FORCH: An Orchestrator for Fog Computing service deployment

Gianluca Davoli, Davide Borsatti, Daniele Tarchi, Walter Cerroni
University of Bologna, Italy
Email: {gianluca.davoli, davide.borsatti, daniele.tarchi, walter.cerroni}@unibo.it

***Abstract*—In scenarios where resource locality is the key, Fog Computing helps in bringing the potentialities of Everything-as-a-Service (XaaS) closer to the end user, reducing both service time and load on the Cloud infrastructure. We designed and developed FORCH, a service model-aware Fog Computing orchestrator to dynamically allocate services and manage resources available on Fog nodes, in order to provide for different needs of the end users. An experimental test bed to validate FORCH architecture has been implemented and will be the subject of our live demonstration, showing the feasibility of the proposed approach running on different Fog node types and with different service models.**

I. Introduction

The evolution towards 5G and new generation networks in general has highlighted more stringent needs in terms of network and computing performances [1]. Many applications could benefit from the presence of edge devices for remote processing, computation *offloading*, resource relocation, and service deployment with a reduced latency and improved scalability. In this scenario, the Fog Computing paradigm has emerged as a possible intermediate layer between the end user and the Cloud infrastructure [2], aiming at providing services similar to those offered by its larger counterpart (i.e., Cloud Computing) but focusing in particular on the needs of microservices and modular applications [3].

Dynamicity is inherently a fundamental characteristic of the Fog philosophy, due to the heterogeneity of Fog nodes not only in terms of quantity and capacity, but also in terms of offered services. Therefore, a viable approach toward a flexible and dynamic service deployment in a Fog Computing environment is to adopt the "Everything-as-a-Service" (XaaS) model [2]. In order to handle such a variety in terms of resources and services, a suitable Fog Computing orchestration component is needed [4], [5]. Such orchestrator should be able to discover how many and which Fog nodes are available, keep track of the resources and service models they offer, monitor their utilization, listen to service requests from the end users, and decide which node must be allocated to which service request according to which XaaS model. Then it should proceed to instructing both the end user and the infrastructure on said allocation, allowing the end user to access to the allocated resources and obtain the requested service.

This kind of service model-aware orchestration system has not been proposed yet. For instance, the Fog orchestrator presented in [6] is focused specifically on IoT applications and their requirements, and it does not consider multiple service

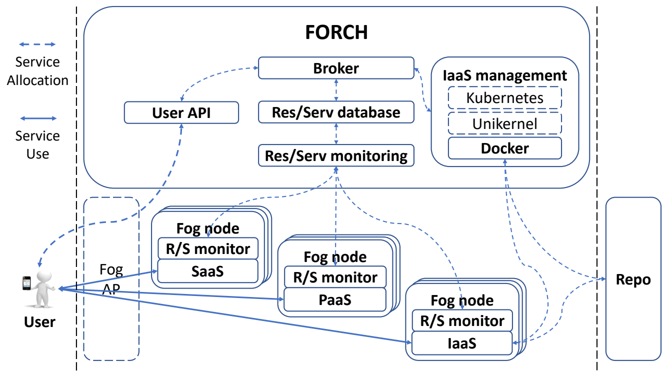

Fig. 1. FORCH reference architecture.

models. Other approaches can be found too, but they are either limited to a single service model [7], or do not include test bed implementations at all [4], [5].

We designed and developed FORCH, a modular orchestration system for Fog Computing infrastructures, which is aware of different service models according to the Software/Platform/Infrastructure-as-a-Service (SaaS/-PaaS/IaaS) classification. FORCH is conceived as a resource and service management layer placed between the end user and the infrastructure itself, composed by a variety of Fog nodes. We devised and implemented a suitable test bed for proof-of-concept (PoC) experimental validation, involving both virtual and physical Fog nodes, which will be the subject of our live demonstration.

II. System Architecture

Figure 1 shows the architecture of the Fog system, highlighting its logical components. The supervising entity is the modular orchestrator we called **FORCH** (Fog ORCHestrator). It coordinates the activities in the Fog system, interacting with the users by providing them information on the available services offered by the Fog infrastructure, and receiving their requests for new service allocations. FORCH also interacts with the Fog nodes to manage services deployed and monitor resources available on them, as well as with repositories in the Cloud, to gather information on the available platforms and software tools. FORCH has multiple components, each developed as an independent module:

- User API (UA): the point of contact for users to interact with FORCH;

- Broker (BR): acts as mediator between users and resource management modules, routing requests and responses, and managing allocation decisions in the process;
- Resource and Service Database (RD): stores information on Fog resources and on their current state, including data on availability, residual capacity and allocated services;
- Resource and Service Monitoring (RM): acts as collector for the monitoring data sent over from the agent modules in the Fog nodes;
- IaaS Management (IM): manages resources allocation and service deployment on IaaS Fog nodes, leveraging multiple lightweight virtualization technologies (e.g., containers, Unikernels) and interacting with related image repositories and management/orchestration platforms (e.g., Docker, Kubernetes, Eclipse fog05).

Users connect to FORCH through a Fog Access Point (AP), and they are only allowed to directly interact with the UA module, through its REST API. All other modules of FORCH expose REST endpoints as well, but those are for internal use only, although they are triggered by user requests.

Fog nodes, regardless of their capabilities, host a module acting as an agent of the RM collector module of FORCH. Through this agent module, the RM can acquire information regarding the services the Fog node can offer along with its residual resources, and monitor its activities once a service is deployed on it. As a particular subset of Fog nodes, IaaS nodes are able to interact directly with the software and platform repository of choice in order to download the software or image needed to provide the service that FORCH wants to deploy on them.

We distinguish three types of entities FORCH can allocate Fog nodes to: Applications (APPs) in the SaaS case, Software Development Platforms (SDPs) in the PaaS case, and Fog Virtualization Engines (FVEs) in the IaaS case. All of them being software entities, they address mutually distinct sets of end user needs with different levels of flexibility. APPs take values as input and return results based on those input values (e.g., a block of a computationally-intensive series of operations), or listen for incoming requests and serve them (e.g., a Web-based application). Diversely, SDPs are meant to accept blocks of code written in a predetermined language and/or using specific development libraries, execute them, and return the output to the end user. (e.g., Remote Java or Python Interpreters). Lastly, FVEs allow a Fog node to host virtualized appliances, enabling the user to deploy its own virtualized system, achieving maximum flexibility.

III. Demonstrator Description

The FORCH live demonstration runs on a test bed that is logically coherent with the architecture depicted in Figure 1. A Virtual Machine (VM) with 2 cores and 4 GB of RAM hosts all FORCH software components, which were developed as separate Python programs meant to be run independently and communicating with each other via REST APIs. A number of Fog nodes are deployed on different hardware platforms. Two nodes are emulated by two separate VMs, each with 1 core and 2 GB of RAM (VM1 and VM2). Two additional nodes are implemented by an Intel NUC MiniPC equipped with a 4-core 8th-gen Intel i7 processor and 16 GB of RAM (NUC1), and a RaspberryPi Single Board Computer, model 3B+, equipped with a 4-core ARMv8 processor and 1 GB of RAM (RP1), respectively. We use Docker as our FVE of choice and Zabbix as the monitoring system used between RM and Fog nodes.

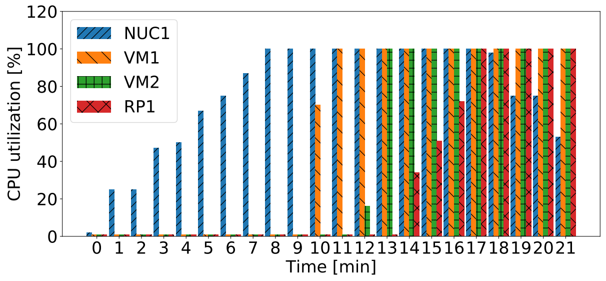

Fig. 2. Example of service allocation based on Fog node CPU utilization.

The live demonstration will showcase how FORCH is able to deploy services on different Fog nodes using different XaaS service models, based on resources and capabilities available on the underlying infrastructure. The different phases of Fog node discovery, resource monitoring, and SaaS/PaaS/IaaS service deployment will be presented. Figure 2 reports how the system behaves in the situation where a sequence of homogeneous requests and allocations increasingly saturates the computation resources of the available Fog nodes. Instances of the requested service are allocated to the NUC1 node as long as its CPUs are not fully loaded (at $t = 8$ min in the example of Fig. 2), then the other nodes are selected for the service requests that follow, until the whole infrastructure is fully utilized (at $t = 17$ min in the example of Fig. 2). In case no additional service request is received, the nodes start to free the resources when each service is completed (from $t = 18$ min to $t = 21$ min in the example of Fig. 2). We remark that this example is intended as a basic PoC, as FORCH is predisposed to handle heterogeneous service requests, monitoring a variety of usage metrics.

References

[1] Y. Ku *et al.*, “5G radio access network design with the fog paradigm: Confluence of communications and computing,” *IEEE Commun. Mag.*, vol. 55, no. 4, pp. 46–52, Apr. 2017.

[2] M. Iorga *et al.*, “Fog computing conceptual model,” The National Institute of Standards and Technology, SP 500-325, Mar. 2018. [Online]. Available: https://doi.org/10.6028/NIST.SP.500-325

[3] C. Mouradian *et al.*, “A comprehensive survey on fog computing: State-of-the-art and research challenges,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, First Quarter 2018.

[4] Z. Wen *et al.*, “Fog orchestration for Internet of Things services,” *IEEE Internet Comput.*, vol. 21, no. 2, pp. 16–24, Mar. 2017.

[5] Y. Jiang *et al.*, “Challenges and solutions in fog computing orchestration,” *IEEE Netw.*, vol. 32, no. 3, pp. 122–129, May 2018.

[6] B. Donassolo *et al.*, “Fog based framework for IoT service provisioning,” in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–6.

[7] S. Tuli *et al.*, “FogBus: A blockchain-based lightweight framework for edge and fog computing,” *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.