The final publication is available at ACM via http://dx.doi.org/10.1145/3213299.3213307

mF2C: Towards a Coordinated Management of the IoT-fog-cloud Continuum *

Xavi Masip-Bruin
Universitat Politecnica de Catalunya (UPC)
08800 Vilanova i la Geltrú
Spain
xmasip@ac.upc.edu

Eva Marín-Tordera
Universitat Politecnica de Catalunya (UPC)
08800 Vilanova i la Geltrú
Spain
eva@ac.upc.edu

Ana Juan-Ferrer
Atos Research and Innovation
08020 Barcelona
Spain
ana.juanf@atos.net

Anna Queralt
Barcelona Supercomputing Center (BSC)
08034 Barcelona
Spain
anna.queralt@bsc.es

Admela Jukan
Technical University of Braunschweig
38106 Braunschweig
Germany
a.jukan@tu-bs.de

Jordi Garcia
Universitat Politecnica de Catalunya (UPC)
08800 Vilanova i la Geltrú
Spain
jordig@ac.upc.edu

Daniele Lezzi
Barcelona Supercomputing Center (BSC)
08034 Barcelona
Spain
daniele.lezzi@bsc.es

Jens Jensen
STFC
Didcot OX11 0QX
United Kingdom
jens.jensen@stfc.ac.uk

Cristovao Cordeiro
SixSq
1217 Geneve
Switzerland
cristovao.cordeiro@sixsq.com

Alexander Leckey
Intel R&D Ireland
R148, Easton, Co. Kildare
Ireland
alexander.j.leckey@intel.com

Antonio Salis
Engineering Sardegna Srl
09123 Cagliari
Italy
antonio.salis@eng.it

Denis Guilhot
Worldsensing
08014 Barcelona
Spain
dguilhot@worldsensing.com

Matic Cankar
XLAB d.o.o.
Pot za Brdom 100, 1000 Ljubljana
Slovenia
matija.cankar@xlab.si

ABSTRACT

Fog computing enables location dependent resource allocation and low latency services, while fostering novel market and business opportunities in the cloud sector. Aligned to this trend, we refer to Fog-to-cloud (F2C) computing system as a new pool of resources, set into a layered and hierarchical model, intended to ease the entire fog and cloud resources management and coordination. The H2020 project mF2C aims at designing, developing and testing a first attempt for a real F2C architecture. This document outlines the architecture and main functionalities of the management framework designed in the mF2C project to coordinate the execution of services in the envisioned set of heterogeneous and distributed resources.

CCS CONCEPTS

• **Computer systems organization** → **Architectures**; Distributed architectures • **Networks** → Network architectures

KEYWORDS

Cloud computing, Fog computing, IoT

ACM Reference format:

X.Masip-Bruin, E.Marín-Tordera, A.Juan-Ferrer, A.Queralt, A.Jukan, J.Garcia,

D.Lezzi, J.Jensen, C.Cordeiro, A.Leckey, A.Salis, D.Guilhot, M.Cankar. 2018. SIG Proceedings Paper in word Format. In *Proceedings of SmartObjects-MobiHoc, Los Angeles, USA, June 2018*

1 INTRODUCTION

The emergence of IoT has led to a rapidly increasing number of connected devices worldwide, from billions of units we have today, to tens of billions of units expected to be deployed in the coming years. Some predictions (see for example [1]) point out that 26 billion edge devices are to be connected by 2020, collecting more than 1.6 zettabytes (1.6 trillion GB) of data. According to Cisco reports, it is expected to have more than 20 billion devices connected by 2020 [2]). There is no doubt that the cloud computing paradigm provides a proper solution to support the management of the processing and storage needs brought by the set of existing and yet unforeseen services in the IoT world. However, it is also widely accepted that many of these services have specific requirements not fully aligned to the cloud characteristics. For example, it is pretty obvious that long distances inherent to the cloud model do not suit real time services which typically need low runtime latency to operate. To address such limitations and also to leverage the ever increasing capacities of edge devices, fog computing (also referred to as edge computing) was recently proposed as an alternative. The main rationale behind fog computing is in bringing cloud resources close to the edge, i.e., the location when services execution is required and where data is generated. Notable benefits brought by fog computing are low latency, reduced network traffic, low energy consumption and often higher offered security. Nevertheless, the best is yet to come. Indeed, fog computing highly complements cloud computing, in a new scenario where services execution may benefit from both paradigms with no need to sacrifice either.

To that end, fog and cloud computing when combined require a novel coordinated management strategy, intended to properly manage the whole set of resources in a harmonious fashion, while also empowering new policies, such as those based on sharing or collaborative models. Several current efforts aim at this management strategy. First and foremost is the OpenFog Consortium that recently issued its first release for the so-called OpenFog Reference Architecture (OFRA) [3]), as a high level definition of the main steps to build its architecture. A parallel effort working on a similar direction is led by the EU H2020 mF2C project [4], aimed at developing the Fog-to-Cloud concept (F2C) proposed in [5]. In this paper we focus on main findings in the mF2C architecture design, particularly emphasizing its main functional modules.

The main objective of the mF2C project is to design and develop a hierarchical, open, secure, decentralized and coordinated management platform facilitating the efficient usage of F2C resources, taking into consideration service requirements and user demands, in a scenario combining cloud and fog computing. The F2C coordinated computing ecosystem has been developed to: i) efficiently and transparently utilize available distributed and heterogeneous resources at the edge; ii) support applications and services that do not fit well into the paradigm of the traditional centralized cloud, and; iii) pave the way to new business models in both cloud and smart devices sectors. Last but not least, security and privacy are also addressed with built-in (by design) capabilities in a complementary fashion in the mF2C project.

Fig. 1 shows a functional architecture of the mF2C ecosystem that integrates a centralized cloud infrastructure, with various levels (refereed to as layers) of dispersed elements, all managed by mF2C agents, and with various degrees of decision making and data processing capabilities (the stack of resources). In this combined scenario, users will see an optimized service performance when the service can decide on-the-fly the best suited set of fog/cloud resources, enabling enriched service execution features to upscale performance, such as parallel tasks execution and computational offloading to the cloud.

In this paper, we first outline the main functionalities of the mF2C management framework being developed in the mF2C project. To that end, section 2 digs into the basic principles of the proposed mF2C architecture. Then, Section 3 describes

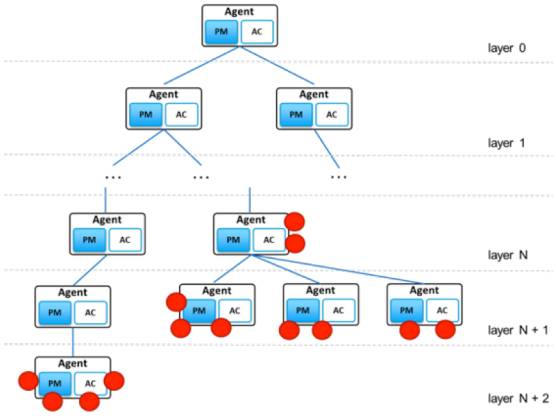

Figure 1: mF2C Architecture.

services execution in an mF2C system and Section 4 pockets into the security issues in mF2C. Section 5 describes the use cases proposed in the project as candidate markets for mF2C and finally, conclusions and future directions are included in Section 6.

2 mF2C: THE BASIC PRINCIPLES

This section describes the main mF2C architectural blocks, defining the key elements building the whole mF2C system.

2.1 mF2C Approach: Agents & Leaders

In order to manage the huge set of heterogeneous devices, we propose to organize them all in a hierarchical architecture, as shown in Fig.1, where resources are grouped into layers, and an mF2C agent entity deploys the management functionalities in every component within the system. We see different layers (from layer 0 at cloud to Layer N+2 at the level closer to the edge) and the agent software installed in all devices capable of supporting it, participating in the mF2C system. Information from those devices incapable of hosting an agent, such as sensors and actuators (red balls in the figure) is gathered, processed and distributed by the agent connecting them to the system. Devices are clustered under the control of one device that is defined as the leader. The clustering strategy and leadership election policy is yet to be defined, although characteristics such as distance and connectivity may be considered in a first approach. Additional assumptions to the envisioned mF2C architecture are:

- Fog area or cluster stands for the set of nodes managed by a leader.
- Only one node acts as leader in each fog area.
- Only one backup node (which becomes the leader when the leader fails), in each fog area.
- IoT devices can be connected to any of the agents in the mF2C system.

The whole set of management and control functionalities within the agent is divided into two main blocks, the Platform Manager (PM), and the Agent Controller (AC). In short, the PM provides high-level functionalities, responsible for inter-agent communications (agents communicate through their PMs) and thus, with the capacity to take decisions with a more global view. On the other hand, Agent Controller (AC) has a more local scope, dealing with local resources and services. From an execution point of view, when a service/task is requested to any of the mF2C agents, the responsibility of deciding if this task can be executed in that agent, or forwarded down (to any of the agents in the area if the agent is a leader) or up (to the higher hierarchical layer) is taken by the PM. If the task is forwarded, the communication is also done through the PMs of the agents. The request is passed to the AC only when an agent can execute the forwarded task, using the agent's local (own) resources.

Regarding the mF2C data management, a distributed approach is considered, assuming:

- An agent contains information about itself and its connected IoT devices
- A leader contains information about itself, its connected IoT devices, and the nodes ("children") within its fog area (maybe aggregated).
- The cloud agent will manage information (possibly aggregated) about all devices in the mF2C system.

2.2 mF2C Agent: Platform Manager and Agent Controller

In this section we detail the mF2C agent functionalities, divided between the Platform Manager (PM) and the Agent Controller (AC).

2.2.1 The Platform Manager (PM).

Fig. 2 shows the envisioned PM blocks, split into three main components, Service Orchestration, Distributed Execution Runtime and Telemetry.

2.2.1.1 Service Orchestration: Responsible for allocating the services to the most suitable resources, is composed by:

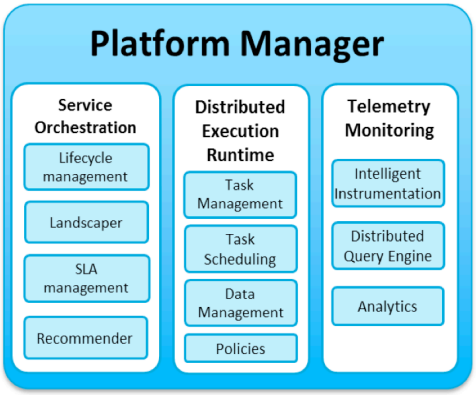

Figure 2: Platform Manager (PM) blocks

- **Lifecycle management**: Responsible for managing the lifecycle of the applications to be executed.
- **Landscaper**: Intended to obtain a view of the whole mF2C infrastructure, including all the physical machines and parts of, e.g., CPU, storage, memory, etc.
- **SLA management**: Responsible for managing the SLAs between the parties collaborating in a service on the mF2C platform.
- **Recommender**: Feeds the Lifecycle with an appropriate recipe of suitable type of resources for a service.

2.2.1.2 Distributed Execution Runtime (DER): Responsible for optimizing services/tasks execution on the available resources, is composed of::

- **Task management**: Its main purpose is to orchestrate the execution of tasks, to optimally exploit the available computing resources.
- **Task Scheduling**: Responsible for distributing the tasks generated by the execution of the applications on the resources selected by the Lifecycle Manager.
- **Policies**: Needed to support the Runtime in the selection of the resources for the tasks scheduling.
- **Data management**: Responsible for storing the metadata of the objects..

Components related to Policies, Task Management and Scheduling are handled by COMPSs [6], while Data Management uses dataClay [7].

2.2.1.3. Telemetry and Monitoring: Responsible for analysing the service performance on the infrastructure it is deployed on. The three main components are:

- **Intelligent Instrumentation**: Responsible for providing the telemetry collectors and aggregators of the metrics, measuring performance of key physical components.
- **Distributed Query Engine**: Provides a single API to facilitate the querying of all telemetry data captured.
- **The Analytics module**: Characterises service execution by mapping the service's deployment configuration against telemetry captured for those same nodes.

2.2.2 The Agent Controller (AC).

The set of AC functionalities is split into three main blocks, Resource, Service, and User Management (see Fig. 3).

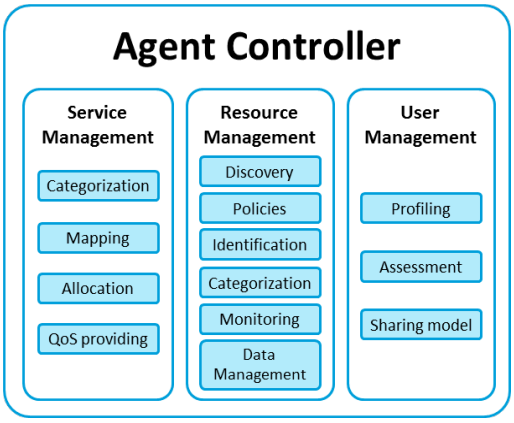

Figure 3: Agent Controller (AC) blocks.

2.2.2.1. Resource management: Responsible for collecting and managing local resources. In the case of a leader device, its 'local view' includes its own resources but also those of the devices forming part of its

cluster. The six components of the resources management block are:

- **Discovery.** Responsible for discovering resources in a fog area managed by a leader.
- **Policies.** Rules to be used by the AC (clustering, leader and backup selection, protection, resource aggregation, etc.).
- **Identification.** Responsible for both providing every device with a globally unique ID, and establishing a mechanism to update and/or revoke the ID.
- **Categorization.** Provides common information about the resources, i.e., hardware, power, software, security, attached components, attached IoT information, and also information about its behaviour.
- **Monitoring.** Responsible for instrumentation of each compute resource. A number of telemetry probes will capture performance metrics of the hardware/software that services are deployed onto.
- **Data management.** Responsible for allowing applications or other functionalities to store, retrieve, and delete data in mF2C.

2.2.2.2 Service Management: Responsible for the orchestration of local services it has the following functionalities.

- **Categorization:** It receives a service request, and categorizes this request according to some defined attributes (CPU, Storage, Network, Memory, Priority, Time limit and Location in a first approach).
- **Mapping:** Responsible for selecting the resources best matching the demanded task requirements in the own resources of the agent.
- **Allocation:** Responsible for the optimal allocation of available resources in the agent to the various tasks requests.
- **QoS provisioning.** Based on previous executions' performance metrics, this block will inform the Lifecycle to discard unsuitable candidates.

2.2.2.3 User Management: Responsible for managing the profiling and the sharing model properties of users. This module is composed by three components, as follows:

- **Profiling:** A user profile is the collection of personal data related with a specific user.
- **Assessment:** Responsible for checking if the mF2C apps meet the sharing model and the profile properties defined by the device's user.
- **Sharing model.** Defines resources that the device's owner wants to share with the mF2C system.

2.3 mF2C Agent: System databases

The database is unique, and both the PM and the AC share the database. Due to the hierarchical mF2C architecture and the shared database, a key AC functionality is to fill in the database to be used by both PM and AC with information about:

- Own resources if the device is part of the cluster but not a leader.
- Own resources and resources of the devices in the cluster if the device is the leader of the cluster.

As can be seen in Fig. 4, the database in each agent will contain its local information, which is periodically copied or summarized (according to a certain policy) in the same device. In turn, this aggregated data, denoted as *AGGR* in Fig. 4, is also periodically synchronized with the local data of its leader.

Finally, in the 3 hierarchical layers proposed as a first approach to the mF2C architecture, the leader will also aggregate its local data (about its own resources and resources of its children devices), and this AGGR information will be periodically (or by another policy) synchronized with the cloud's leader database.

In Fig. 4 we can see that each agent holds local data and aggregated data (AGGR), synchronized with the higher layer database. In order to simplify the example we have considered only three parameters: CPU, Storage and IoT (includes information about sensors, actuators, etc., attached to the agent).

Figure 4: Aggregation example.

3 SERVICE EXECUTION IN mF2C SYSTEMS

In the envisioned mF2C distributed scenario, a service request may be launched by any agent in the system. The execution process, as it is described in Fig. 5, runs as follows:

- The service is always requested to the PM, reaching out the Lifecycle.
- If the PM of the agent can solve the service request with its own resources (leader and children), it starts the process of resources deployment and services execution.
- Otherwise the request is forwarded to the PM of the leader in the upper layer of the hierarchy.
- The leader's PM will check if the service can be executed in its own resources, otherwise the request will be forwarded to the PM of the leader in the upper hierarchical layer.
- Deployment and execution of a service

Let us assume a service is launched in Agent X in Fig. 5. After checking its own resources, the PM in Agent X forwards the request to its leader in the upper layer. We also assume that the leader or/and some other agents in the cluster have the requisite resources to execute the service, with no need to forward the request further to an upper layer in the hierarchy.

The service request reaches out to the Lifecycle that queries the Recommender for a recipe (comprising resources best suiting the service demands). The Lifecycle matches this recipe with a snapshot showing the actual resources availability obtained from the Landscaper, the information from the QoS Providing and the Profiling to select the

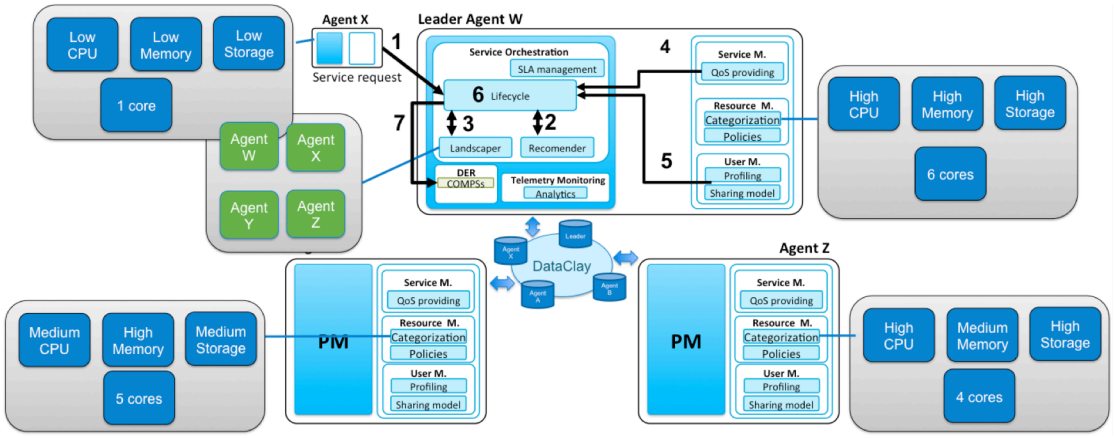

Figure 5: Service execution in mF2C systems

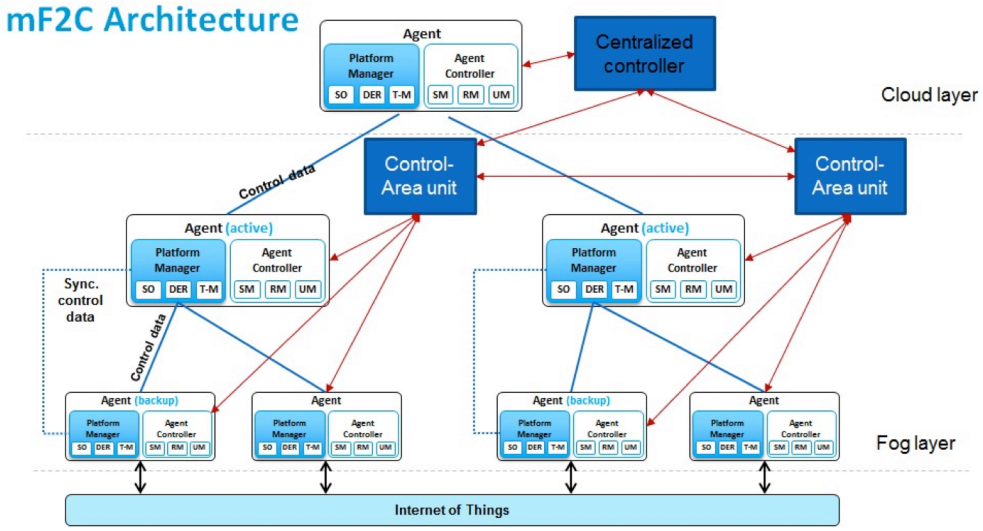

Figure 6: Security in mF2C systems

specific set of resources the service must be deployed at. The DER is then called to allocate and execute the service.

4 SECURITY IN mF2C SYSTEMS

The mF2C usage scenario combines cloud, fog and IoT devices at the edge rendering security an arduous challenge. There is no doubt that experience from each individual area may assist the development of an appropriate solution for mF2C, we need a single solution that works from f2C. To this end, efforts have been split into two directions. First, defines a comprehensive list of security requirements. Second, design a security architecture addressing these specific demands for mF2C systems.

To that end, [8] proposes a security architecture for the first time, leveraging the concept of decoupling security from other functionalities. The main rationale behind this decoupling concept boils down to considering security as a transversal service. This approach abstracts away the need to provide security from each block in the mF2C architecture (see Fig. 6). A "control-area unit" (CAU), which is factored out of the agent, provides security services locally to all nearby agents.

The strategy uses a centralized controller in cloud (layer 0 in mF2C) and distributed

CAUs to provide the security requirements in distributed fogs (layer 1 to layer n in mF2C). The distributed CAUs, in the registration and initialization phases, get their authentication and authorization artifacts from a centralized controller at cloud to provide security to their corresponding fogs.

Notice that this should not be seen as a modification of the architecture, but rather a complementary service unit running alongside the agents within layer 1. In simple code terms, the question is whether security functionality is provided by a library linked into the agent, or by a nearby web service (or similar).

5 USE CASES

The mF2C project proposes three real-world use cases for validation purposes.

5.1 Emergency Situation Management in Smart Cities (ESM)

This use case proposes to use mF2C to handle emergency situations in smart cities, leveraging the inherent mF2C characteristics to evaluate the obtained benefits in terms of service performance. Basically, while the fog layer provides a rapid response to the emergency, the connection with the cloud allows optimizing the resources to be used based on the historical knowledge of similar situations (applying predictive models, etc.). The use case is deployed at the UPC testbed located at the CRAAX lab that emulates a smart city in a 25m^2 area. The

envisioned service for validation purposes will consist in: i) detecting the collapse of a city construction, and ii) triggering a set of actions in the city to mitigate the effects produced by the accident.

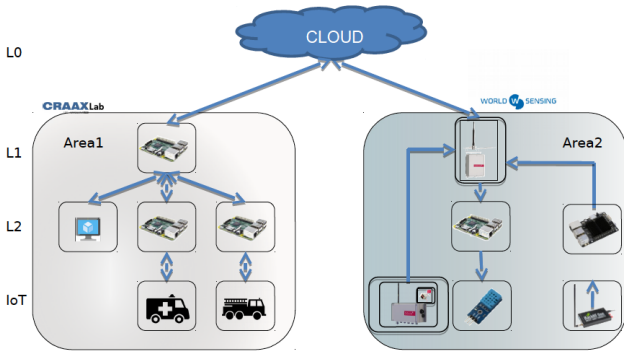

Figure 7: Emergency management (ESM).

The proposed scenario is split into two areas, as shown in Fig. 7. The area on the left includes the different components (ambulance, traffic control systems, fire truck, traffic lights, street lights) deployed in reaction to the accident and the one on the right includes the sensors (inclinometer connected through LoRa, a jammer detector and a temperature sensor) deployed to detect the accident. The two areas deploy agents in the Fog layer and are interconnected through an agent in Cloud.

In the proposed scenario, the inclinometer is periodically monitored by an agent in the building, responsible for triggering a request for accident mitigation (supported by IoT devices previously associated with the service) reacting to a sensor warning. This request is first handled by the PM of the same building agent to check if it has the necessary resources to handle the incident. Otherwise, the request is sent to the PM of the leader, which checks if any of the agents in its cluster (area) have the requested IoT resources. Otherwise, the request is escalated to the Cloud layer, which has a complete view of the entire system. The cloud leader will find which leader has agents with the requested IoT resources, and forwards the request for resources to this leader. The corresponding leader checks that the resources are available in the agents and sends a resource assignment request. The agents involved in the care of the service establish direct communication with the agent that activates the emergency and the action order is given (for example, the traffic lights turn green/red to facilitate the access of the fire truck that, together with the ambulance, etc.).

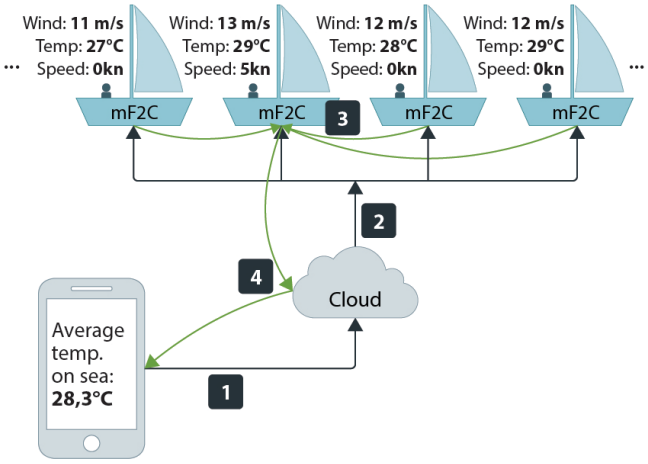

Figure 8: Smart Boat Services (SBS).

5.2 Smart Boat Services (SBS)

This use case focuses on intermittent communication availability in Fog and IoT environments. The vessels (such as yachts or boats of different sizes) generate large amounts of data, useful for either safer boating or potential business cases, ranging from insurance to social media. Similarly to the previous one, this use case also considers the deployment of some sensing devices, in this case Sentinel, an IoT device consisting of different sensors currently applied in the navigation sector for vessel monitoring. The mF2C solution can help implement novel services and sustain a part of the Smart Boat device functionality when the boat is outside of 3G/4G network coverage as well, by facilitating boat connectivity using alternative technologies. Core services that benefit from this are continuous monitoring for fleet management, anomaly detection, offline and anonymous anchorage payment and data plan sharing based on fair exchange policies.

An example of the proposed service, as shown in Fig. 8, assumes an agent located on land or at sea, wants to know the average temperature on a particular area at sea. To that end, the deployed system performs as follows:

- The service request is sent to the Cloud agent.
- The PM of the Cloud agent selects the requisite resources and allocates tasks to resources in different ships.
- A selected device aggregates the data and sends it to the Cloud agent.
- The Cloud agent replies with the average temperature at sea.

One of the main characteristics of the proposed services is the use of WiFi or LoRa for maintaining connections between ships, and the use of 3G/4G for connecting to the agent in the cloud. The same functionality will be available whether there is connectivity to the cloud or not. In this particular scenario, the leading PM would include only the nearby agents, visible through alternative connections.

5.3 Smart Fog-Hub Service (SFHS)

The main rationale behind the third use case is setting up hubs in public environments (e.g. airports, train stations, hospitals, malls and related parking areas), capable of tracking the presence of people and other objects in the field, and developing added value services on top for proximity marketing, prediction of path/behaviour of consumers, and making real time decisions based on prediction of path/behaviour of consumers.

Let us consider an airport as a small city where a large number of people must spend a long period of time wandering through it while waiting for their flights. In this space, many services are offered to users such as shops, restaurants, relaxation areas, etc., which are distributed throughout the airport. Users can while away their waiting time by exploiting these services but they must always be attentive to the time of boarding their flight and to the location of the boarding gate, which may change. The uncertainty caused by not knowing the time a user needs to get to the boarding gate means that, in large airports, the use of these services is limited to their physical proximity to the boarding gate and the amount of spare time before boarding. This means that most users do not move out of a radius near the boarding gate during the last hour of boarding their flight. Thus, many of the services offered at airports depend on the area where the boarding gate is located and, therefore, are not used by users from other areas.

The main features of the developed services will be: i) tracking people and objects; ii) developing added value services for proximity marketing; iii) recommending best use of airport services, and; iv) predicting path/behaviour of consumers

Fig. 9 shows the architecture overview of the proposed Smart Fog-Hub Service, including: i) Layer 0 at cloud; ii) Layer 1 consisting in Fog leaders; iii) Layer 2 bringing together the lower devices installing the mF2C agent (workers), and; iv) layer 3 including edge devices with no mF2C agent running.

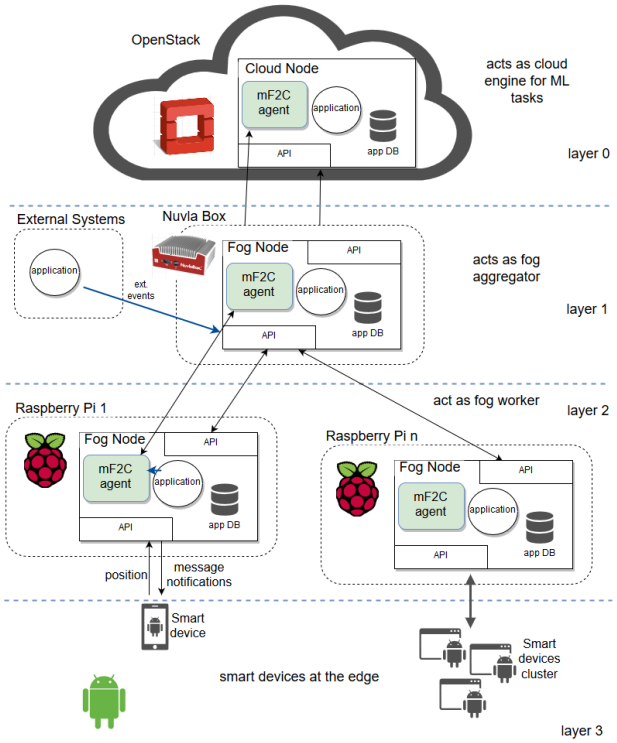

Figure 9: Smart Fog-Hub Service (SFHS).

6 RELATED FRAMEWORKS

The mF2C project has already produced several papers to introducing its main concepts (see [9] or [10]). Considering that the mF2C project proposes a strategy to optimally map services into resources which span the full stack from cloud to the edge,

we find a key initiative working with similar objectives, the OpenFog consortium.

The OpenFog Consortium is in fact working on providing a complete solution to manage fog and cloud resources, using a high level architecture (referred to as the OpenFog Reference Architecture (OFRA) [3]). Although both initiatives share similar objectives, it is important to note a critical difference, the Open Fog Consortium architecture is based on the concept of Fog Node. The scalable pervasive computing architecture in OFRA is built based on fog nodes, which are specific hardware devices (legacy brown-field devices), setting the communication and computing entities that support hardware virtualization and trusted computing on one hand while perform secure communication and service provisioning on the other. Instead, the agents in mF2C are devices with enough capacity to support the mF2C agent software.

Other important differences pertain to the strategy to manage security. For example, mF2C considers data privacy, which is not considered in OFRA. In OFRA the security boundaries are open whereas in mF2C an underlying uniform architecture is proposed. OFRA ranks security threats according to the severity of impact while mF2C threats ranks would be used by people monitoring the system; OFRA builds on Hardware Trusted Platform Models, an expensive solution requiring hardware Trusted Computing and notable sysadmin support.

Finally, it is also worth mentioning other related initiatives, such as the ETSI Multi-Access Edge Computing (MEC) [11], and the OpenEdge Computing organization [12].

7 CONCLUSIONS

There unstoppable deployment of devices at the edge is bringing new challenges requiring much attention by the scientific and industrial communities. As the devices are becoming ever smarter, the concepts that take advantage of such "smartness", will lead to a wider adoption fog computing. However, fog and cloud play similar and complementary roles and thus some coordination among them would help optimize service execution. The mF2C project aims at providing such coordination by creating an innovative management architecture, deployed through software instantiations, hence with no need for specific hardware deployments.

In this paper, we highlighted the motivation for resources coordination akin to mF2C, and illustrated the main components of the envisioned mF2C architecture, with particular detail on the set of functional blocks in the mF2C system. Finally, and most interestingly, we also showed three different use cases that will be deployed to validate the mF2C development.

ACKNOWLEDGMENTS

This work was supported by the H2020 mF2C project (730929). For UPC authors is also partially supported by the Spanish Ministry of Economy and Competitiveness and by the European Regional Development Fund under contract TEC2015-66220-R (MINECO/FEDER), and for BSC authors, by the Spanish Ministry of Science and Innovation under contract TIN2015-65316, and by Generalitat de Catalunya under contract 2014-SGR-1051.

REFERENCES

[1] D.C. Plummer. 2016. Top Strategic Predictions for 2016 and Beyond: The Future is a Digital Think, https://www.gartner.com/binaries/content/assets/events/keywords/symposium/sym26/gartner_top_strategic_predictions_2016.pdf [Accessed: Feb. 2018].

[2] Dave Evans. 2011. The Internet of Things. Cisco White paper at https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf [Accessed: Feb. 2018].

[3] OpenFog Consortium Working Group. 2017. OpenFog Reference Architecture for Fog Computing. Feb. 2017.

[4] mF2C project at http://www.mf2c-project.eu. [Accessed March 2018].

[5] Xavi Masip-Bruin, et al. 2016. Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud (F2C) computing systems. *Wireless Communication Magazine*, Vol. 23, Issue 5, Oct. 2016.

[6] COMPS at https://www.bsc.es/research-and-development/software-and-apps/software-list/comp-superscalar , [Accessed March 2018].

[7] Toni Cortes, et al. 2015. DataClay: Towards Usable and Shareable Storate. Big Data and Extreme-Scale Computing (BDEC), 2015

[8] Sarang Kahvazadeh, et al. 2017. Securing combined Fog-to-Cloud System through SDN approach , *4th Workshop on CrossCloud Infrastructures & Platforms (ACM Digital Library)*, Serbia, Belgrade, April 2017.

[9] Xavi Masip-Bruin, et al. 2018. Managing Resources Continuity from the Edge to the Cloud: Architecture and Performance. *Future Generation Computer Systems*, Vol. 37, February 2018.

[10] Wilson Ramírez, et al. 2017. Evaluating the Benefits of Combined and Continuous Fog-to-Cloud Architectures. *Computer Communications*, Vol.113, pp.43-52, November 2017

[11] ETSI, Multi-access Edge Computing (MEC) http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing, [Accessed: Feb. 2018].

[12] OpenEdge Computing at http://openedgecomputing.org, [Accessed March 2018].