforms are still very challenging tasks for non-expert users. In this context, *binding* means enabling IoT applications to access IoT objects on a higher level of abstraction, so that the required hardware expertise is kept at minimum.

In this demonstration paper, we present our open-source IoT platform, the Multi-purpose Binding and Provisioning Platform (MBP) [2], [3]. The MBP was primarily developed to easy the management of IoT environments, however, we move a step deeper into supporting users already during the binding and provisioning of IoT environments. In many IoT platforms, IoT objects are registered, bound, and provisioned to IoT platforms in a manual fashion. Such tasks are complex and require technical knowledge about the IoT objects. That is, operators (i.e., software code) to extract and provision sensor data to IoT applications, as well as to receive actuator control commands from IoT applications are required. Such operators need to be created and deployed for each sensor and actuator manually. Furthermore, monitoring functionality needs to be implemented and deployed manually as well.

This work is partially funded by the BMWi project IC4F (01MA17008)

However, deploying operators manually is error-prone and time-consuming, since a hardware expert has to configure IoT objects, install necessary operators for the specific sensors and actuators, bind them, and provide accessible interfaces to IoT applications. In real-world scenarios, e.g., for situation recognition, efficiency and accuracy requirements are of vital importance. However, these requirements cannot be fulfilled through manual binding and provisioning.

To tackle the aforementioned issues, the MBP was developed to support users in the complete life-cycle of IoT environments, so that the amount of manual tasks are kept to the possible minimum. This includes the automated deployment, management, and monitoring of IoT environments. In the following section, the MBP is explained in detail.

II. Overview on the MBP

The Multi-purpose Binding and Provisioning Platform (MBP) is an open-source platform, which is provided as a GitHub repository in [3]. The MBP has as primary goals the easy deployment, management, and monitoring of IoT environments. For this, the MBP provides a user interface, which is depicted in Fig. 1, as well as a REST API reflecting the same functionalities provided by the user interface. The main functionalities of the MPB are explained in the following.

A. Modeling IoT environments

IoT objects can be registered to the MBP either separately or as part of a specific IoT environment. The second option additionally enables the user to model the connections among IoT objects within the IoT environment. For this, the MBP provides a graphical tool for the modeling of IoT environments, as depicted in Fig. 2. In this tool, IoT objects including their properties and connections are specified. These properties describe specific information about the IoT objects, such as IoT object type, identifier, or MAC-address. To automate the modeling of IoT objects, the MBP provides QR code templates and an Android-based smartphone application to scan these QR codes and automatically fill properties that are possible to be inferred for the IoT object being modeled. Furthermore, following the same fashion, IoT objects can

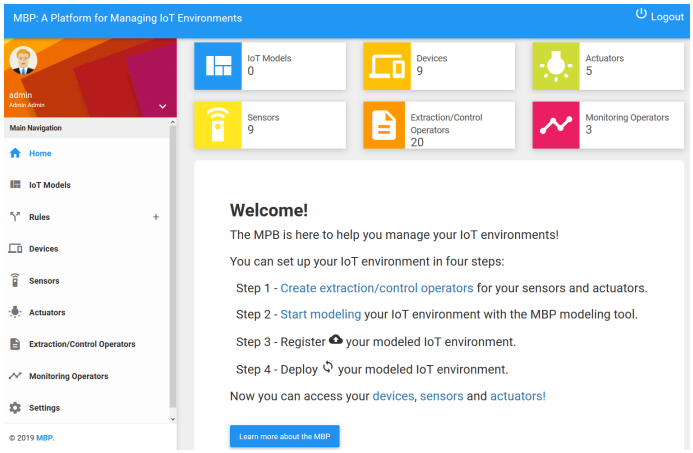

Fig. 1. The MBP UI

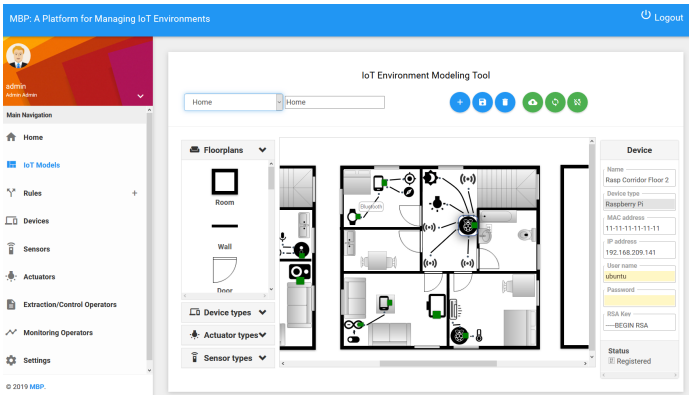

Fig. 2. The MBP UI - IoT environment modeling tool

be automatically discovered by connecting themselves to the *MBP registration hotspot*. Once an IoT environment is modeled and saved, it can automatically be registered with the MBP through the MBP UI. At this point, basic monitoring information about IoT objects can be already visualized in dashboards, such as network accessibility, CPU usage, and CPU temperature of an IoT device.

B. Deploying IoT environments

The MBP provides several ready-to-use extraction and control operators in the form of scripts. One exemplary extraction operator reads measurement values of an analog temperature sensor (TMP36 module) connected to a Raspberry Pi. This operator sends the extracted values to the MBP through MQTT, a publish/subscribe communication protocol. In the MBP, such operators can be simply linked to registered devices and, in sequence, be deployed onto these devices. Through this approach, sensors are bound automatically to the MBP and measured sensor values can be live-visualized immediately, and furthermore, are available as historical data. Nonetheless, the MBP also enables the users to provide their own operators, which can be implemented in any programming language. The MBP only requires the existence of specific lifecycle management scripts (e.g., install, start, or stop) for the operator, in order to be able to automate the deployment of the user-defined operators. These management scripts are executed by the MBP onto IoT devices at deployment time. The application logic of these management scripts can be still defined by the user, e.g., specifying necessary software that needs to be installed. In this way, users can create operators fulfilling their specific requirements, such as the use of specific hardware types or programming libraries.

C. Monitoring IoT environments

Once IoT objects are bound to the MBP, these IoT objects can be monitored by the users through provided dashboards. These dashboards show status and statistical information about IoT devices, sensors, and actuators. Furthermore, both live sensor data and historical sensor data can be visualized.

To realize an automatic monitoring based on sensor data, and furthermore, to trigger actuators automatically based on recognized conditions, the MBP provides the means for creating IoT applications based on user-defined rules. Such rules, which are based on the event-condition-action approach, can be defined on the graphical rule modeling tool integrated in the MBP. These high-level rules are then transformed into complex event processing (CEP) queries and are evaluated using corresponding CEP systems [4], which are able to provide an efficient, timely sensor data processing. An example of an IoT application that can be modeled with the MBP is a heating, ventilation, air conditioning (HVAC) system, in which the temperature of a room is continuously monitored in order to recognize and react when the temperature exceeds or goes below the thermal comfort zone.

D. MBP setup

The MBP can be installed by an installation script on Linux or it can be used as a docker container. After installation, the MBP user interface can be accessed through an internet browser (we recommend Chrome or Firefox). The MBP user interface is based on Bootstrap templates and is implemented in Angular JS. Furthermore, the data management and processing in the MBP is based on a lambda-based architecture, in which both live and historical data can be managed and processed. To store measurement data, i.e., sensor data and timestamp-based data, we use a time-series database. Further metadata are stored in a NoSQL database to clearly separate data and metadata. Furthermore, in [5], [6], we provide an approach realizing the placement of operators of a data stream processing model onto IoT environments. In this approach, requirements of operators (e.g., minimum required storage) and capabilities of IoT objects are searched and matched by algorithms for an optimal placement decision.

III. MBP Demonstration

In this section, we explain how a simple IoT scenario, a smart office, can be realized with the MBP. An office is defined as a room in which people conduct their work tasks. In order to make an office *smart*, it can be enhanced with

Fig. 3. Prototype: Lego miniature smart offices

computing power and sensing and acting capabilities through IoT devices, sensors, and actuators. A common practice in real-world scenarios is to integrate a heating, ventilation, air conditioning (HVAC) system into rooms. The goal of such a system is to keep the indoor environment comfortable for its occupants, while taking over tasks that can be executed automatically. For example, the room's temperature can be automatically regulated for its occupants based on user-defined goals and continuous sensor measurements.

In this demo, we enhance several Lego miniature offices with IoT devices, sensors, and actuators, in which many IoT applications, including a HVAC system, can be realized. Our Lego prototype is depicted in Fig. 3. To implement the HVAC system, we use two IoT devices: (i) a *Bosch XDK* device embedded with eight sensors, including humidity and temperature sensors, and (ii) a *Raspberry Pi*, which is connected to a relay actuator controlling a cooler fan.

A. Setting up the IoT environment

At first, the IoT devices are registered through the MBP Android-based smartphone application by scanning the QR code templates for the two different IoT device types, i.e., Bosch XDK and Raspberry Pi. The MBP smartphone application is depicted in Fig. 4.

Afterwards, the sensor and the actuator are registered and are each linked to the corresponding IoT device. Then, the sensor and actuator are bound to the MBP. For the sensor and actuator, we use extraction operators for temperature and humidity sensors of the XDK, and a control operator for the relay actuator. These operators are Python and Shell scripts and are provided in the MBP GitHub repository.

Finally, once the IoT objects are registered and bound to the MBP, the sensor values can be visualized in the detailed sensor view of the MBP. Moreover, several rules can be created for the set up IoT environment.

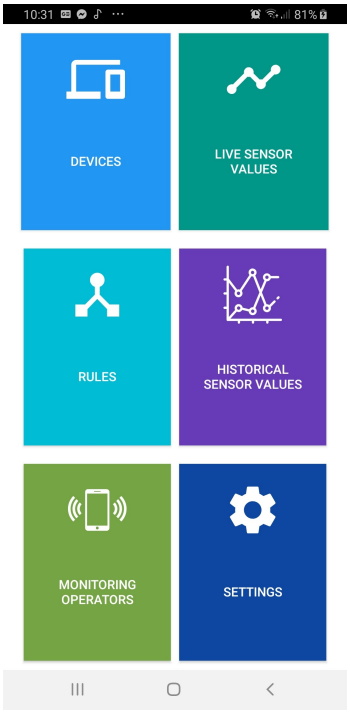

Fig. 4. The MBP smartphone application

B. Creating rules implementing the HVAC system

In the graphical rule modeling tool of the MBP, we create a rule in the form of an event-condition-action construction. The *event* part corresponds to the temperature values of the aforementioned sensor which form the input of the rule. The *condition* part is defined as "if the temperature values are under or above the thermal comfort zone inside the office, e.g., from 20 to 22 Celsius degrees". If the condition evaluates to true, the *action* part is executed which corresponds to sending control commands to the actuator controlling the cooler fan.

In this way, room temperature can be automatically regulated based on user-defined conditions and continuous sensor measurements.

REFERENCES

[1] O. Vermesan, and P. Friess, Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems. River Publishers, 2013.

[2] P. Hirmer, U. Breitenbücher, A. C. Franco da Silva, K. Képes, B. Mitschang, M. Wieland, "Automating the Provisioning and Configuration of Devices in the Internet of Things," in Complex Systems Informatics and Modeling Quarterly (CSIMQ), Vol. 9, pp. 28–43, 2016.

[3] IPVS – University of Stuttgart, Multi-purpose Binding and Provisioning Platform (MBP), https://github.com/IPVS-AS/MBP.

[4] D. C. Luckham, The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., 2001.

[5] A. C. Franco da Silva, P. Hirmer, R. Koch Peres, and B. Mitschang, "An Approach for CEP Query Shipping to Support Distributed IoT Environments," in Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops: IEEE, 2018.

[6] A. C. Franco da Silva, P. Hirmer, and B. Mitschang, "Model-based Operator Placement for Data Processing in IoT Environments," in Proceedings of the IEEE International Conference on Smart Computing (SMARTCOMP): IEEE, 2019.