Reactive Microservices for the Internet of Things: A case study in Fog Computing

Cleber Jorge Lira de Santana
Federal Institute of Bahia
Federal University of Bahia
Salvador, Bahia, Brazil
cleberlira@ifba.edu.br

Brenno de Mello Alencar
Federal University of Bahia
Salvador, Bahia, Brazil
brenno.mello@ufba.br

Cássio V. Serafim Prazeres
Federal University of Bahia
Salvador, Bahia, Brazil
prazeres@ufba.br

ABSTRACT

The Future Internet will be able to connect most of the objects that are not yet connected on the current Internet. The Internet of Things (IoT) is an important part of the Future Internet and involves connectivity between several physical and virtual objects, allowing the emergence of new services and applications. These intelligent objects, along with their tasks, constitute domain-specific applications (vertical markets), while ubiquitous and analytic services form independent domain services (horizontal markets). The development of these applications and services in these markets brings challenges such as deployment, scalability, integration, interoperability, mobility and performance. Recent research indicates that Microservices has been successfully applied by companies such as Netflix and SoundCloud to address some of these issues in their cloud computing applications. However, in the field of IoT, the use of Microservices to deal with these challenges still presents unresolved issues. In this paper, we present a reactive Microservices architecture and apply it in a Fog Computing case study to investigate these challenges at the edge of the network. Finally, we evaluate our proposal from the perspective of performance of Microservices provided by intelligent objects (IoT gateways) at the edge of the network.

CCS CONCEPTS

• **Software and its engineering** → **Software development techniques**; • **Computer systems organization** → *Redundancy*; • **Software development methods** → Design patterns;

KEYWORDS

Reactive Microservices, Architecture, IoT Platform, Internet of Things, Fog Computing

ACM Reference Format:
Cleber Jorge Lira de Santana, Brenno de Mello Alencar, and Cássio V. Serafim Prazeres. 2019. Reactive Microservices for the Internet of Things: A case study in Fog Computing. In *The 34th ACM/SIGAPP Symposium on Applied Computing (SAC '19), April 8–12, 2019, Limassol, Cyprus.* ACM, New York, NY, USA, Article 4, 9 pages. https://doi.org/10.1145/3297280.3297402

SAC '19, April 8–12, 2019, Limassol, Cyprus

ACM ISBN 978-1-4503-5933-7/19/04…$15.00
https://doi.org/10.1145/3297280.3297402

1 INTRODUCTION

The IoT enables common objects of everyday life to see, hear, think, perform tasks, share information, and coordinate decisions by transforming these objects into smart devices [1]. Gartner [42] expects about 21 billion devices by 2020. These new devices will be able to interact by providing new services in different domains and environments, generating added value for the user. According to Al-Fuqaha et al. [1], and Shahid and Aneja [38], there are diverse domains that constitute vertical markets (e.g. smart cities, smart houses, healthcare, industrial automation) and horizontal markets (e.g. ubiquitous computing and analytical services), which form domain-independent services. As a consequence, IoT can improve people's quality of life by using the technologies of the horizontal markets on the vertical markets domains.

Some research projects discuss some problems and point solutions related to the enabling of technologies and protocols to make vertical and horizontal markets a reality. For instance, Khan et al. [22] present some of these discussions, where the authors address the challenges associated with developing and deploying applications over the IoT; Khoi et al. [40] discuss the challenge of allowing the interaction between smart objects and mobile users under an intermittent Internet; Jarvinen et al. [21] discuss the challenges related to providing support for the heterogeneity of IoT devices and communication protocols; and Reinfurt et al. [35] present some patterns to deal with recurring problems in IoT.

However, building large-scale systems and platforms is challenging [25], once it is necessary to design applications that are more robust, resilient, flexible and better adapted to support current demands. Some works [27][2] discuss the challenges of developing and deploying IoT applications. The dynamically changing IoT environment requires these systems to scale and evolve over time by adopting new technologies and requirements. Krylovskiy et al. [25] point out that technology and the ever-changing market require new approaches to designing architectures that are highly available and scalable on demand. On-devices IoT applications need to ensure that data and messages have been received [28]. For instance, healthcare applications monitor the patient's health status and allow healthcare professionals to access real-time information, which helps them make more accurate decisions and therefore provide better results.

From these considerations, we claim that monolithic applications may not be designed to deal with these issues, due to the dynamic communication characteristics of the devices, to the need for elasticity of applications and to the decentralized approach provided by different stakeholders [8].

In order to overcome some of these challenges, service-oriented architecture (SOA) has been adopted in the last few years in the project of IoT applications and in cyberphysical systems (CPS) as discussed by Shancang et al. [29], and Kumar el at. [26]. However, it can be noted that this is not enough to achieve interoperability among several solution providers as pointed out by Butzin et al. [8], and it becomes more difficult to scale an application because services cannot be deployed independently and in face of this perspective the SOA can still be seen as a monolithic approach [45].

In this context, the design of an architecture for IoT applications needs to consider modularity, resilience, scalability, extensibility and interoperability among heterogeneous devices. Since objects may move or interact in real-time with their environment, an adaptive architecture is needed to help different devices interact dynamically with other objects. In addition, IoT applications are deployed from mobile devices and gateways to clusters in the cloud that generate data in Zettabytes. This leads to the need to design these applications from new concepts. Therefore, in response to the challenges for the construction of applications and distributed platforms that are resilient, elastic and responsive, two concepts can be used together: Microservices and Reactive Systems.

Microservices aim to build, manage and design architectures of small autonomous units [13]. Microservices has emerged in the software industry and has been successfully adopted by companies such as NetFlix, Amazon, Soundcloud and Gilt in order to meet the maintenance and scalability demands of their online applications.

Reactive Systems [46] are built according to the following characteristics: if possible, the system will be (i) **responsive** in a timely and (ii) **resilient** manner, and will continue to respond in case of failure; and (iii) **elastic**, the system will respond according to the variation of demand. To have these characteristics, the system must use asynchronous messages in order to guarantee low coupling, isolation and location transparency [7].

In this paper, we have also presented a model for reactive Microservices that can be used in the development of IoT applications, including in Fog Computing scenarios, where part of the data processing capacity and service delivery operations are processed locally on servers with lower processing capacity. In addition, we have extended the SOFT-IoT platform [34], adding the necessary components for the implementation of reactive Microservices and instantiating the proposed model in a case study in Fog Computing.

The remainder of this paper is organized as follows: Section 2 discusses the current research in SOA and Microservices for IoT. Section 3 presents the model for the construction of reactive Microservices proposed in this paper. In Section 4 we present a platform, which is a proposal for the implementation of reactive Microservices. In Section 5, we discuss and present the results obtained when instantiating the proposed model. Finally, we present some concluding remarks and future works in Section 6.

2 SOA AND MICROSERVICES FOR IOT

2.1 SOA for IoT

In the last years researchers have presented solutions for the development of IoT applications from the perspective of the SOA architectural style. Eliasson et al. [12], López and García [4], Zhang et al. [47] are part of the research that has been applied in different projects, aiming to provide scalability, portability and reusability.

However, IoT deals with new requirements and demands a different development approach in comparison with traditional SOA. In a more recent work, Issarny et al. [18] proposed a new approach to SOA for the development of IoT applications based primarily on a Service-Oriented Middleware and on eVolution Service Bus (VSB). VSB follows the Enterprise Service Bus (ESB) paradigm and aims to interconnect things that employ heterogeneous interaction protocols at the middleware layer (e.g. CoAP, MQTT, etc.). However, even ESB can solve some issues related with modularity, scalability, and extensibility, it can be a fault handler and therefore affect the operation of an entire application considering services communicating through ESB [31].

In addition, according to Sun et al. [39], Cerny et al. [11], and Newman [32], SOA applications present other limitations: (i) Deployment: it is difficult to selectively deploy services in SOA. (ii) Scalability: A change in a function may affect other functions due to the strong coupling among services and for that reason it becomes more complex to reimplement, maintain and perform the continuous integration of IoT applications. In addition, it causes bottlenecks in the integration unit and limits elasticity. (iii) Performance: At times, in order to improve system performance, multiple deployments are required, and overhead functions create system-wide bottlenecks. (iv) Management: Centralized management may lead to the unavailability of a system from cascading errors. (v) Interoperability: Finally, the entire system uses a single technology platform and development standards, which in turn limits methods to solve the problem of physical heterogeneity. Besides that, considering that IoT depends on the integration of a large number of technologies, it is reasonable to conclude that a monolithic approach is not an adequate solution.

2.2 Microservices for IoT

To address these limitations when developing IoT applications, researchers (Table 1) are recently adopting Microservices. According to Fowler and Lewis [13], Microservices is a model for implementing an application as a set of small services, each running in its own process and communicating with lightweight mechanisms, often as an HTTP resource API. In contrast to traditional architectures for designing applications in IoT, the Microservices architecture offers advantages such as those discussed by Newman [32]: i) Technology Heterogeneity; ii) Resilience; iii) Scalability; iv) Ease of Deployment; v) Organizational Alignment; vi) Composability; vii) and Optimizing for Replace ability, which will enable the development of large-scale IoT applications.

In a recent study, Di Francesco, Malavolta and Lago [14] stated that the Microservices was applied in domains such as the IoT and Mobile. In addition, billions of IoT devices produce data that is measured in Zettabytes, with the result that the architecture of IoT applications is then designed with reactive features [7][32]. For this reason, we have performed a Mapping Study [37] to identify studies on the adoption of Microservices published by the scientific community, to address the inherent challenges of the IoT architecture application layer. From this, we extract and propose an architectural

model for the construction of Microservices in the IoT as presented in Section 3.

The academic interest in Microservices for IoT application development is recent [10][37], as can be viewed through the year of publication of several works presented in Table 1. Considering the phases of the software lifecycle, the works presented in Table 1 show that when architecting Microservices for IoT, Design is the predominant phase (see Table 2 - Solution Proposal). With gradual maturity, there is a need fo works that investigate complexities in the other phases (i.e, implementation, maintenance, evaluation and operation). In this direction, our work presents the following contributions:

- Introduction to Microservices and their use in the development of IoT applications (results presented in Tables 1 and 2);
- Design of a model for the development of reactive Microservices in IoT that can be used in Cloud Computing and Fog Computing solutions (design phase);
- Design and Implementation of a Gateway (Figure 4) for Reactive Microservices IoT Platform and Reactive Microservices Infrastructure (design and implementation phase);
- Implementation of the reactive Microservices model in a Fog Computing platform (implementation phase);
- Performance evaluation of Reactive Microservices Infrastructure at the edge of the network in a Smart Water scenario (evaluation phase).

3 REACTIVE MICROSERVICES FOR IOT

Microservices are based on the UNIX philosophy [13]: programs must do only one task and do it well; programs should be able to work together; programs must use a universal interface.

These ideas lead to a reusable component design, supporting modularization. The main point is that services are deployed in the production environment independently of each other, which is one of the main differences with most existing SOA solutions.

Microservices enables the creation of a system from a collection of small and isolated services capable of managing their own data [13]. REST[1] is frequently used in the development of Microservices, which produces synchronous communication among services; however, communication between Microservices must be based on the use of asynchronous messages. This is necessary to establish an asynchronous limit between each service whose objective is to decouple the communication flow of services from the perspectives of time and space[7]. The time-decoupled communication flow enables simultaneity while the decoupling in space enables the mobility and distribution that are indispensable perspectives for the development of applications in IoT scenarios. Asynchronous and non-blocking operations reduce the congestion of resources on the system and produce scalability and low latency. In addition to these issues, communication based on asynchronous messages produces systems with two characteristics: i) Resilience, which is associated with the ability of the system to handle failures; ii) Elasticity, which is associated with the ability of the system to scale horizontally. Systems that have their design constructed from these perspectives are considered reactive. According to Halbwachs [16], reactive systems are those which react continuously to their environment at a rate determined by that environment. The aim is to keep the distributed systems responsive, resilient and elastic based on the use of asynchronous messages.

Based on the aforementioned characteristics of Reactive Microservices for IoT, we propose an approach for modeling the design and implementation of reactive applications in IoT scenarios. IoT applications on devices need to ensure that the data/messages have been received and executed properly. Figure 1 illustrates our proposal, which includes components for the development of reactive Microservices. We have separated the proposed model into two modules: Application and Data. This approach will make it possible to scale data and application logic independently.

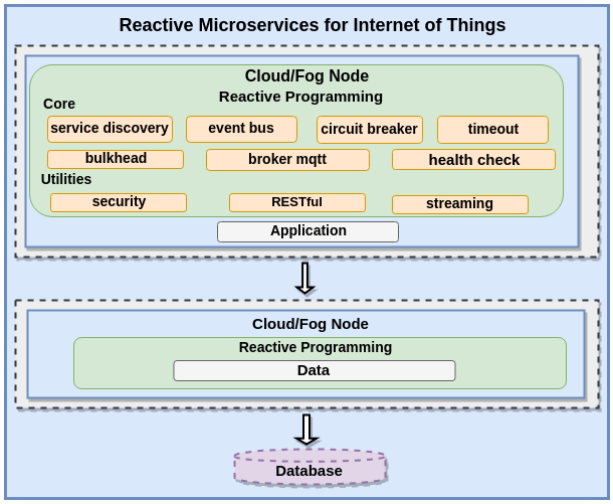


Figure 1: OSGI Reactive Microservices for IoT.

In the Data module (see Figure 1) each Microservices provides an interface for maintaining the data in the database. In this module, the developer can use a polyglot persistence strategy to solve different data storage problems [43].

In the Application module (see Figure 1) we consider the construction of reactive Microservices in the IoT from Core and Utilities perspectives. From the Core perspective, there are some required components for: the construction of a system based on the exchange of asynchronous messages (Event Bus and MQTT Broker components in Figure 1); the control of faults (Circuit Breaker component in Figure 1), to allow services to be found and operated independently wherever they are (Service Discovery component in Figure 1). From the Utilities perspective, there are components: to allow the display of devices from a RESTful API (RESTful component in Figure 1). In addition to these issues, streaming will have also be discussed, because a number of IoT scenarios currently make use of of data streaming, we implemented it in the case study discussed in Section 4. In the remainder of this section, we explain each one of the Application components.

The **Event Bus** component enable different parts of an application or service to communicate in a weakly coupled way. Messages are sent to addresses and consumers register at these addresses to receive the messages. The Event Bus is also clustered, which

[1]https://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm

Table 1: Publication Selected.

ID	Paper Name	Paper Format	Year
P1	Architecture of an interoperable IoT platform based on microservices [44]	Conference	2016
P2	Microservices approach for the internet of things[8]	Conference	2016
P3	Designing a Smart City Internet of Things Platform with Microservice Architecture [25]	Conference	2015
P4	SeCoS: Web of Things platform based on a microservices architecture and support of timeawareness [46]	Journal	2016
P5	Location and Context-Based Microservices for Mobile and Internet of Things Workloads [5]	Conference	2015
P6	Microservice-based IoT for smart buildings [23]	Conference	2017
P7	Microservices as Agents in IoT Systems [24]	Conference	2017
P8	Role Modeling of IoT services in industry domains[41]	Conference	2017
P9	Exploiting interoperable microservices in web objects enabled Internet of Things[20]	Conference	2017
P10	A Secure Microservice Framework for IoT[30]	Conference	2017
P11	Distribution of microservices for hardware interoperability in the Smart Grid[36]	Thesis	2015
P12	An Open IoT Framework Based on Microservices Architecture [39]	Journal	2017
P13	NIMBLE collaborative platform: Microservice architectural approach to federated IoT[17]	Conference	2017
P14	A Microservice Architecture for the Intranet of Things and Energy in Smart Buildings: Research Paper [6]	Conference	2016
P15	Microservices model in WoO based IoT platform for depressive disorder assistance [3]	Conference	2017
P16	Investigating Performance Metrics for Scaling Microservices in CloudIoT-Environments [15]	Conference	2018
P17	A Reactive Microservice Architectural Model with Asynchronous Programming and Observable... [33]	Thesis	2017
P18	Towards Osmotic Computing: Analyzing Overlay Network ... [9]	Conference	2018

Table 2: Research Contributions.

Classification	Paper	Contribution	Problems	Domain
Evaluation research	P2	Patterns and best practices	Self-Containment, Monitoring, Orchestration, Versioning	Agnostic
	P10	Architecture	Secure	Smart building
	P14	Middleware	reuse, deployment, development	CloudIoT
	P16	Performance metrics	Scaling	Agnostic
	P17	Methodology	Development of IoT middleware	Cloud, Edge and Fog
	P18	Middleware	Connectivity and Network Overlays	Vertical market
Solution Proposal	P1	Middleware	scalability/interoperability	Agnostic
	P4	Plataform	scalability/maintenance	Web of Things
	P5	Context-based applications	development	Agnostic
	P6	Prototype platform	scalability	Smart Building
	P7	Management Service	Collaboration of distributed modules	CloudIoT
	P9	Architecture	Interoperability	Web of objects
	P12	Framework	Heterogenety/Scalability/Discovery	Agnostic
	P13	Architecture	scalability/integration	Industry 4.0
	P15	Architecture	Interoperability	Web of Objects
Validation research	P11	Middleware	Heterogeneity	Smart Grid
Experience Report	P3	Plataform	Data management	Smart City
Philosophical papers	P8	Service Modeling	Modeling	Industry

means it can send messages over the network among senders and distributed consumers. In addition, the event bus sends messages to the instances that are available, and thus balances the load between the different nodes that are registered at the same address.

The responsibility of the **Circuit Breaker** component is to support resilience. This component acts as a state machine containing the following states: Closed: the request of a Microservices is forwarded to the operation. The component keeps a count of the number of recent faults, and if the call to the operation is unsuccessful, the component increments this count. If the number of recent faults exceeds a predetermined threshold in a given time period, the state transitions to Open: the Microservices request fails immediately and an exception is returned to the Microservices requestor. HALF-Open: When a certain number of requests happens successfully, it is assumed that the fault that was previously causing the failure has been fixed and the component switches to the Closed state. If any request fails, the component will assume that the failure is still present and reverts back to the Open state.

The **Timeout** component enables the isolation of faults and the use of a specified time- out to continue running a Microservices in a

degraded mode. On the other hand, **Bulkhead** allows the isolation and the low coupling among Microservices from the exchange of asynchronous messages which, in our proposal, is facilitated by the Event Bus and MQTT Broker components.

The **Health check** component provides an integrity checker to express the current state of the Microservices: UP or DOWN.

The **MQTT Broker** component provides a server that is capable of handling connections, communication and exchange of messages with remote MQTT clients. In our proposal, the broker MQTT is an implementation based on OSGI and therefore easy to deploy in different OSGI containers, as explained in Section 4.1.

In a dynamic environment such as the IoT, there must be a way to perform discovery and composition of services regardless of their location, which is known as location transparency. Therefore, the **Service Discovery** component provide means to discover services and to compose with discovered service, in order to attend non predetermined functionalities. This component also produces elasticity since services can decrease or increase at any time.

The **RESTful** component expose services as a RESTful API for access the IoT devices.

Data from various sources and domains produced by IoT is, in some cases, data streams, such as numerical data from different sensors or social media text inputs. As a result, the **Streaming** component allows the data streams produced by the various IoT devices to be processed and analyzed in real time, that is, as the data is being produced it will be processed and analyzed by this component.

The **Security** component provides functionalities for authorisation (i.e verify if a user has an authority) and authentication (i.e verifying the identity of a user).

In order to better understand some of the aforementioned components, Figure 2 illustrates the communication model of our proposal for the construction of IoT applications. In our proposal, the communication among the Microservices is based on the exchange of messages supported by the Event Bus, Service Discovery and MQTT Broker. Each Microservices must register in the Service Discovery, which sends the information regarding the Microservices to the Event Bus. In this context, the requests to a Microservices must be made from the Event Bus. The components Circuit Breaker, Timeout and Health check, are related to fault management, should asynchronously occur without errors.

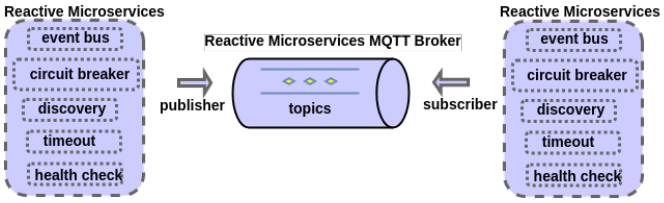


Figure 2: Overview of Reactive Microservices Core and their interactions.

4 A PLATFORM FOR REACTIVE MICROSERVICES IN IOT

This work extends the SOFT-IoT platform [34] to instantiate the model proposed in Section 3. Therefore, Sections 4.1 and 4.2 present, respectively, the extensions made in the SOFT-IoT architecture and in its implementation.

4.1 Concrete Architecture

We implemented the Microservices, according to the model proposed in this work in (Section 3), using the set of tools offered by Vert.x[2] and implemented them in Apache ServiceMix. One of the advantages of Vert.x compared to other technologies (e.g Akka[3], Ratpack[4]) is the provision of a set of components to design reactive applications in different languages. In addition, Vert.x has best performance than Akka and other frameworks in some benchmarks such as TechEmpower[5]. On the other hand, ServiceMix enables the deployment of run-time applications that meet the elasticity characteristics required in reactive systems.

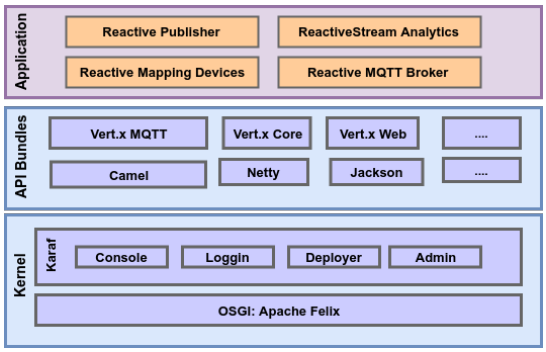


Figure 3: Reactive Microservices Infrastructure.

As shown in Figure 3, the Karaf container is the main component of Apache ServiceMix. For this reason, it is the foundation for the implementation of Bundles that support reactive Microservices (Figure 4). To implement the scenario described in Section 5.1 we implemented the following reactive Microservices:

4.1.1 Reactive Publisher. This Microservices collects data from sensors of the connected devices and with the Reactive MQTT Broker it provides this data for analysis of the Microservices Reactive Stream Analytics.

4.1.2 Reactive MQTT Broker. This Microservices is responsible for enabling communications with remote MQTT clients. In addition to the already known advantages of using a modular architecture, using the Vert.x MQTT Server API makes it possible to scale the Reactive MQTT Broker according to, for instance, the number of system cores and thus enabling horizontal scalability.

[2]https://vertx.io/
[3]https://akka.io/
[4]https://ratpack.io/
[5]https://www.techempower.com/benchmarks/

4.1.3 Reactive Mapping Devices. The Microservices Reactive Mapping Device is responsible for mapping devices connected to each Gateway. This Microservices can be used by other Microservices to obtain access to basic information about devices and their sensors/actuators.

4.1.4 Reactive Stream Analytics. The Microservices Reactive Stream Analytics aims to enable real-time data stream analysis at the edge of the network, enabling responses with low latency and reduced data traffic across the network. As runtime for data stream processing Apache Edgent[6]. It allows the implementation of a reactive programming model for the analysis of data stream and events on the limited devices of Fog Computing.

4.2 Implementation

Because in SOFT-IoT the IoT services may be deployed at devices (e.g. Gateways and Servers) that are located near the edge of the network, we implemented the Gateway presented in Figure 4. The following components were deployed on devices (gateway) with limited processing and memory capacities (Raspberry Pi and Intel Galileo Gen2 in Figure 4) and on small local servers: a service-oriented middleware based on the implementation of Apache OSGi (Karaf in Figure 4), a message-oriented reactive MQTT broker (Vertx MQTT in Figure 4) and APIs for service discovery, resiliency, security and elasticity support (Vert.x Core Vert.x Service Discovery, Vert.x Circuit Breaker, Vert.x Health Check, Vert.x Auth in Figure 4).

In the architecture shown in Figure 4, the service-oriented middleware provides interfaces for applications to access devices via RESTful Web Services (Vert.x Core HTTP in Figure 4). In turn, the Vert.x EventBus enables the exchange of asynchronous messages among the Microservices. Vert.x Service Discovery provides mobility support, considering IoT devices that provide services at the edge of the network may change their location at any time. Vert.x Auth enables the authentication and authorisation. Finally, the Vert.x Circuit Breaker, Vert.x Health Checks and Vert.x Core provides features to prevent cascading failures.

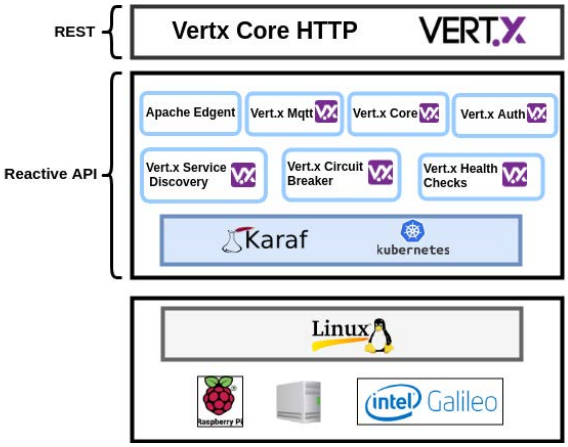


Figure 4: Gateway for Reactive Microservices IoT Plataform.

[6]http://edgent.apache.org/docs/home.html

5 EXPERIMENTS

5.1 Problem Statement: an IoT Scenario for Fish Farm

In order to validate the proposal of this work, the reactive Microservices platform was applied to manage the simulation of a Smart Water scenario applied in the Fish Farm domain. Smart Water is a smart network for the management of water environments, which involves the use of smart detection systems allocated throughout infrastructures and also in the environment for monitoring hydrological, hydraulic and quality parameters. Fish Farm is the cultivation of a variety of marine species, including: shellfish; sport fish; baits; ornamental fish; crustaceans; molluscs; algae. These species are bred in different aquatic environments, such as ponds, rivers, lakes and oceans. Managing a Fish Farm environment involves changes in water variables which directly affect the development of cultivated species. Therefore, the platform proposed in this paper, implemented in the Fish Farm scenario, aims at detecting anomalies or environmental events. The main objective is to respond to changes in water quality variables. The main sensors used in this type of environment are: temperature; pH; dissolved oxygen and nitrite concentration. In this scenario the first evaluated event is if the pH of the water is between 6 and 8. Another monitored measure is if the dissolved oxygen value in the water is lower than 5.5 mg/L. It is also verified if the concentration of nitrite in water is greater than 0. The architecture of the proposal of this work applied in the Fish Farm scenario is presented in Figure 5. This architecture allows scaling, low latency, resiliency and elasticity in the development of Fish Farm scenario.

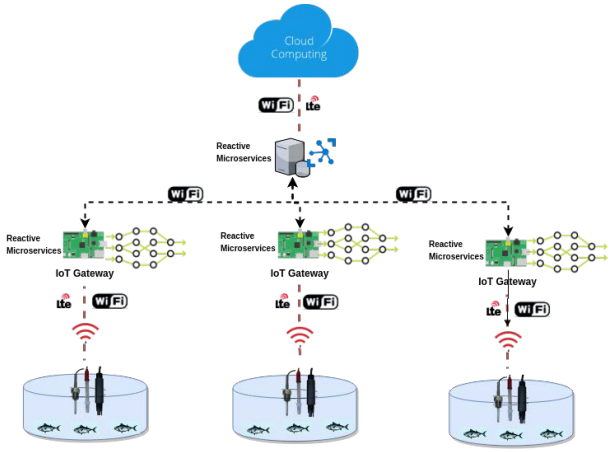


Figure 5: Smart Water Scenario.

5.2 Experimental Design

To perform a preliminary performance analysis of the proposed communication model with remote MQTT clients, we have developed an experimental study based on case study described in Section 5.1. In short, we use simulated sensors, which are Java programs that communicate with Reactive Broker MQTT by sending data in the same way as real sensors do. We have chosen simulated

sensors to perform viable experiments regarding scalability with a different number of sensors. In addition, the data produced by the sensors was evaluated as they were being produced, to meet the requirements of a Fish Farm environment as described in Section 5.1.

Table 3: Factors and levels used in the experimental design.

Factors	Levels	Response variables	
Architecture	TMA	Throughput	Latency
	RMA		
Number of Sensors	6		
	12		
IoT Gateway	1		
	2		

In order to conduct the full factorial design plan [19], the main parameters from our experimental work are shown in the Table 3. **(I - Factors)**: Architecture, Number of Sensors, and IoT Gateway. These factors are chosen because several previous experiments show the meaningful influences performed by them in the environment.

(II - Levels): 6 or 12 sensors; Intel Core Duo (1) or Raspberry pi 3 (2) as IoT Gateway. These levels are chosen to show a light scenario and other overloaded scenario of Microservices provided by intelligent and constrained objects at the edge of the network. To show the differences between the solutions, we designed the IoT application discussed in Section 4 with the Reactive Microservices architecture (RMA) (model and platform proposed) and compared the same IoT application using a traditional Microservices Architecture (TMA). The TMA approach discussed in this work is a FOG-based platform that enables the implementation of Microservices for the construction of IoT applications.

In turn, **(III - Response Variables)** are defined by the Throughput Published – number of messages that the reactive MQTT Broker publishes at the edge of the network per unit time; Throughput Received – number of messages that the reactive MQTT Broker receives at the edge of the network per unit time; Throughput Messages – number of messages that the reactive Stream Analytics processes at the edge of the network per unit time; and Latency – time required since a publisher sends a message until a subscriber receives it.

In the experiments model, the average and standard deviation are calculated for each combination (factors and levels), which will serve as the basis for calculating the sum of squares, resulting in the influence of each factor on the response variables. Figure 6 shows the influence of each factor and its combinations on the overall Latency and Throughput Published of the experiment. In addition, Figure 7 shows the influence of each factor and its combinations on the overall Throughput Received and Throughput Stream of the experiment. It is considered as significant influence the ability of a given factor to modify the values of the response.

The following setup were used to conduct the experiments: Operating System Ubuntu 16.04 LTS, Intel Core Duo-2.1 GHz processor/ 3GB of RAM and Operation System Debian, Raspberry pi 3 1.2GHZ processor/1GB of RAM. To generate the results from planning model used in this work, 8 simulations - each with 10 replications and each replication lasting 600 seconds-were performed for each scenario, that is, for each combination of factors and levels. All

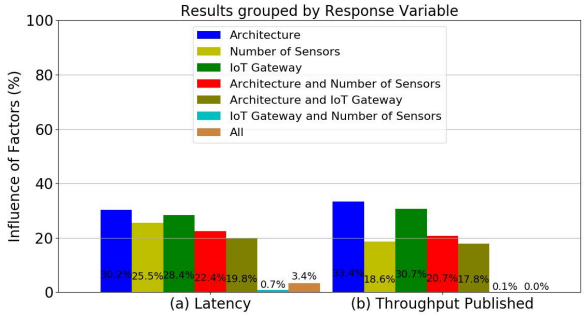


Figure 6: Influence of factors on the Latency and Throughput Published

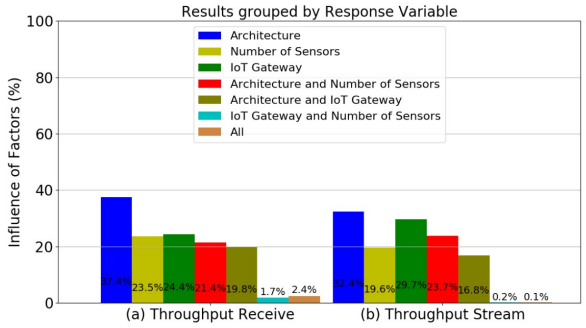


Figure 7: Influence of factors on the Throughput Received and Throughput Stream

values in the replication stage were used, since the results obtained in each scenario did not present significant changes. According to Figure 6 the most significant factor is the Architecture with 30.2% of influence on latency and 33.4% of influence on throughput published. Similarly, according to Figure 7 the most significant factor is the Architecture with 37.4% of influence on throughput receive and 32.4% of influence on throughput stream.

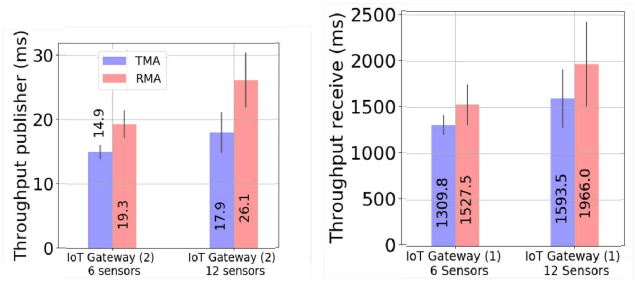
The results shown the factors used in the design of experiments had different influences on the response variables. Furthermore, the proposed platform and architecture had a considerable influence on scenarios (Throughput Published, Throughput Received, Throughput Messages and Latency). In order to increase analytical information of our work, Section 5.3 presents a detailed analysis of how the reactive platform and architecture impacts the response variables of the system.

5.3 Analysis of Results

In this section we have compared the results with and without our proposal in the same scenario (factors with the same levels). In order to do that, we have presented the data generated from our experiments for combination (a scenario) of factors and levels defined in Table 3.

The use of the RMA approach has improved the Throughput Published and Throughput Received for the scenarios shown in Figure 8(a) and Figure 8(b).

In addition, in Figure 10(a) and Figure 10(b) we show scenarios (i.e. we varied the number of sensors and the type of the IoT gateway) in which data analysis at the edge of the network is performed. Also, in this scenarios, our approach (RMA) proved to be more efficient for designing IoT applications compared to the traditional approach (TMA).



(a) Scenario with 6 sensors (b) Scenario with 12 sensors

Figure 8: Analysis for Throughput Publisher and Throughput Receive

Finally, Figure 9 also shows that the adoption of our proposal has reduced the overall response time of the IoT application.

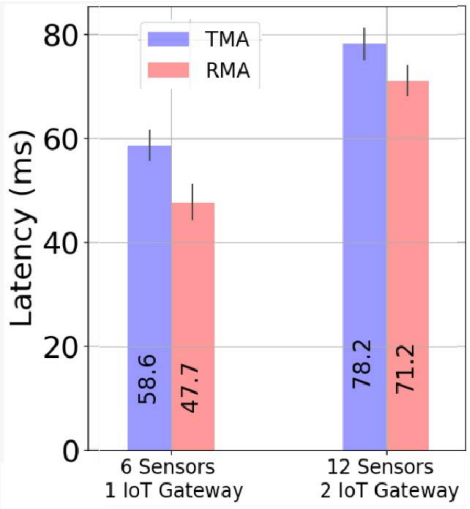
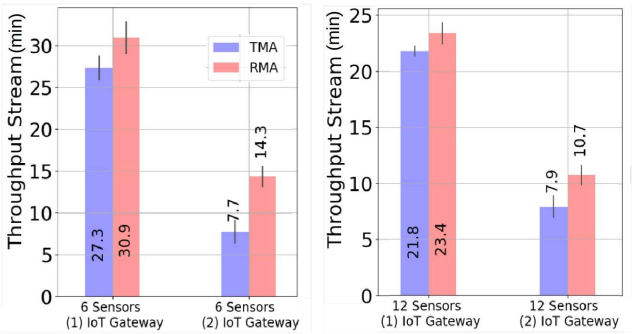


Figure 9: Analysis for Latency

6 FINAL REMARKS

In this paper we have presented an architecture proposal for the construction of reactive Microservices, whose objective is to enable the construction of applications in IoT that are performative, resilient and scalable, which is made possible by the exchange of asynchronous messages. As proof of concept, we have compared our proposal with a FOG-based platform that enables the construction of IoT applications that adopt Microservices. The results showed that, for the scenarios evaluated, the use of our approach resulted in better performance when compared to the same scenarios without our approach of reactive Microservices.



(a) Scenario with 6 sensors (b) Scenario with 12 sensors

Figure 10: Analysis for Throughput Stream

As future work, we are developing an experimental study to evaluate the resilience of IoT applications built from the proposed model. In this study we will use Chaos Monkey[7], which is a proposal of NetFlix to terminate instances in production and Gremlin framework[8] which is a proposal of IBM to conduct failure recovery testing on Microservice applications and evaluate if the IoT applications are resilient to the induced faults. In addition, we are carrying out a comparison study on the advantages and disadvantages of using our proposal in the implementation of IoT applications in Cloud Computing and Fog Computing.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. 2015. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys Tutorials* 17, 4 (Fourthquarter 2015), 2347–2376. https://doi.org/10.1109/COMST.2015.2444095

[2] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. 2015. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials* 17, 4 (2015), 2347–2376.

[3] S. Ali, M. G. Kibria, M. A. Jarwar, S. Kumar, and I. Chong. 2017. Microservices model in WoO based IoT platform for depressive disorder assistance. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, Jeju, South Korea, 864–866. https://doi.org/10.1109/ICTC.2017.8190800

[4] Edgardo Avilés-López and J Antonio García-Macías. 2009. TinySOA: a service-oriented architecture for wireless sensor networks. *Service Oriented Computing and Applications* 3, 2 (2009), 99–108.

[5] P. Bak, R. Melamed, D. Moshkovich, Y. Nardi, H. Ship, and A. Yaeli. 2015. Location and Context-Based Microservices for Mobile and Internet of Things Workloads. In *2015 IEEE International Conference on Mobile Services*. IEEE, New York, NY, USA, 1–8. https://doi.org/10.1109/MobServ.2015.11

[6] Kaibin Bao, Ingo Mauser, Sebastian Kochanneck, Huiwen Xu, and Hartmut Schmeck. 2016. A Microservice Architecture for the Intranet of Things and Energy in Smart Buildings: Research Paper. In *Proceedings of the 1st International Workshop on Mashups of Things and APIs (MOTA '16)*. ACM, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/3007203.3007215

[7] Jonas Bonér. 2016. *Reactive microservices architecture: design principles for distributed systems.* O'Reilly Media, Gravenstein Highway North, Sebastopol.

[7] https://netflix.github.io/chaosmonkey/

[8] https://developer.ibm.com/code/open/projects/gremlin/

[8] B. Butzin, F. Golatowski, and D. Timmermann. 2016. Microservices approach for the internet of things. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation*. IEEE, Berlin, Germany, 1–6. https://doi.org/10.1109/ETFA.2016.7733707

[9] A. Buzachis, A. Galletta, L. Carnevale, A. Celesti, M. Fazio, and M. Villari. 2018. Towards Osmotic Computing: Analyzing Overlay Network Solutions to Optimize the Deployment of Container-Based Microservices in Fog, Edge and IoT Environments. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*. IEEE, Washington, DC, USA, 1–10. https://doi.org/10.1109/CFEC.2018.8358729

[10] G. Campeanu. 2018. A mapping study on microservice architectures of Internet of Things and cloud computing solutions. In *2018 7th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, Budva, Montenegro, 1–4. https://doi.org/10.1109/MECO.2018.8406008

[11] Tomas Cerny, Michael J. Donahoo, and Michal Trnka. 2018. Contextual Understanding of Microservice Architecture: Current and Future Directions. *SIGAPP Appl. Comput. Rev.* 17, 4 (Jan. 2018), 29–45. https://doi.org/10.1145/3183628.3183631

[12] J. Eliasson, J. Delsing, A. Raayatinezhad, and R. Kyusakov. 2013. A SOA-based framework for integration of intelligent rock bolts with internet of things. In *2013 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, Cape Town, South Africa, 1962–1967. https://doi.org/10.1109/ICIT.2013.6505979

[13] Martin Fowler and James Lewis. 2014. Microservices. http://martinfowler.com/articles/microservices.html. [Online; accessed 26-April-2018].

[14] P. D. Francesco, I. Malavolta, and P. Lago. 2017. Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption. In *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, Gothenburg, Sweden, 21–30. https://doi.org/10.1109/ICSA.2017.24

[15] Manuel Gotin, Felix Lösch, Robert Heinrich, and Ralf Reussner. 2018. Investigating Performance Metrics for Scaling Microservices in CloudIoT-Environments. In *Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (ICPE '18)*. ACM, New York, NY, USA, 157–167. https://doi.org/10.1145/3184407.3184430

[16] Nicolas Halbwachs. 2013. *Synchronous programming of reactive systems*. Vol. 215. Springer Science & Business Media, Grenoble, France.

[17] J. Innerbichler, S. Gonul, V. Damjanovic-Behrendt, B. Mandler, and F. Strohmeier. 2017. NIMBLE collaborative platform: Microservice architectural approach to federated IoT. In *2017 Global Internet of Things Summit (GIoTS)*. IEEE, Geneva, Switzerland, 1–6. https://doi.org/10.1109/GIOTS.2017.8016216

[18] Valérie Issarny, Georgios Bouloukakis, Nikolaos Georgantas, and Benjamin Billet. 2016. Revisiting Service-Oriented Architecture for the IoT: A Middleware Perspective. In *Service-Oriented Computing*, Quan Z. Sheng, Eleni Stroulia, Samir Tata, and Sami Bhiri (Eds.). Springer International Publishing, Cham, 3–17.

[19] Raj Jain. 1990. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, ...

[20] M. A. Jarwar, S. Ali, M. G. Kibria, S. Kumar, and I. Chong. 2017. Exploiting interoperable microservices in web objects enabled Internet of Things. In *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, Milan, Italy, 49–54. https://doi.org/10.1109/ICUFN.2017.7993746

[21] I. Järvinen, L. Daniel, and M. Kojo. 2015. Experimental evaluation of alternative congestion control algorithms for Constrained Application Protocol (CoAP). In *IEEE World Forum on Internet of Things*. IEEE, Milan, Italy, 453–458. https://doi.org/10.1109/WF-IoT.2015.7389097

[22] R. Khan, S. U. Khan, R. Zaheer, and S. Khan. 2012. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *2012 10th International Conference on Frontiers of Information Technology*, Vol. 1. 1, https://ieeexplore.ieee.org/document/6424332, 257–260. https://doi.org/10.1109/FIT.2012.53

[23] K. Khanda, D. Salikhov, K. Gusmanov, M. Mazzara, and N. Mavridis. 2017. Microservice-Based IoT for Smart Buildings. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Vol. =. IEEE, Taipei, Taiwan, 302–308. https://doi.org/10.1109/WAINA.2017.77

[24] Petar Krivic, Pavle Skocir, Mario Kusek, and Gordan Jezic. 2017. Microservices as Agents in IoT Systems. In *Agent and Multi-Agent Systems: Technology and Applications*, Gordan Jezic, Mario Kusek, Yun-Heh Jessica Chen-Burger, Robert J. Howlett, and Lakhmi C. Jain (Eds.). Springer, Cham, 22–31.

[25] A. Krylovskiy, M. Jahn, and E. Patti. 2015. Designing a Smart City Internet of Things Platform with Microservice Architecture. In *3rd International Conference on Future Internet of Things and Cloud*. IEEE, Rome, Italy, 25–30. https://doi.org/10.1109/FiCloud.2015.55

[26] K. Kumar, C. Mouli, and U. Kumar. 2017. A survey on the Internet of Things-based service orientated architecture. In *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*. IEEE, Mysuru, India, 435–439. https://doi.org/10.1109/ICEECCOT.2017.8284544

[27] In Lee and Kyoochun Lee. 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons* 58, 4 (2015), 431 – 440. https://doi.org/10.1016/j.bushor.2015.03.008

[28] In Lee and Kyoochun Lee. 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons* 58, 4 (2015), 431–440.

[29] Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information Systems Frontiers* 17, 2 (01 Apr 2015), 243–259. https://doi.org/10.1007/s10796-014-9492-7

[30] D. Lu, D. Huang, A. Walenstein, and D. Medhi. 2017. A Secure Microservice Framework for IoT. In *2017 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. IEEE, San Francisco, CA, USA, 9–18. https://doi.org/10.1109/SOSE.2017.27

[31] Ima Miri. 2017. Microservices vs. SOA. https://dzone.com/articles/microservices-vs-soa-2. [Online; accessed 26-November-2018].

[32] Sam Newman. 2015. *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", Gravenstein Highway North, Sebastopol.

[33] Pedro Manuel Taveras Núñez. 2017. *A Reactive Microservice Architectural Model with Asynchronous Programming and Observable Streams as an Approach to Developing IoT Middleware*. Ph.D. Dissertation. Colorado Technical University.

[34] C. Prazeres and M. Serrano. 2016. SOFT-IoT: Self-Organizing FOG of Things. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. IEEE, Crans-Montana, Switzerland, 803–808. https://doi.org/10.1109/WAINA.2016.153

[35] Lukas Reinfurt, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Andreas Riegg. 2016. Internet of Things Patterns. In *Proceedings of the 21st European Conference on Pattern Languages of Programs (EuroPlop '16)*. ACM, New York, NY, USA, Article 5, 21 pages. https://doi.org/10.1145/3011784.3011789

[36] Jesús Rodríguez Molina. 2015. *Distribution of microservices for hardware interoperability in the Smart Grid*. Ph.D. Dissertation. ETSIS_Telecomunicacion.

[37] C. Santana, B. Alencar, and C. Prazeres. 2018. Microservices: A Mapping Study for Internet of Things Solutions. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. IEEE, Cambridge, MA, USA, USA, 1–4. https://doi.org/10.1109/NCA.2018.8548331

[38] N. Shahid and S. Aneja. 2017. Internet of Things: Vision, application areas and research challenges. In *International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*. IEEE, Palladam, India, 583–587. https://doi.org/10.1109/I-SMAC.2017.8058246

[39] L. Sun, Y. Li, and R. A. Memon. 2017. An open IoT framework based on microservices architecture. *China Communications* 14, 2 (February 2017), 154–162. https://doi.org/10.1109/CC.2017.7868163

[40] Nguyen Khoi Tran, Quan Z. Sheng, Muhammad Ali Babar, and Lina Yao. 2017. Searching the Web of Things: State of the Art, Challenges, and Solutions. *ACM Comput. Surv.* 50, 4, Article 55 (Aug. 2017), 34 pages. https://doi.org/10.1145/3092695

[41] P. Tsoutsa, P. Fitsilis, and O. Ragos. 2017. Role Modeling of IoT Services in Industry Domains. In *Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences (ICMSS '17)*. ACM, New York, NY, USA, 290–295. https://doi.org/10.1145/3034950.3035002

[42] Rob Van Der Meulen. 2015. Gartner Says 6.4 Billion Connected 'Things' Will Be in Use in 2016, Up 30 Percent From 2015.

[43] Luís H. N. Villaça, Leonardo G. Azevedo, and Fernanda Baião. 2018. Query Strategies on Polyglot Persistence in Microservices. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC '18)*. ACM, New York, NY, USA, 1725–1732. https://doi.org/10.1145/3167132.3167316

[44] T. Vresk and I. Čavrak. 2016. Architecture of an interoperable IoT platform based on microservices. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, Opatija, Croatia, 1196–1201. https://doi.org/10.1109/MIPRO.2016.7522321

[45] Eberhard Wolff. 2016. *Microservices: Flexible Software Architecture*. Addison-Wesley Professional, ..

[46] Herwig Zeiner, Michael Goller, Víctor Juan Expósito Jiménez, Florian Salmhofer, and Werner Haas. 2016. SeCoS: Web of Things platform based on a microservices architecture and support of time-awareness. *e & i Elektrotechnik und Informationstechnik* 133, 3 (01 Jun 2016), 158–162. https://doi.org/10.1007/s00502-016-0404-z

[47] Y. Zhang, L. Duan, and J. L. Chen. 2014. Event-Driven SOA for IoT Services. In *2014 IEEE International Conference on Services Computing*. IEEE, Anchorage, AK, USA, 629–636. https://doi.org/10.1109/SCC.2014.88