EMBERS

HORIZON 2020

Information and Communication Technologies

Integrating experiments and facilities in FIRE+

Deliverable D1.2
Infrastructure Description

Grant Agreement number: 687992

Project acronym: EMBERS

Project title: Enabling a Mobility Back-End as a Robust Service

Type of action: Innovation Action (IA)

Project website address: www.embers-project.eu

Due date of deliverable: 2016-08-31

Dissemination Level		
PU	Public	X
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document properties

Leader partner	**Fraunhofer (FhG)**
Author(s)/editor(s)	Thomas Günther (FhG), Ricardo Vitorino & André Duarte (UW), Saint-Marcel Frédéric (INRIA)
Version	0.5

Abstract

There are currently many IoT middleware's that mainly addressing specific sectors and make it difficult for a customer to choose from the variety of available platforms and technologies. They are often implemented based on proprietary protocols or data information models. That causes vendor lock and hinders flexibility or integration of additional devices or technologies that might be required later on.

The EMBERS project addresses such problems, while providing an infrastructure with standardized open APIs. Supported standards will be OneM2M, ETSI M2M and OMA NGSI that make use of widely used application protocols like HTTP, CoAP and MQTT. This gives developers and IoT platform operators the freedom to choose their preferred technologies, while still interoperable with other standard compliant devices and applications.

This document contains a generic infrastructure description addressing the identified requirements for the platform. Furthermore, services and capabilities of the involved FIRE+ testbeds are specified as well as project relevant modifications. Additionally, deployments scenarios of the IoT backend and sensor nodes are discussed. It concludes with the events specific infrastructure requirements that have been identified to support a hackathon, app challenge and open call.

Table of Contents

1 Introduction

The Internet of Things (IoT) market is currently fragmented, with many vendors developing their own proprietary solutions. Particularly municipalities demand open, standardised interfaces which provide flexibility and also cost savings. With the purpose of making Smart Cities a reality, interoperable communication platforms need to support various devices and applications, hence EMBERS is providing a Mobility Back-end as a Service (MBaaS), which has been developed by Ubiwhere. The MBaaS is the central component of the EMBERS infrastructure that offers a domain-specific and open, standardised API to integrate devices that exchange information about traffic and mobility. These standardised interfaces guarantee a simple integration of existing devices and applications that already support such specification.

One of the objectives of EMBERS is to bring the MBaaS to the market. In order to do so, extensive testing is required, with the purpose of strengthening the platform to become market-ready. FIRE+ testbeds will be used to enable several tests to validate the platform's standard compliance and the performance. FIT IoT-Lab facility provides a significant number of wireless sensor nodes that can be configured to support multiple sensor topologies, while FUSECO Playground offers a standard compliant M2M platform that will emulate certain devices (e.g. parking and traffic sensors), which will prove the compliance regarding device registration. Both FIRE+ testbeds will be used for load testing the MBaaS, where the number of sensors, frequency of measurements and payload of sensor messages can be configured to identify the platform's limits.

The EMBERS infrastructure needs to be designed to support the planned load testing from multiple sites. Furthermore, both the MBaaS and the FIRE+ testbeds need to be accessible by third-party device or application developers. Such involvement will occur thanks to several promotion activities, as well as specific developer events, such as hackathons, app challenge and open call.

Another advantage of using standardised APIs is to make the MBaaS interoperable with other device vendors or application developers, instead of being bound to specific equipment, which gives high flexibility to the operators of such platform. However, there are many standardisation organizations and industry consortia that are developing standards to address domain or technology-specific challenges, making it difficult for developers to decide which standard to follow or which protocols to implement. The OneM2M [1] standard seems to be a promising candidate as its architecture addresses a horizontal approach on the service layer across verticals.

This deliverable introduces the available components of the EMBERS infrastructure consisting of the MBaaS, which is a central backend and the sensor nodes of the FIT IoT Lab and FUSECO Playground. Starting with a high level architecture that describes the interconnection of the infrastructures as well as the integration of device or application developers. Section 2 one can find the available devices and tools of the FIRE+ testbeds. Furthermore, it is outline what are the expected role of this components are and how they will connect to the MBaaS. Which deployment scenarios are supported by the sensor nodes or the mobility backend are discussed in section 3. Finally, section 4 covers the specifics that needs to be considered by the EMBERS infrastructure regarding the access of third parties during the planned dissemination events.

1.1 Generic EMBERS Infrastructure

The central component of the EMBERS infrastructure is the Mobility Backend, which is developed and operated by Ubiwhere. An essential part of this backend is the device broker since it takes care of the devices management, which can be either sensors or gateways that aggregate information of one or multiple constraint devices. If devices have enough computational resources and support application protocols, they can communicate directly with the broker, avoiding the need for a gateway. Applications can retrieve sensor data from the backend and provide innovative services targeting to solve mobility or transport challenges of smart cities.

The generic infrastructure addresses two scenarios. The first is where real applications or devices connect to the Mobility Backend to provide services for people or customers, which can assert the market-readiness of the solution and its operations. However, the Mobility platform needs to be tested from a performance and interoperability perspective. Secondly to evaluate the maximum performance of the platform regarding connected devices, frequency and payload the devices are generating simulation is required. The FIRE+ testbeds are used to assess the performance and interoperability of the Mobility Backend. Afterwards, through the organisation of several events and with third-party developers (from outside the consortium) using the platform and giving valuable feedback, it will be possible to strengthen and further enhance the platform.

All external interfaces of the mobility backend that are supposed to communicate with devices or applications need to be publicly reachable. Therefore, no specific requirements are expected to interconnect the FIRE+ testbed or any other developer's application or device with the MBaaS. Nonetheless, access to the open APIs of the MBaaS will be authorised. Furthermore, it is under investigation how the communication channel (transportation of the data) can be secured.

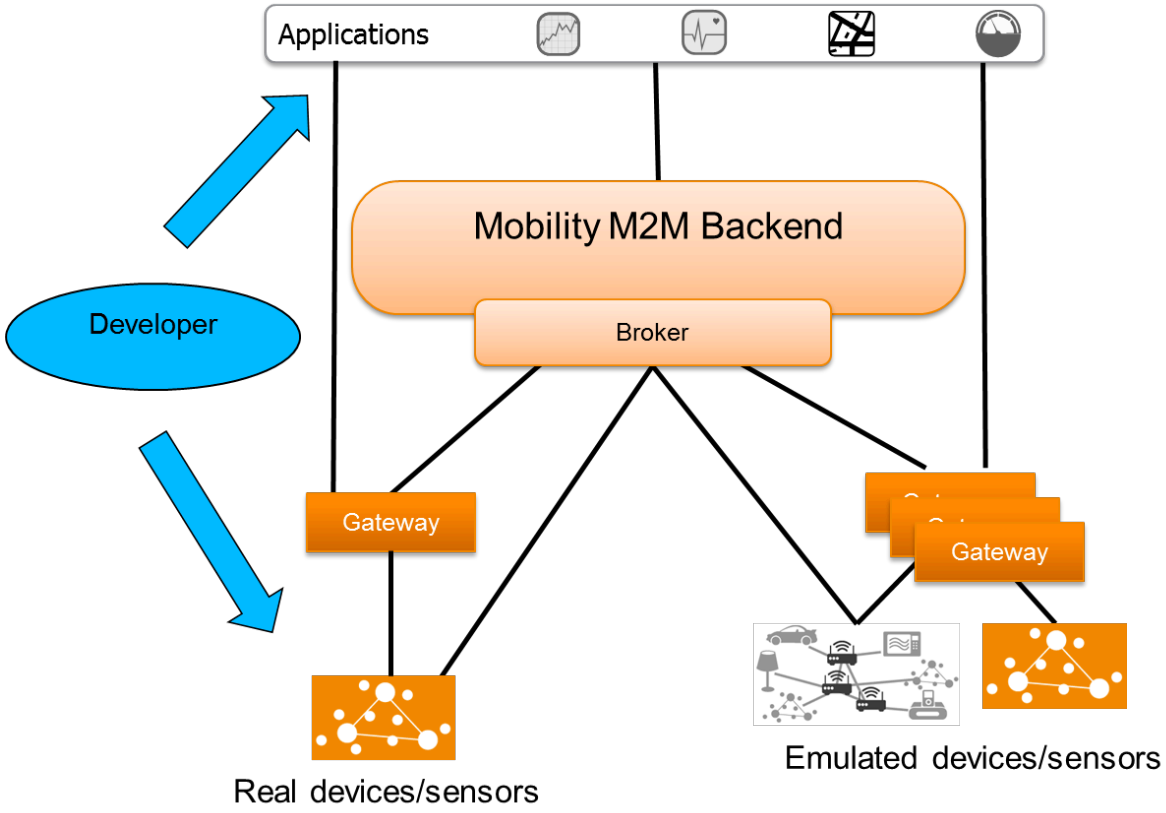

Figure 1 - Generic EMBERS infrastructure

2 Preparation and Enhancement of FIRE+ testbeds

This section reports about the available physical components and tools of the FIRE+ testbeds, which are namely the FIT IoT-Lab from INRIA and the FUSECO Playground from Fraunhofer FOKUS. Additionally it outlines the enhancements of the testbeds that where necessary to fulfil the requirements that have been analysed in T1.1.

2.1 FIT IoT-LAB

FIT IoT-LAB [2] offers a very large scale IoT testbed and offers to IoT designers, engineers, researchers an accurate open access multi-user scientific tool to support design, benchmarking, experimentation and accelerate the development of protocols, applications, IoT services and IoT technologies ranging from low-level protocols to advanced services integrated with Internet/Cloud. IoT-LAB testbeds are dispersed among six different locations across France (Grenoble, Lille, Paris, Rennes, Paris, Strasbourg, Saclay) offering access, for the first time, to 2728 wireless IoT fixed and mobile nodes are equipped with various sensors: Ambient sensor light, Temperature, Atmospheric pressure and temperature, Tri-axis accelerometer, Tri-axis magnetometer, Tri-axis gyrometer.

Main services offered include:

- Total remote access to nodes reserved, e.g., allowing users to flash any firmware, without any condition or constraint. Any language or OS can be used to design, build, and compile applications;
- End-to-End IP connectivity. Each node could be visible from Internet with end-to-end IP connection using IPv6 and 6LoWPAN [3] for example;
- Strong support with a set of useful detailed tutorials, OS supports (Contiki [4], FreeRTOS [5], and RIOT [6]) including full protocol stacks and communication libraries such as OpenWSN [7] providing open-source implementations of IoT protocol standards;
- Access to the serial ports of all nodes for a real-time interaction, with optional aggregation; The user can collect and aggregate if he/she wants all serials ports of all reserved nodes;
- A unique fleet of mobile robots (eg. TurtleBot);

To offer a development environment which is representative of today's range of IoT hardware, two types of IoT-LAB Nodes are deployed, differing by the capabilities of their ON:

The M3 ON features a 32-bit ARM CortexM3 micro-controller (STM32F103REY), an AT86RF231 IEEE802.15.4 [8] radio chip, and sensors. The user can interact with the M3 ON over its serial port and also debug it using the JTAG protocol. The M3 ON is representative of today's state-of-the-art IoT devices.

Figure 2: IoT-LAB M3 Open Node

The A8 ON is the most powerful node in the IoT-LAB. It features a ARM Cortex-A8 600 MHz 32-bit mini-computer, a ARM Cortex-M3 micro-32-bit controller, an AT86RF231 IEEE802.15.4 radio chip and sensors. The A8 ON is representative of more advanced devices such as set-top gateway and may run Linux.

Figure3: IoT-LAB A8 Open Node

Two classes of software run on the IoT-LAB. The first includes the source code, libraries and tools available for a user and IoT-LAB nodes programmation. The second is the "back-end" software which glues together all the pieces of the IoT-LAB. Most of the back-end software runs on the servers in the IoT-LAB backbone network. It is in charge of user management, nodes allocation and experiment scheduling. The main objective of the back-end software is to allow a user to interact with all IoT-LAB nodes in a reliable and real-time fashion.

A user starts by creating a user account on the IoT-LAB's main website or using his/her OneLab federation account. He/she then communicates with IoT-LAB through its web portal or REST API that are available through command-lines tools (eg. CLI tools) to offer a simpler user experience. To launch an experiment, the user reserves one or more nodes, in one or more sites, for some duration. To each node are attached some features: mobile/fixed, location and radio chip. After reserving the nodes, the user can make additional configurations for the experiments, including specify the firmware image(s) to load on the nodes. The scheduler starts the experiment as soon as the nodes are available, or at a precise date, if specified. Once an experiment is running, the user has full control over the nodes reserved and dedicated to him/her. Nodes can be reseted, reconfigured, or reprogramed individually or globally.

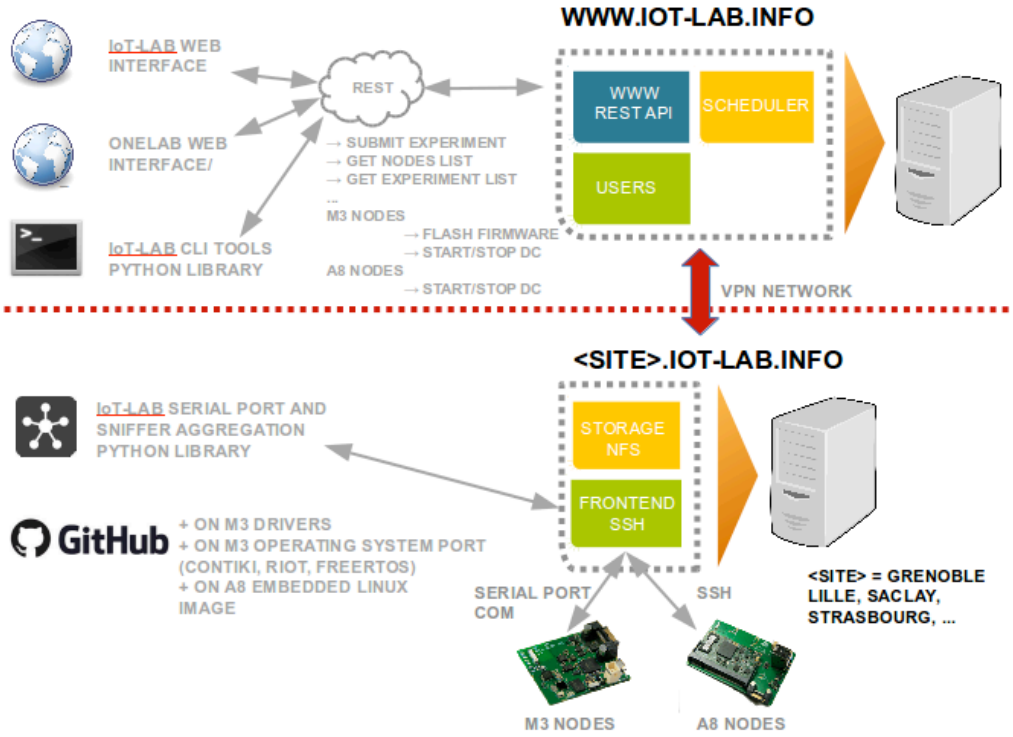

Figure4: IoT-LAB testbed infrastructure

As part of the project EMBERS and the MBaaS experimentation support, we provide different software components.

In the first place we developed MBaas application firmwares executed on the IoT-LAB M3 Open Node during the experiment in charge of reading sensors values. We have defined that we send to the MBaaS broker device three hardware sensors measurement (eg. temperature, pressure and luminosity) and one emulate sensor measurement with parking event. This emulation is based on Poisson discrete probability distribution and could be improve with others distribution model during the project. Moreover, these firmwares are configurable by user and you can specify the period of sensor measure. It is also expected that we will test different firmwares implementation with different IoT embedded Operating Systems (eg. RIOT, Contiki, FreeRTOS) and wireless radio standard protocols like IEEE 802.15.4e [9] (eg. 6TISCH)

Next we provide experimentation tools (Python scripts) for automate and benchmark the MBaaS device broker interaction. This part of software must be installed on a gateway hosting in IoT-LAB testbed (eg. SSH frontend server or A8 node). These scripts use IoT-LAB python libraries for executing experiment control plane (eg. submit and launch an experiment, choose and book nodes, flash firmwares) and aggregate all experiment nodes measurement data. Moreover we have developed device brokers client implementation for register nodes and sending measurement data to the MBaaS middleware with different protocols (eg. HTTP, CoAP and MQTT). The first implementation and tests are executed on an open-source device broker Meshblu instance provide by Ubiwhere partner with HTTP protocol communication.

2.2 FUSECO Playground

FUSECO Playground [10] operated by Fraunhofer FOKUS unifies all the research activities and toolkits in one testbed to do proof of concept evaluation or to run experiments. One of the toolkits that is highly relevant for the EMBERS project is OpenMTC [11]. This toolkit is a reference implementation of the OneM2M standard and supports multiple application protocols. One of the components is the OpenMTC gateway which enables heterogeneous sensors (e.g. Zigbee or FS20), through the help of

interworking proxies, to translate the technology specific communication into a standardized protocol which is understood by the M2M/IoT backend. Thus the OpenMTC gateway will be used to conduct the interoperability tests.

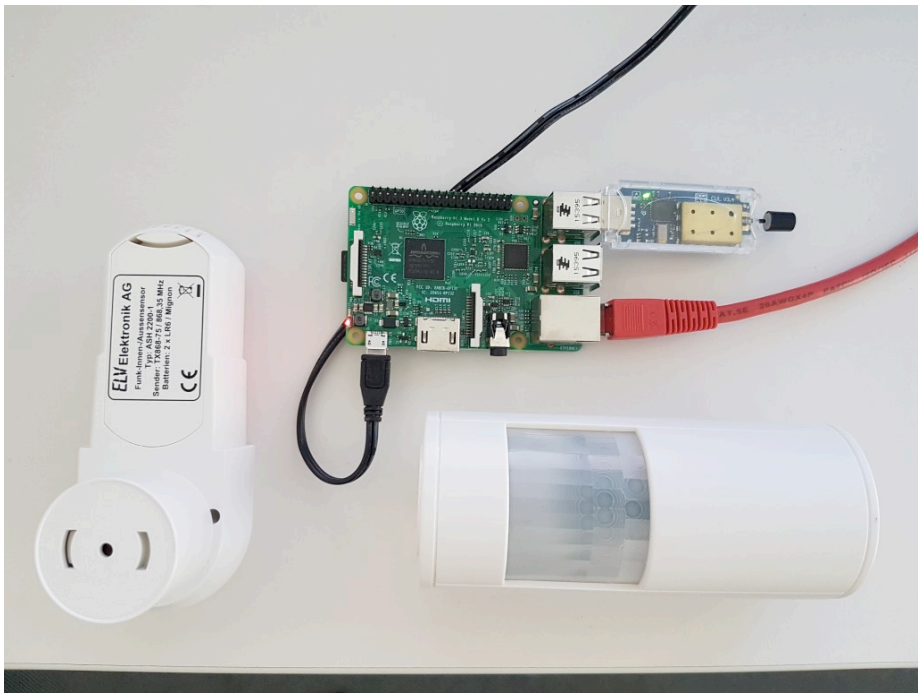

Figure 5: OpenMTC gateway on Raspberry Pi with attached sensors

OpenMTC is a lightweight software that can be deployed on any device supporting a Linux operating system. Typically the OpenMTC gateway is deployed on customer of the shelf (COTS) hardware, like a Raspberry Pi. The latest Raspberry Pi 3 has integrated LAN, Wi-Fi and Bluetooth LE which provides multiple communication interfaces that can be used to connect to the backend. Other than that the single-board computer can be easily extended with modules or USB sticks to support additional RF technologies (LTE, Zigbee etc.). FUSECO Playground supports multiple radio access technologies that could be used for experimentation.

As the FUSECO Playground is designed to support PoC rather than large scalability test in terms of available sensor devices, it was decided to implement a load generator. Thus the FUSECO Playground will be extended to support dozens of emulated sensor devices. Emulation of devices is needed as it allows to perform benchmarking tests, without the requirement to have them physically available. The following requirements have been identified. Provide a central and simple interface to the user for the emulation of devices and sensors or even applications that is remotely accessible. Users should be able to adjust parameters for their needs by using a configuration file. Design the software as generic as possible, to support testing of multiple device broker and protocols. As already explained in D1.1 [12], the testbeds are supposed to benchmark multiple device broker, which are responsible for the device management of the MBaaS. This includes device registration and authorization, while each device could contain multiple sensor data. After the execution of the experiment the results (error rate) should be provided to the user. Figure 3 depicts a high level view of integration of the load generator into the EMBERS architecture.

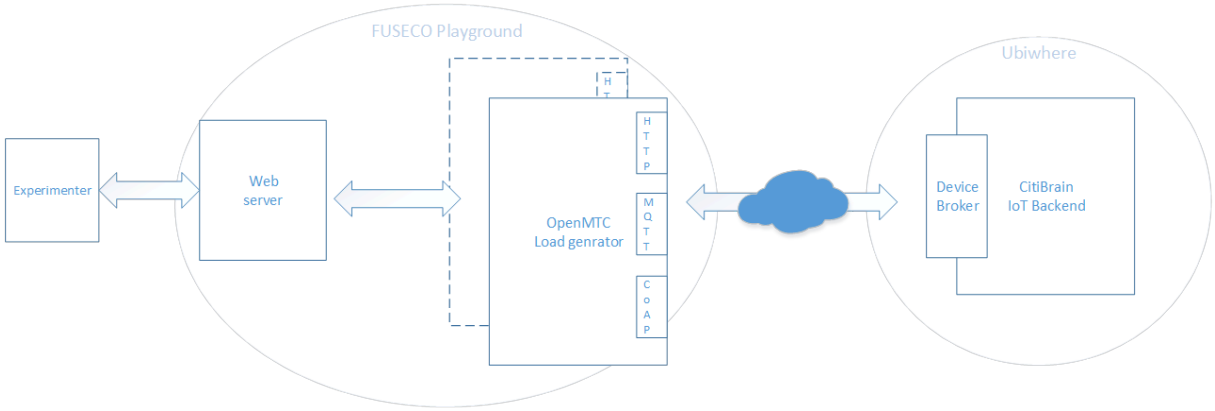

Figure 6: High level architecture of load generator scenario

The picture above shows a typical workflow of an experimenter using the FUSECO load generator. An experiment is started by sending a configuration file in JSON syntax via RESTful API to the web server. This file will be forwarded to the load generator. The load generator will parse the given configuration and setup a specific amount of virtual sensors. After that all virtual sensors will be registered with the configured device broker (e.g. MeshBlu). Additionally, the load generator will at least subscribe itself to every virtual sensor once in order to measure latency. If all virtual sensors are registered successfully, the experiment is started with each virtual sensor sending data in the configured fashion. When the experiment duration is exceeded, the experimenter is able to acquire the experiment log (most notably containing the measured latencies) from the web server via a RESTful API.

Foreseen configuration parameters are the amount of devices, as well as the frequency of information send and the size of the payload. To make the performance test realistic as possible it is planned to integrate historical data sets (e.g. parking events) of municipalities are used as input source for the load generator. The load generator will be implemented to support multiple application protocols (e.g. HTTP, CoAP, MQTT) and service endpoint that can be defined in a configuration file.

It turned out through internal discussions that the benchmarking will be the first step. Therefore the load generator developed by Fraunhofer FOKUS will be placed at the Ubiwhere facilities. This will provide the most accurate results as there are no Internet hops in between that cause additional latency. For experimenters that are interested in the emulation of sensor data or devices, that are not currently offered by EMBERS, a dedicated instance of the load generator will be provided at the FUSECO Playground to support more customized scenarios.

Besides the emulation of sensor data, it is additionally possible to get access to the real sensor data that are attached to OpenMTC gateways distributed across the FOKUS building. Measurement of temperature, humidity, air pressure and brightness are supported by the available Zigbee sensors. The workflow of configuring the OpenMTC gateway is similar to the load generator use case. The experimenter needs to connect to a web server where he needs to select the real gateway that will be used for experimentation. Furthermore, the experimenter can configure through a UI of the web server the interval for querying the sensor data. As soon the configuration is completed the web server triggers the process on the OpenMTC gateway to start the device broker specific application which includes the registration procedure of the device including the sensors. Afterwards the OpenMTC gateway will send updates of the sensor data depending on the previous configured interval.

3 Deployment Scenarios

EMBERS infrastructure components are supporting multiple deployment scenarios. Several topologies can be deployed in the backend as well on the sensor node level. Typically one or more sensors are connected to a gateway, which aggregates the data a forward the information to the backend. The architecture strongly depends on the capabilities of the sensor devices. While IoT-LAB nodes have the sensors integrated on board and supporting operating systems with full IP stack, FS20 sensors from FUSECO Playground are more constraint because they send sensor information via a RF signal, that needs to be received by a gateway which parses the signal and forwards it to the backend. That is the reason while currently FUSECO Playground topology capabilities are limited to Point to Point scenarios. However more complex scenarios are supported by the IoT-LAB which are described in section 3.1.2. Other than that IoT backends need to adopt to certain conditions e.g. increasing load or density of sensors inside the infrastructure. That is why section 3.2 discuss multiple deployment scenarios from the backend perspective and what are potential consequences.

3.1 M2M gateway/sensor deployments

3.1.1 Point to Point

The first deployment scenario is based on “Point to Point” network between an IoT-LAB gateway and experiment nodes. For this purpose, we use the serial port nodes communication (eg. tcp socket) accessible by the user during the experiment from IoT-LAB SSH frontend site which acting the role of gateway. Thanks to experimentation tools, we provide IoT-LAB python libraries for aggregate all the experiment nodes serial port communication on the gateway. With this scenario we choose and book the number of M3 nodes required and flash a firmware on them.

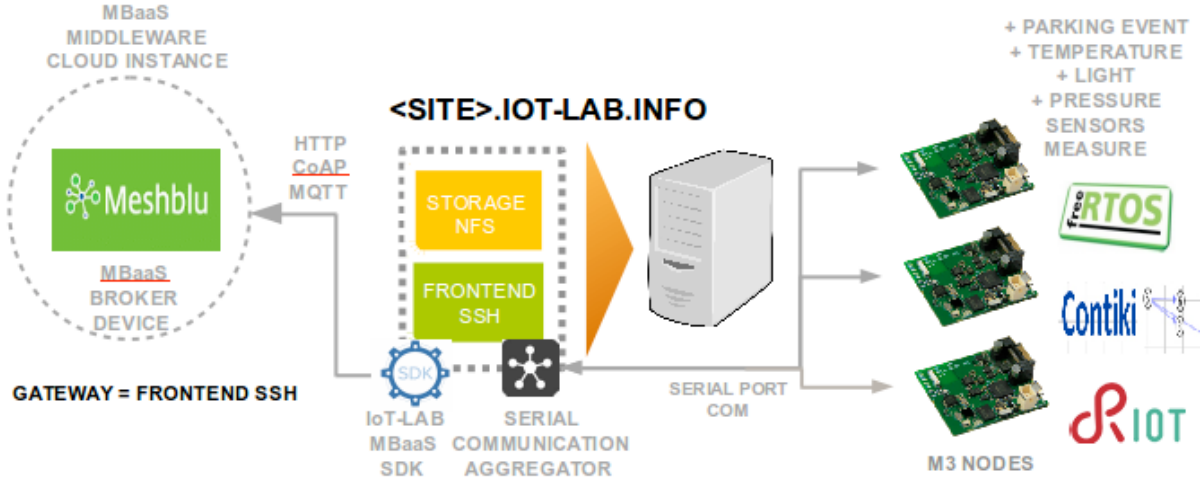

Figure 7: Point to Point deployment scenario

This firmware waits a measurement configuration (eg. which sensors and measure period) send from the gateway by the serial port and start measurement process. It reads sensors values with the period configuration and write the results on its serial port. The gateway collects the measurement data and send it to the MBaaS device broker.

3.1.2 Meshed

In this second deployment scenario we used 6LoWPAN mesh network (IPv6 over Low power Wireless Personal Area Networks) between an IoT-LAB gateway and experiment nodes. 6LoWPAN is an open standard defined in RFC 6282 by the Internet Engineering Task Force (IETF). A powerful feature of 6LoWPAN is that while originally conceived to support IEEE 802.15.4 low-power wireless networks in the 2.4-GHz band like IoT-LAB M3 nodes.

The uplink to the standard Internet (eg. MBaaS cloud instance) is handled by the Access Point (AP) acting as an IPv6 router (eg. IoT-LAB gateway). It could be done on the IoT-LAB testbed with SSH frontend site or with an A8 node. The 6LoWPAN network, with experiment nodes (eg. IoT-LAB M3 nodes) are connected to the AP using an edge router (eg. one M3 experiment node) and its serial port communication. For example, in the case of ContikiOS embedded operating system we use tunslip6 tool which encapsulate IPv6 traffic between AP and the edge router over serial line communication. Moreover, we use RPL routing protocol defined by IETF in RFC 6550 for route-over the 6LoWPAN network. It is also expected in this project that we will use 6TiSCH implementation ("IPv6 over the TSCH mode of IEEE 802.15.4e") for testing the reliability of 6LoWPAN network.

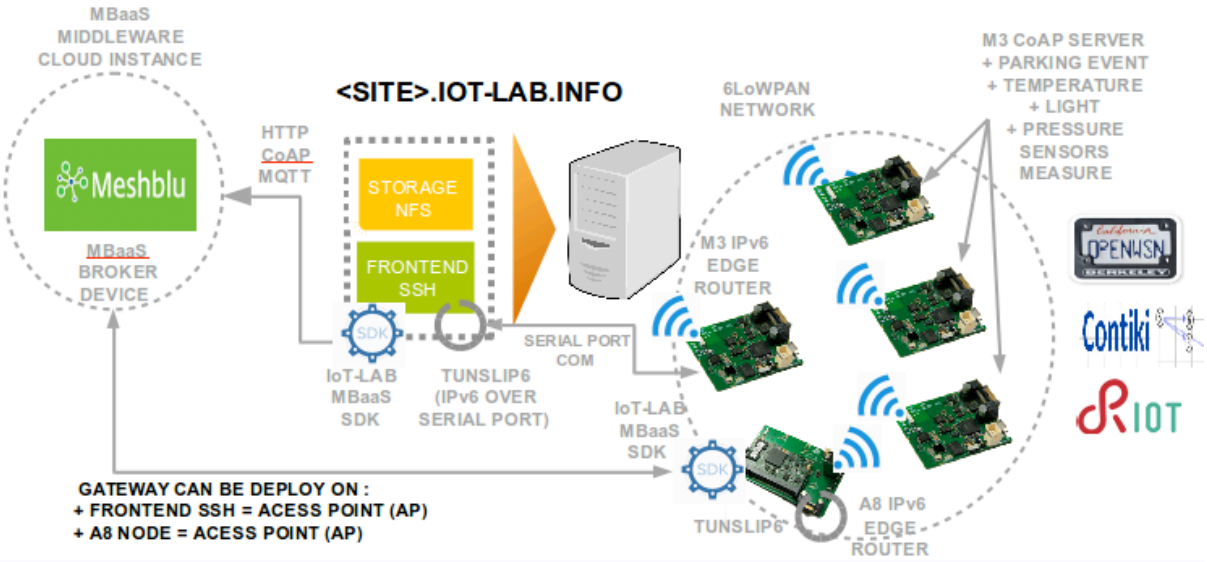

Figure 8: Meshed deployment scenario

Finally, as IoT-LAB M3 nodes demanding ultra-low power consumption we use UDP, lower overhead and connectionless protocol, and the constrained application protocol (CoAP). Indeed, we wrote MBaaS application firmwares with CoAP server from which you can send a measurement sensors configuration and get the measurement data with different CoAP resources.

On the IoT-LAB gateway the experimentation tools send the measurement configuration and aggregate all the measurement data with CoAP clients and send it to the MBaaS device broker in the same way of “Point to Point” deployment scenario.

3.2 Server-side deployments

The MBaaS can be deployed on any cloud or physical infrastructure. Therefore, the process of deploying new instances of the MBaaS can be done at fast pace, since it takes advantage of a microservices architecture based on Docker. The performance available on the system depends on the resources available in the machine.

The MBaaS, fully separated as a product from the devices and apps, yet interacting with them via well-defined RESTful APIs. For optimal performance during the lifecycle of the MBaaS, the deployment

process should be configured by Ubiwhere's IT team. At this point, we can define, as main deployment scenarios the following three:

- Single Machine
- Multiple Machine

Detailed descriptions of these scenarios are available in the following subchapters.

3.2.1 Single Machine

On the single machine scenario, the MBaaS is fully deployed on a given server, where the typical design consists of a device broker, an event broker, an authorization and authentication server and multiple other services that allow the functionalities of the backend. This architecture is fully modular and uses Docker containers for deployment purposes. Below, it is possible to find a picture describing the typical deployment scenario for a single machine, where, for the purpose of this example it is assumed that the machine contains 2 CPUs, 4GB of RAM and 100GB of storage.

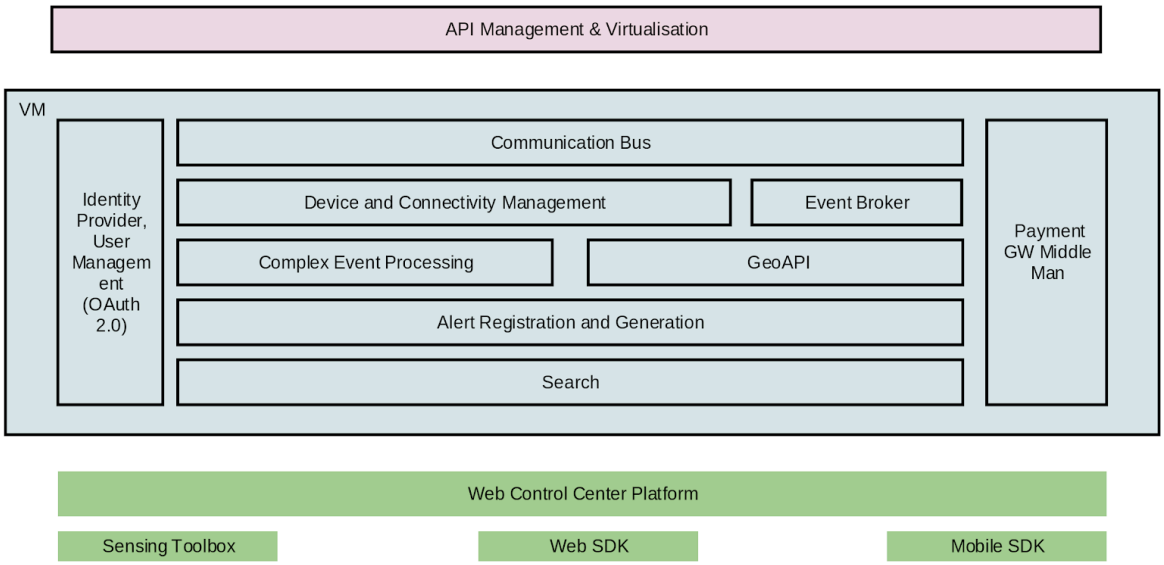

Figure 9: Single machine sample logical architecture

In this scenario, every logical module of the MBaaS architecture belongs to a single instance of a virtual machine, which despite simplifies the deployment and maintenance processes, turns the process of scaling the platform into a difficult task, due to the physical limitations of the Virtual Machine.

3.2.2 Multiple machines

On a multiple machines scenario, the MBaaS architecture will be separated throughout the available servers. The components of the MBaaS communicate with each other via AMQP or REST APIs. This allows for seamless communication from every component that is deployed regardless of its machine or location. Depending on the resources available for each machine, one or more platform components can be deployed there. For such type of deployment there are two possible scenarios:

- Every server has the same available resources, which means that the deployment order of the MBaaS components is only taken into account when there is the need for critical components to allocate more resources (thus separated from the less greedy components), where the lower resource-consumers can be clustered on a machine. The figure below helps to understand how the deployment architecture should look like and to clarify this particular

scenario. With the purpose of simplifying, one assumes that every machine has a single CPU, with 1GB of RAM and 30GB of storage.

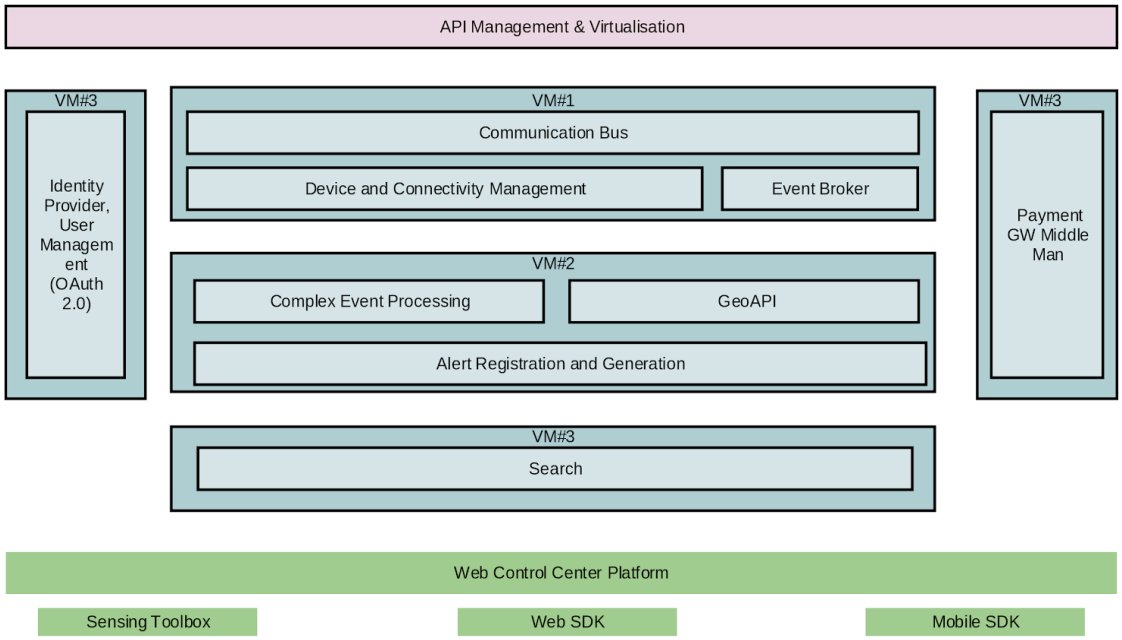

Figure 10 – Multiple machine deployment scenario (Machines have the same resources)

- Each server differs in the resources provided, meaning that the scenario requires particular attention to the deployment process needs regarding the resources of each machine to enable optimal performance at every step of the MBaaS flow. To clarify, the figure below shows an example of this with sample resource values, assuming that, for the purpose of this example, there were only available the following four machine specifications:
    - 1 CPU, 1GB of RAM, 30GB of storage
    - 1 CPU, 2GB of RAM, 30GB of storage
    - 1 CPU, 1GB of RAM, 30GB of storage
    - 1 CPU, 4GB of RAM, 100GB of storage

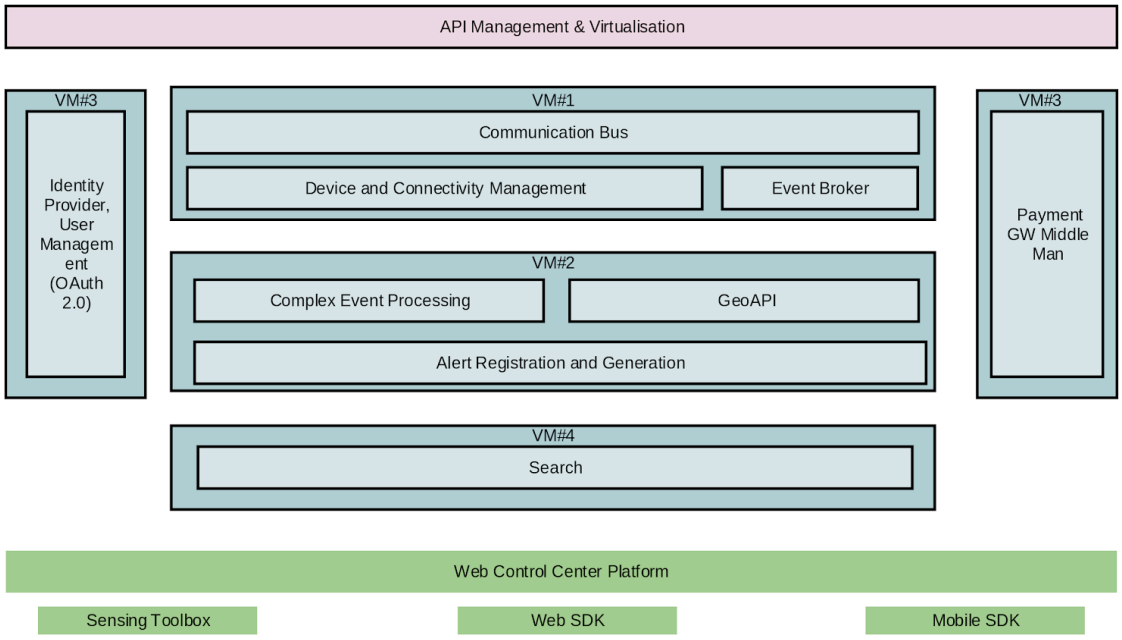

Figure 11: Multiple machine deployment scenario (Machines differ on resources)

With this in mind, there can be a sample separation of the backend throughout these machines:

- VM#1 - Since this machine is responsible for holding the communication bus, device management and the event broker, with this approach the requirements consist of 1 CPU, 1GB of RAM and 30GB of storage. Considering the available machines, these specifications are enough to deploy these components.
- VM#2 - This machine holds the logic to the upper-tier APIs and CEP engine; therefore, the chosen specifications are 1 CPU, 2GB of RAM and 30GB of storage to allow for better performance due to the amount of computations performed by these components.
- VM#3 - The authorization server and the payments components usually do not need much computational power, hence the 1 CPU, 1GB of RAM, 30GB of storage is enough.
- VM#4 - The last, and most powerful machine, with 1 CPU, 4GB of RAM, 100GB of storage, is responsible for the search/logging system of the platform since it is imperative to allow for better and quicker debugging when needed, by the means of an efficient centralised logging module.

4 Event Infrastructure Set-Up

At each event, an instance of the MBaaS will be available. This instance will be configured by Ubiwhere and will be used at every event. Using the same instance enables the optimisation of the preparation time for each of the events and to have more data available during the events. Every deployed instance will have datasets available from cities that have provided them, therefore the MBaaS will be deployed with data so that every user can experiment. Moreover, data added during the events is used in the subsequent ones to improve the overall quality of the datasets, the quantity and diversity of the available data. Moreover, it is important to clarify that user information will not be shared throughout the events leaving each user with a registration procedure for each of the events.

The registration procedure can be completed by using a web app developed with this objective. After registration, the users can access the same web app to register their devices, regardless of their protocol or testbed. This we app also serves to register user's own devices, if that is the case.

Lastly, to improve the ease of use of the MBaaS and understand its APIs documentation is available during every event. This documentation will be hosted at APIary (http://docs.emberscity.apiary.io/#), which allows the user to look at the API specification and try the API calls to a mock server which returns sample response data. The mock server will help bring ideas to life by interacting with a real API from an early stage.

5 Conclusions

One of the main objectives of the EMBERS project is to make the MBaaS platform from Ubiwhere market ready. This should be achieved through benchmarking and interoperability testing, where the FIRE+ testbeds (FIT IoT-LAB, FUSECO Playground) will be utilized. The results of the stress tests will help Ubiwhere to improve the reliability and scalability of the MBaaS. Furthermore, third-party developers should use the EMBERS infrastructure, over the period of the project organized events. This will help Ubiwhere to get valuable feedback from developers outside the project consortium if the platform fulfils the requirements of the market. In this deliverable it is described what are the functional elements of the EMBERS infrastructure and which role they are going to play to reach the objective. There are a lot of IoT middlewares available out in the market. However the market acceptance strongly depends on the usability and interoperability. While supporting multiple IoT standards and involving the mobility and transport community EMBERS is placing the corner stone for a successful outcome of the project.

6 References

[1] oneM2M, „oneM2M,“ [Online]. Available: http://www.onem2m.org/.

[2] INRIA, FIT IoT-LAB, [Online]. Available: https://www.iot-lab.info.

[3] 6LoWPAN, [Online]. Available https://tools.ietf.org/html/rfc6282

[4] Contiki, [Online]. Available http://www.contiki-os.org/

[5] FreeRTOS, [Online]. Available http://www.freertos.org/

[6] RIOT, [Online]. Available https://riot-os.org/

[7] OpenWSN, [Online]. Available https://openwsn.atlassian.net/

[8] IEEE 802.15.4, [Online]. Available http://www.ieee802.org/15/pub/TG4.html

[9] IEEE 802.15.4e, [Online]. Available http://www.ieee802.org/15/pub/TG4e.html

[10] FRAUNHOFER FOKUS, „FUSECO Playground,“ [Online]. Available: https://www.fokus.fraunhofer.de/go/en/fokus_testbeds/fuseco_playground.

[11] FRAUNHOFER FOKUS, „openMTC,“ [Online]. Available: http://www.openmtc.org/.

[12] R. V. (UW), "http://embers.city/," 2016. [Online]. Available: http://embers.city/wp-content/uploads/2016/05/EMBERS_Deliverable_D1_1_v0_2.pdf.