A Service Orchestration Architecture for Fog-enabled Infrastructures

Mathias Santos de Brito, Saiful Hoque, Thomas Magedanz
Architekturen der Vermittlungsknoten (AV)
Technische Universität Berlin, Germany

Ronald Steinke, Alexander Willner, Daniel Nehls, Oliver Keils, Florian Schreiner
Software-based Infrastructures (NGNI)
Fraunhofer FOKUS, Berlin, Germany

***Abstract*—The development of Fog Computing technology is crucial to address the challenges to come with the mass adoption of Internet Of Things technology, where the generation of data tends to grow at an unprecedented pace. The technology brings computing power to the surrounds of devices, to offer local processing, filtering, storage and analysis of data and control over actuators. Orchestration is a requirement of Fog Computing technology to deliver services, based on the composition of microservices. It must take into consideration the heterogeneity of the IoT environment and device's capabilities and constraints. This heterogeneity requires a different approach for orchestration, be it regarding infrastructure management, node selection and/or service placement. Orchestrations shall be manually or automatically started through event triggers. Also, the Orchestrator must be flexible enough to work in a centralized or distributed fashion. Orchestration is still a hot topic and can be seen in different areas, especially in the Service Oriented Architectures, hardware virtualization, in the Cloud, and in Network Virtualization Function. However, the architecture of these solutions is not enough to handle Fog Requirements, specially Fog's heterogeneity, and dynamics. In this paper, we propose an architecture for Orchestration for the Fog Computing environment. We developed a prototype to prof some concepts. We discuss in this paper the implementation, and the tools chose, and their roles. We end the paper with a discussion on performance indicators and future direction on the evaluation of non-functional aspects of the Architecture.**

***Index Terms*—Fog Computing, Orchestration, Infrastructure Management**

I. Introduction

With the advent of Internet of Things (IoT) [1], expectations are that by 2020 there will be 6.4 Billion Connected devices generating an unprecedented amount of data, the storage, and processing of this data is a major challenge today and will become even more challenging in the future[9]. The attempt to use the cloud as the only solution for this problem stopped in the stochastic nature of the Internet, and also the need to transfer massive amounts of data towards the cloud, and the inability to achieve responsiveness in some application fields that demands rapid reaction to events. The Fog Computing concept[6] arrives to fill this gap, by bringing computing power and storage to the surrounds of devices while keeping the smooth approach of the Cloud by introducing virtualization as one of its fundamental aspects. The so-called Fog Nodes are computational nodes, that must be able to talk with a variety of devices, sensors, and actuators and offers services based on data gathered and eventually processed/filtered locally. This services can be deployed dynamically and are composed of a set of microservices. Nowadays, Orchestration is an overused word; it is presented in different scenarios to indicate the management of the life-cycle of a one or more distributed components that together deliver a service or functionality. As an example of multiple uses of the term we have Virtual Machines and Container Orchestration; in the Service Oriented Architectures (SOA)[7] world Services Orchestration; and in networking, Network Function Virtualization (NFV) [4, 17] Orchestration. In all these areas Orchestration Solutions have been proposed. However, work in the direction of Fog Computing still needed [12].

Although orchestration of all sorts is relatively mature in datacenters and clusters, in the Fog, it still a challenging due to the dynamic and heterogeneous nature of the devices involved in the process. The number of constraints in an orchestration and its building blocks (e.g. microservices) can be many. Also, the orchestration can be done centrally or distributed, meaning that a Fog Node can start orchestrating services if necessary and capable of, trying to maintain functional its core services. The orchestration feature of the Fog also ensures the programmability aspect of its fog nodes. Deployment of new functionalities, which can rely on a variety of capabilities, locally attached devices, sensors, and actuators, needs to ensure the quality of service necessary.

In this paper, we present an Architecture for service orchestration based on the core requirements of Fog Computing. Based on a virtualized environment, where Fog Nodes are capable of running virtualized and containerized applications and services, offering them access to attached/connected devices, over different communication technologies, to accomplish their tasks.

The remainder of the paper is structured as follows. We give a brief overview of related work in the context of orchestration in virtualized environments in Sec. II. In the subsequent Sec. III the architecture is presented. A broad discussion and outlook are presented in Sec. IV. Finally, in Sec. V, we present the conclusions and considerations and highlight future work.

II. Related Work

To place the contribution of this paper in context and identify the gap the work is intended to fill, we discuss in this section, state of the art on Orchestration of services in Virtualized Environments. For that purpose, we look into details on the European Telecommunications Standards Institute (ETSI) NFV Management and Orchestration (MANO) [8] and open and commercial solutions for the Cloud.

Services Orchestration is not a new topic[16]. However, its application in different areas of distributed system keeps it as an actual issue. The idea, popular in the context of SOA and enterprise systems, grew in popularity with the REST architectural style and Cloud Computing. With microservices, virtualization, containerization and NFV and their exposition as services, the word service orchestration became overused. The consequence was the appearance of multiple solutions in the literature. From efforts to solve the problem, standardized solutions as the case of ETSI NFV MANO emerged. In fact, the literature shows that orchestration was extensively discussed in the NFV community[13]. The need to define, for each application domain, the scope, properties, and requirements of service orchestration concepts is necessary, and this is not different in the Fog Computing environment. Due to the maturity of the discussion in the NFV community, we consider the findings in the field as a starting point for a Fog Orchestration Solution.

A. *ETSI NFV MANO Reference Architecture*

The ETSI NFV MANO is clean and flexible. Initially conceived to orchestrate VNF, due to its characteristics it is being considered to be adapted to handle other kinds of services. Core components like the Virtual Infrastructure Manager (VIM) that manages the virtualized infrastructure, the VNF Manager responsible for handling the life-cycle of VNFs, and the NFV Orchestrator who does Resource and Service orchestration, can be generalized to other scenarios other than NFV. Already implemented by different solutions ETSI NFV MANO consolidates its position in the field[15]. Indeed, we borrow from ETSI MANO some characteristics as we will see in Sec. III where we make a deeper comparison on it, however, differences of the Fog Computing environment needs to be considered. Specifically, the infrastructure management can see significant differences. A Fog Node can start, autonomously, to orchestrate. It can leave the condition of resource to become an orchestrator. Also, while the ETSI MANO is intended to deal with computing nodes that can vary on hardware (CPU, memory, network) and software specifications (Hypervisor, Operating System, etc.), in the Fog Environment Physical devices have capabilities other than that(e.g. sensors and actuators). These devices play a major role in the process and need to be taken into consideration in the architecture. In the MANO approach we could model these devices as NFVI Resources, however in its specification reads: "Management of non-virtualised resources is restricted to provisioning connectivity to PNFs (Physical Network Functions), necessary when an NS instance includes a PNF that needs to connect to a VNF, or when the NS instance is distributed across multiple NFVI-PoPs or N-PoPs."[8]. In the Fog, these devices can be anything, known or unknown, which brings challenges on how to model information about these devices, like vendor, identification and device capabilities.

B. *Container Orchestrators*

Since our initial solution supports only Docker Containers to delivery services, an overview of the solutions in container Orchestration is necessary.

In the market it is available many container orchestration solutions, who can manage the entire life-cycle of containers or a set of related containers scaling them up or down, migrating, self-healing, doing storage orchestration and other features. However, these solutions, Kubernetes[1], Mesos Marathon[2], Docker Swarm[3], to cite some of the most popular, targets clusters and datacenters where one can achieve a high level of homogeneity. Again the solutions are not device-aware, customization of requirements are not part of the core of the solutions, so one cannot enforce the deployment of a specified container in nodes that have some particular device attached to it. In both Mesos Marathon and Kubernetes, it is possible to set CPU, Memory and Storage Constraints. In the case of Docker Swarm, it offers a constraint system based on labels assigned to containers, images, and nodes, the placement of a container in a node is done only if labels from both the desired image/container matches with the node's labels. This is a very flexible and extensible solution, however, the management of the labels is manual and do not cope with the dynamics of the Fog. Different from service orchestration, where the functionalities delivered by deployed artifacts and their relationship are central, in the container orchestration scenario don't. However extending these tools to address our specific needs for container orchestration is possible and investigated in this work. It is important to note that these solutions do not compete with MANO, that in its context consider these container orchestrators as Virtualized Infrastructures that offers resources to the NFVO.

C. *Mobile Edge Computing Reference Architecture*

In the field of Mobile Edge Computing[10], the ETSI MEC Reference Architecture[5] conceives an Orchestrator that have very similar requirements to the Fog. The community still discussing the differences and similarities of both technologies, in [12] the authors examine the differences between each approach. Like Fog concepts, MEC emphasizes the need to consider a comprehensive set of constraints, and also refers to triggered application instantiation and relocations, one of the characteristics of our solution. However, a detailed specification or/and Reference Architecture for the orchestrator still missing. Again MANO can be an important starting point to an initial proposal[2].

[1]https://kubernetes.io
[2]http://mesos.apache.org
[3]https://www.docker.com/products/docker-swarm

D. OpenFog Reference Architecture for Fog Computing

The just released OpenFog Reference Architecture[6] clarifies the characteristics and requirements for the Fog, focusing primarily in the Fog Node. The mentions to the Orchestrator are superficial, and it is expected that new specifications in this direction. In this work, we will map the OpenFog RA architecture to our solution and present a broad discussion in Sec. IV.

E. TOSCA

Last but not least, Topology and Orchestration Specification for Cloud Applications (TOSCA) [14] in recent years, is becoming the *de facto* standard for modeling service orchestration, highly extensible and flexible. This work relies on TOSCA to define the services, its building blocks, requirements and capabilities, which should be taken into consideration by the orchestrator. In the orchestration scenario, TOSCA has been applied with success; this is the case of OpenStack Heat[4], Open Baton[3] and Cloudify[5].

At this point, it is clear that the primary purpose of this work is to, based on the outputs of the current status in the field of orchestration, propose a Fog Orchestrator Architecture that reflects the characteristics of Fog Computing, especially in what concern the resource heterogeneity and constraints. The definition of the scope and requirements is important since it will show the necessity for a new approach for orchestration, be it by extending existent architectures or proposing new ones.

III. The Fog Orchestrator Architecture

Figure 2 provides an overview of the proposed architecture. The solution is based on two essential components, Fog Orchestrator (FO) and Fog Orchestration Agent (FOA), each composed of a set of subcomponents. To better understand our approach we need to keep in mind that both the FO and the FOA, can reside on the same machine, as shown in the upper box. However, this is not mandatory.

Before explaining each component, it is important to clarify the use of the term resource in our context. It is anything that can be accessible through a computer interface and made available to the resource manager, figure 1 list some examples of resources that can be managed by a Fog Node and used in a Fog Infrastructure.

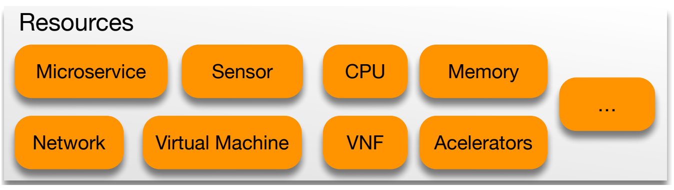

Fig. 1: Resources in a Fog Infrastructure

[4]https://www.openstack.org
[5]http://getcloudify.org

A. Fog Orchestration Agent

We assume that every Fog Node will have an instance of the Fog Orchestration Agent (FOA) component. The FO is responsible for: *i.)* managing locally the containerized services running on the Node, *ii.)* start and stop resources when requested by an FO if able or authorized to, *iii.)* manages its resources, *iv.)* start, stop containers when some event happens following pre-defined rules (e.g., a certain device was attached), *v.)* detect and manage attached devices, *vi.)* interface with a local monitoring system and manage monitoring data, *vii.)* offer a standardized way to do M2M communication taking into consideration different protocols while providing standardized access to the data. Although we use the term container, the FOA can be able to deal with different virtualization and container solutions.

We must consider that a Fog Node can be a long time without contact with a central node or even never contact one, it is an autonomous system. In the absence of a central node, and to accomplish its tasks the FOA can become an FO, and start trying to set up infrastructures based on visible Fog Nodes to keep the services required by it. Note that, to assume the role of an orchestrator the Fog Node must be able to run the necessary components and have the necessary computing power and resources to do so. The Fog Nodes assuming the role of Orchestrator still part of its domain and should return to its FOA status if connectivity to the domain's central orchestrator is back.

1) The Virtualization/Container Adapter: act as an adapter to abstract the underlying virtualization solution, the FOA must be able to easily change adapters or even use multiple of them at the same time.

2) The Resource Manager: subcomponent will manage resources locally, be it a microservice or a device, a VNF or the CPU. The lifecycle of the resource is the responsibility of this component. If the Orchestrator needs some resource in a Fog Node, it will need to deal with the Resource Manager.

3) The Monitoring: component will gather information about the current state of the machine and the condition of resources like CPU, memory, storage, battery, from sensors like temperature, etc. This component can also gather information from attached sensors through the M2M Framework; the FO can then retrieve these data if necessary.

It is desired, all the Fog Nodes to be M2M capable by employing some standard architecture, for example, OneM2M. The M2M Framework is responsible for talking with devices, in different M2M protocols and allowing applications and services to make use of devices data in a standard way. The lack of M2M capabilities does not prevent the Fog Node from joining an infrastructure.

4) Self-Announcement and FO detection: are another important aspects of Fog Nodes. When connected to a network the Fog Node can announce itself, informing if it is acting as FO or FOA. It can be added to some infrastructure if it detects an FO or receives responses from FOAs asking to join it as FOA if it is playing the role of the FO.

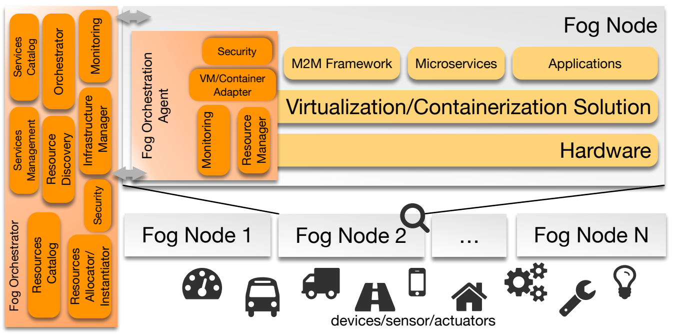

Fig. 2: The Fog Orchestrator Components Overview.

H

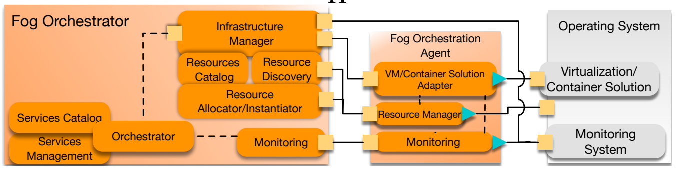

Fig. 3: Relationship between components and subcomponents in the architecture.

B. The Fog Orchestrator

From the perspective of an FO, it sees the infrastructure as a centralized system. It manages a group of Fog Nodes as its physical infrastructure. It can organize Fog Nodes in groups, what we will call Logical Infrastructures, allowing the handle of multiple domains. Also, these Logical Infrastructures can be useful to separate nodes by their capabilities or objectives, for example.

1) The Infrastructure Manager: the first task is to manage the life-cycle of the Fog Nodes associated to the FO, and its connections. In figure 2, we can see that the Infrastructure Manager overlaps with a set of other components, this overlap means that they are related and can be implemented as a single component or separately. It has a Resource Catalog subcomponent which holds a model representing each known resource. The discovery of resources is the responsibility of the Resource Discovery component, who keeps a database of resources found in the known Fog Nodes. Another important task done in the context of the Infrastructure Manager is the instantiation and allocation of resources. Each orchestration is built upon the composition of resources, like devices, sensors, actuators, and microservices, etc., after the FO make a plan for a given orchestration template it needs to instantiate or allocate the necessary resources, this task is done by the Resource Allocator/Instantiator subcomponent.

2) The Orchestrator component: While the Infrastructure Manager manages resources, the Orchestrator compose them to offer new services. Based on an orchestration template the Orchestrator needs to consult its catalogs and the Infrastructure Manager catalogs as well monitoring information to build an Orchestration Plan and instantiate it. The Orchestrator can start an orchestration manually based on user input or automatically by reacting to events (nodes arriving in the infrastructure, resources becoming unavailable/available, etc.). Note that this is not the same behavior of the FOA since it can handle resources based on events that happen locally in the Node, this means to start/stop a container, detach a device, etc. The FO, on the other side, manages the lifecycle of the orchestrations, which involves multiple resources, maybe residing in different Fog Nodes.

C. Relationship Between Components

Figure 3 shows the interactions between components. The interaction is achieved through interfaces represented by small boxes attached to the components. We do not specify any technical aspects of these interfaces, and we only show how the system components and subcomponents should communicate, and how they relate to each other. Dashed lines mean internal communication between subcomponents, that is not of interest to the outside world and can vary between implementations. The full lines indicate that the communication involves another Component and some level of standardization must be enforced. This work doesn't have the intention to specify such interfaces. The triangular shapes attached to some subcomponents means Adapters, meaning that the interface between the current Operating System, Virtualization/Container solution and monitoring tool can vary and that any implementation of the architecture needs to provide the means to change these adapters, supporting different solutions. To finalize, overlapping subcomponents, despite the fact that these subcomponents can exist and run independently, indicates a heavy dependency. That means that the functionality of some depend on the others, and can be implemented as one single piece of software, but not required.

D. Multi-Tenancy and Multi-Domain

Multi-tenancy and Multi-Domain are expected as a characteristic of Fog Nodes. Multiple services can run in a single Fog Node. Also, one service can be spread in many Fog Nodes. That brings connectivity requirements, be it internally in the virtualized environment in the Fog Node, be it between fog nodes. The architecture doesn't specify details on connectivity and communication, and implementations are free to define it. Once established connectivity, communication can be achieved by Messaging Queuing system, or publish/subscribe approaches, for example. The decision to not define one approach is the different approaches being taken by the development community to compose Microservices; we understand that flexibility is necessary for the Fog Environment. Also, Multi-domain is needed to mention. A Fog Node can join multiple domains, while an Orchestrator can separate its infrastructure in multiple domains. It is important to have Fog Computing providers serving multiple clients, and clients making use of multiple providers. Let's assume a truck transporting its load between different companies. It must be able to join both the sender Fog infrastructure as well the receiver. In this regard, the Security component plays an important role, by guaranteeing that only authorized Fog Nodes can join a Fog Infrastructure and also that a Fog Node is joining a safe domain.

E. security

The communication between components comprise the rules defined in the Security component. Since a Fog Node can take part in multiple domains, authorization and authentication mechanisms must take that into consideration. Secure communication channels also are desired, but no enforced. It is highly desired the implementation of data cryptography to store sensitive data, since in many applications Fog Nodes can be used to keep sensitive data, remember that Fog Nodes can be fixed or not, and one example of on-premise storage are Smart Factories, who should opt to keep production information, machinery condition and other sensitive information on premise. In this direction, the Security component must make sure that security rules are being enforced in the system.

IV. Discussion and Outlook

In this section, we discuss the importance of our proposal by analyzing its synergy with the OpenFog RA and by comparing it to the ETSI NFV MANO what is considered as a potential base for a Fog Orchestration solution, as mentioned in the OpenFog RA. With that, we highlight the specificities of Fog Computing regarding orchestration and how our solution overcomes these requirements.

A. An Architecture Prototype

An initial prototype was developed based on Docker Swarm and OpenMTC M2M Framework[18] in the Fraunhofer FOKUS IoT testbed[6]. A custom implementation of the FOA was developed backed by the OpenMTC as the M2M framework. OpenMTC abstracts standardize access to different devices by implementing the OneM2M Architecture. The FOA implementation can start containerized services when detecting new devices attached/connected to the Fog Node. Also, the FOA monitoring subcomponent was implement, and a Docker adapter developed. In the FO side, the Infrastructure Manager handles a Docker Swarm cluster, after the addition of a node, the Infrastructure Manager handles the addition of the node to the Docker Swarm infrastructure. The FOA Monitoring tool was also developed and a basic Orchestrator implemented. As mentioned before, modeling the resources, the nodes and its requirements and capabilities, is necessary. These definitions are used to create the orchestration template. The template informs the orchestrator about resources requirements and dependencies, we used TOSCA Simple YAML 1.0. Resource discovery and all the catalogs are missing, as well the management of services, being the orchestrator only able to start them respecting the given constraints.

To deal with the constraints, the Orchestrator makes extensive use of Docker Labels by labeling images, containers, and nodes with its capabilities. Containers are started only if labels found in the requirements met the labels found in the Fog Node.

[6]http://openiotfog.org/en/

B. The ETSI NFV MANO and Fog Computing

ETSI NFV MANO is a successful Reference Architecture for Infrastructure Management and Orchestration, defined in the context of NFV. Some architectural elements in this proposal shares concepts with the MANO architecture, however, the nature and behavior of the Nodes in the infrastructure lead us to a different approach. The main difference is the role of nodes in the architecture, so we propose the abstraction of the Virtualization/Containerization software running in the Node (Docker, Xen, etc.), not the abstraction of the virtualized infrastructure (e.g. OpenStack). ETSI NFV MANO was conceived to be applied in Clouds, Datacenters, and Clusters which delivers a high level of homogeneity, in which the infrastructure have a management layer, on the other hand, the Fog Infrastructure Manager must be able to setup and manage its infrastructure and its nodes. As an example, let's imagine a catastrophe situation, in which rescue vehicles are equipped with Fog Nodes. In the catastrophe site, the network infrastructure is damaged, potential Fog Nodes in the area, which could offer potential computational power for video processing for example, as well provides access to sensor data in buildings and vehicles, maybe, are available. A Fog Node in a Rescue Vehicle can try to setup an infrastructure by contacting these Nodes and adding them to the infrastructure, after that it starts to orchestrate services. In the case of Fog Computing, the Infrastructure Manager will in some extent deal with the Physical Infrastructure. It must, for example, detect the absence of a Fog Node and reacting to it is a task of the Infrastructure Manager, in such case the Infrastructure Manager is the only one that has knowledge of the existence of a Node. ETSI NFV MANO Virtual Infrastructure Managers (VIM)s, can be implemented to be dynamic, hiding the details of how the infrastructure is managed to the NFVO. What differs basically from our approach to the MANO approach is, MANO abstract virtualized infrastructures, we abstract the Nodes virtualization underlying system. How this impact the adoption of some MANO implementation as a solution for the Fog is under investigation by our team by applying the Open Baton[3] implementation of the ETSI MANO. Another point that needs consideration, the use of the term resource. In the MANO context, it is used in general to describe Computing resources, storage and networks even defining resources as, quote *"Visualized resources in-scope are those that can be associated with visualization containers and have been cataloged and offered for consumption through appropriately abstracted services"*[8]. We understand that investigating the possibility of using ETSI MANO implementations as the base for an implementation of our architecture is necessary. In fact we are studying it, however, how to extend is still an open issue due to the central role of the Fog Node in a Fog Computing solution.

C. Synergy with the OpenFog Reference Architecture for Fog Computing

In the moment of writing this paper, it was released the OpenFog Reference Architecture (OpenFog RA), it is the first step to settle down the concepts and requirements for the Fog. To be aligned with the Fog community in this section we highlight how the concepts of the new RA relates to our proposal.

From the eight pillars of Fog Computing, defined in the OpenFog RA, this architecture is mainly related to the Autonomy, Programmability and the Openness pillars receiving support from the Scalability pillar and dependent on the Security pillar.

In the OpenFog RA, the architecture is defined concerning stakeholders' views and cross-cutting concerns referred as perspectives. Regarding the views, our approach contemplates the Node View, which requires, Protocol Abstraction Layers, Sensors and Actuators control, that is achieved by M2M Framework and implemented in our prototype using the OpenMTC platform. In the OpenFog RA, the System View deal with Hardware Virtualization, in our solution we abstract the hardware virtualization solution, offering a standardized way to access node resources. And last, our Orchestrator and Infrastructure Manager fits in the Software view, which contemplates Node Management, Application Services, and Support. In the OpenFog RA, we are related to the Manageability Perspective while giving support to the Data, Analytics, and Control, and to the Performance and Scale Perspectives. The Security perspective is a requirement of our solution.

V. Conclusion and Future Work

We presented an architecture for Infrastructure Management and Orchestration for Fog Computing. Through the implementation of an initial prototype, we could validate the functional aspects of the proposed architecture. It was possible to orchestrate in the IoT Testbed, services in distributed Fog Nodes, verifying the functional aspects of the architecture. However, evaluation of non-functional aspects of the implementation are missing and are part of the next steps to come. In this tests two Key Performance Indicators is being considered, *i.* Time to Orchestrate, that will measure the start time and how the system behaves when orchestrating different services, in different setups, for example, variating images sizes, image location, *ii.* Opportunity Losses, that will verify if, having all the requirements at a point in time, the orchestration is successful instantiated. It is important to measure due to the dynamics of some Fog Computing scenarios, and *iii.* Time to scale and time to migrate both microservices or the entire service. Also, the implementation of a MANO solution is under work by applying OpenBaton, to validate how ETSI NFV MANO will behave and how it must be extended to support Fog applications. Tests should also be extended to consider 5G aspects by using the Open5GCore solution[11].

VI. Acknowledgments

This work is partly funded by the Universidade Estadual de Santa Cruz and by the Brazilian National Council for Scientific and Technological Development (CNPq) under the contract number 229914/2013-6 and supported by the Fraunhofer FOKUS Institute.

References

[1] Antonio Marcos Alberti and Dhananjay Singh. "Internet of Things: Perspectives, Challenges and Opportunities". In: *Int. WS on Telecommunications (IWT)*. IEEE, 2013, pp. 1–6.

[2] Giuseppe Antonio Carella and Thomas Magedanz. "MEC Enablement by Means of an Open Source ETSI MANO Orchestrator". In: *Newsletter* 2016 (2015).

[3] Giuseppe Antonio Carella and Thomas Magedanz. "Open Baton: A Framework for Virtual Network Function Management and Orchestration for Emerging Software-Based 5G Networks". In: *Newsletter* 2016 (2015).

[4] Margaret Chiosi et al. *Network Functions Virtualisation–Introductory*. White Paper. European Telecommunications Standards Institute (ETSI), 2012.

[5] ETSI Mobile Edge Computing. "Framework and Reference Architecture". In: *ETSI GS MEC* 3 (), p. V1.

[6] Openfog Consortium and Architecture Working. "OpenFog Reference Architecture for Fog Computing". In: February (2017), pp. 1–162.

[7] Thomas Erl. *Service-oriented architecture: concepts, technology, and design*. Pearson Education India, 2005.

[8] ETSI. "GS NFV-MAN 001 - V1.1.1 - Network Functions Virtualisation (NFV); Management and Orchestration". In: *ETSI* 1 (2014), pp. 1–184.

[9] Gartner Inc. *Gartner Says 6.4 Billion Connected 'Things' Will Be in Use in 2016, Up 30 Percent From 2015.*

[10] Hang Liu et al. "Mobile Edge Cloud System: Architectures, Challenges, and Approaches". In: (2017), pp. 1–14.

[11] Thomas Magedanz et al. "Prototyping new concepts beyond 4G–The Fraunhofer Open5GCore". In: *it-Information Technology* 57.5 (2015), pp. 314–320.

[12] Redowan Mahmud and Rajkumar Buyya. "Fog Computing: A Taxonomy, Survey and Future Directions". In: *arXiv preprint arXiv:1611.05539* (2016).

[13] R Mijumbi et al. "Management and Orchestration Challenges in Network Function Virtualization". In: *IEEE Communications Magazine* January (2015), pp. 98–105.

[14] OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0*. Tech. rep. January. 2013.

[15] Marie-Paule Odini. *OpenSource MANO*. 2016.

[16] C. Peltz. "Web services orchestration and choreography". In: *Computer* 36.10 (Oct. 2003), pp. 46–52.

[17] Various. *Network Functions Virtualisation (NFV); Architectural Framework*. Group Specification GS NFV 002. European Telecommunications Standards Institute (ETSI), 2013, pp. 1–21.

[18] Sebastian Wahle, Thomas Magedanz, and Frank Schulze. "The OpenMTC framework - M2M solutions for smart cities and the internet of things". In: *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2012 - Digital Proceedings* (2012), pp. 2–4.