http://www.mf2c-project.eu/

The mF2C Project: Challenges & Lessons Learnt

Xavi Masip, Eva Marín, Ana Juan, Anna Queralt, Admela Jukan, Cristovao Cordeiro, Jens Jensen, Michael McGrath, Antonio Salis, Denis Guilhot, Matija Cankar

November 1, 2019

Abstract

Fog, edge, cloud, IoT and big data are individual concepts that when treated in an integrated and holistic manner provide a rich technology fabric which be used to support the development of innovative services in distinct verticals. However, realisation of this scenario imposes significant challenges, requiring innovative approaches in order to catalyse the expected evolution towards Next Generation IoT.

One of the fundamental technology pillars necessary to address in this scenario is the appropriate management of heterogeneous infrastructures available to host and execute services. Addressing this challenge is the key focus for mF2C. The projects' main goal is to define an innovative management solution which can select the optimal set of available resources to run a service regardless of where these resources are located.

This paper presents the technological advancements in the second iteration of the mF2C project, emphasizing the key challenges addressed by the project, the lessons learnt which may help similar initiatives to accelerate their progress, and the KPIs used to quantify and qualify the benefits of deploying the mF2C framework based on three real-world use case pilots.

Table of Contents

Scenario Overview

There is a significant body work in the form of scientific papers, white papers, blogs, etc., dealing with issues related to the high profile technological of IoT, cloud, fog, edge, etc. These publications all highlight the potential benefits, weaknesses and possible usage scenarios for these layers of computing. Certainly, such a large record of contributions does not randomly emerge. Instead it represents a tangible response to the current trend of designing new computing models focused on accommodating large data volumes, new resources deployment approaches, increasing infrastructure heterogeneity, service demands as well as enhanced end user experiences. In fact, any new computing scenario to be considered, must meet emerging demands for ultra-low latency (ULL) to support real time services managing increasing large volumes of data collected at the edge of the cyber and physical worlds, e.g. IoT sensors, and providing users with the expected QoS according to an agreed SLA. Although new technologies must substantially contribute to making that objective to becoming a reality, such as the deployment of 5G to minimize latency at the edge, some issues relating to resource and service management require further research. Indeed, the advent of 5G exacerbates this need and as a consequence issues related to the management of heterogeneous and usually highly mobile devices at scale have never been considered until now.

Aligned to this need, several efforts are currently active in the design and development of novel management strategies focused on appropriately managing the collective set of available resources in the so-called resource continuum [1], cloud to thing continuum [2] or fog-to-cloud [3] among others. Indeed, such a coordinated management framework is mandatory to benefit from the entire set of available resources, i.e., maximizing resource utilization while simultaneously optimizing services execution. This assessment is fundamental for both providers and users in order to make the most out of the available resources (impacting CAPEX). It is also necessary in order to attract users to run innovative and resource intensive services.

However, designing a management framework capable of addressing this need is a significant challenge. This rationale is mainly driven by the particular characteristics inherent to systems close to the edge, i.e., mobility, volatility or low capacity, imposing hard constraints and limitations in device management. For example, the dynamics associated with energy saving policies for edge devices in order to increase battery life (resulting in devices on/off power cycles), makes it difficult to compose a stable real-time graph of available resources.

In this context, the H2020 mF2C project is focused on designing and developing a hierarchical and decentralized management architecture for ta fog-to-cloud scenario which is intended to be secure, robust, scalable and efficient. The project has followed an iterative approach setting two main iterations (IT-1 and IT-2) each lasting 18 months and each including their own design, deployment and validation phases. IT-2 builds on top of IT-1, so while IT-1 started its design phase from scratch, IT-2 design efforts have been allocated to generalize and optimize the preliminary design in IT-1 according to the preliminary results obtained in the IT-1 validation phase.

mF2C Architectural Concepts

Main Concepts

The mF2C system proposes a coordinated management solution for all resources, from the edge to the cloud, with a view of optimizing services execution. The proposed solution leverages a hierarchical and decentralized architecture whose main component is referred to as the **Agent**. The Agent is deployed on different elements participating in the overall mF2C system, which includes the complete set of functionalities considered within the management framework. In short, the mF2C management solution is a software suite to be deployed on the different devices willing to participate in mF2C. From an architectural perspective, the set of mF2C devices are distributed to different layers, setting a hierarchical management architecture where several nodes acting as leaders (or cluster heads), are responsible for managing the nodes behind in a scalable way. Figure 1 shows the envisioned layered architecture for mF2C, illustrating the different agents in a hierarchical structure and potential IoT devices directly attached to the agents.

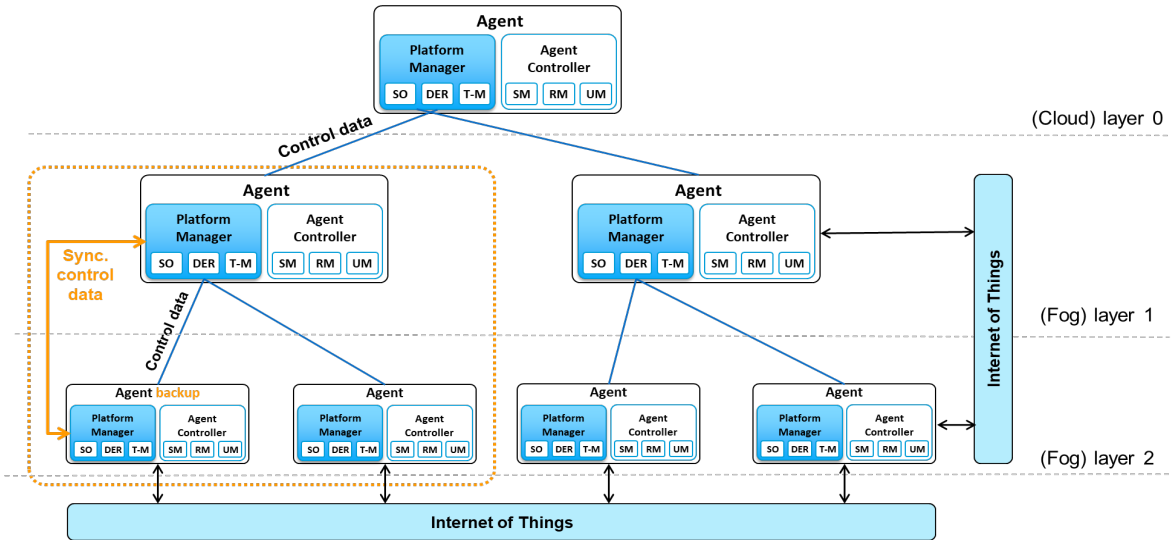

Figure 1 mF2C layered architecture

From a functional perspective the Agent instantiates the set of functionalities the mF2C management framework is expected to provide. Functionalities are split

into two main functional components, the **Agent Controller (AC)** and the **Platform Manager (PM)**, as shown in Figure 2, The PM provides the high-level functionalities, responsible for inter-agent communications (agents communicate through their PMs) and has the capacity to take decisions with a more global view. On the other hand, the Agent Controller (AC) functionalities have a more local scope, dealing with local resources and services.

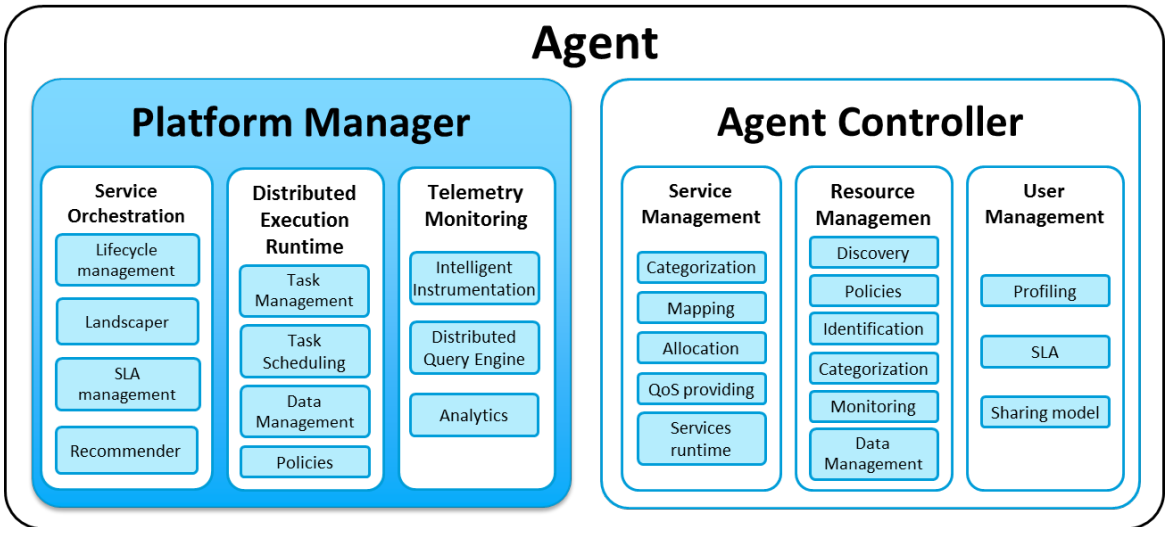

Figure 2 Functional components of mF2C Agent

As shown in Figure 2, the AC and the PM include a number of internal components to support both interoperability and to provide global functionality. More detailed information on the internal components may be found in the different publications and deliverables on the project's website (https://www.mf2c-project.eu).

Architectural Iterations

As previously mentioned the mF2C project has conducted an iterative approach with two iterations, with the second one a notable refinement of the initial one, addressing aspects related to the deployment and demonstration through three real-world pilots. The second iteration also improved the overall mF2C architecture and functional block descriptions, leveraging the preliminary implementation and validation feedback collected from IT-1. Consequently, substantial improvements are considered in IT-2 vs IT-1 in order to demonstrate progress and evolution of the entire mF2C design. Progress between both iterations may be summarized into three main aspects:

- architectural concepts,
- blocks functionalities
- architectural specifications.

Architectural Concepts

From an architectural perspective the main improvements in IT-2 may be grouped into three contributions as illustrated in Figure 3: i) the whole hierarchical architecture is extended with more layers; ii) adding a microagent; and iii) the addition of resilience capabilities through the deployment of additional backup features.

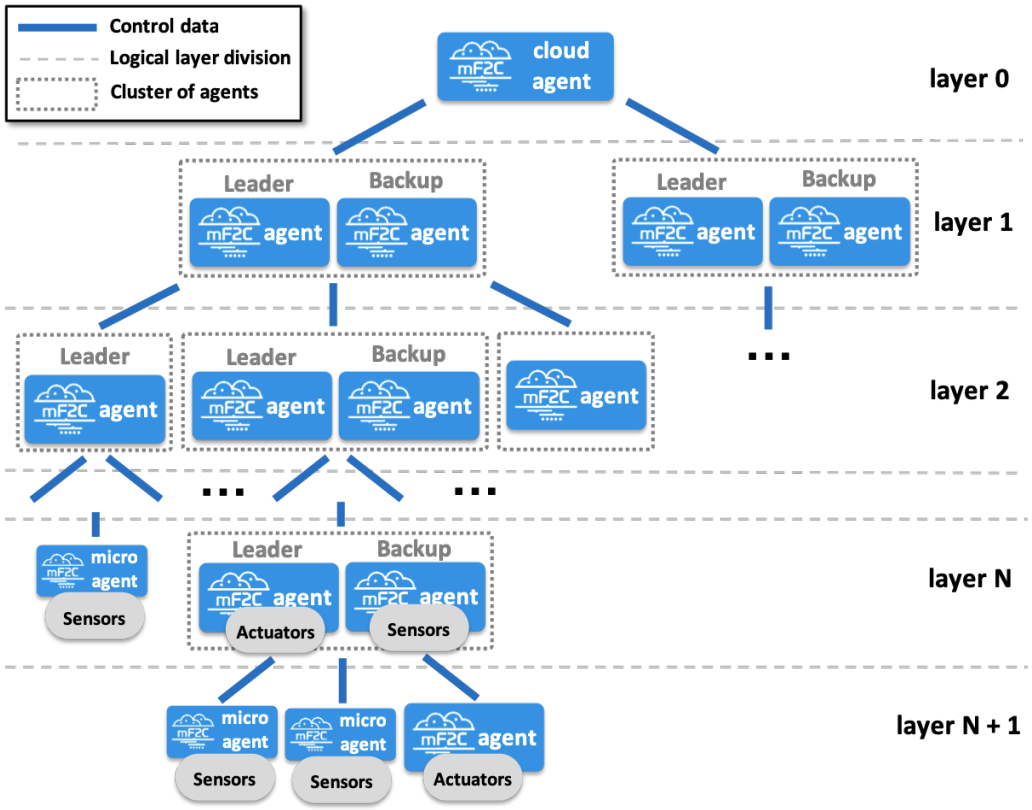

Figure 3 IT-2 mF2C architecture

The rational for extending the number of layers is not purely theoretical. Indeed, the fact of considering more layers is observed to ease clustering granularity, and as consequence enlarges the capacity to adopt new elements within the

architecture while maintaining scalability. Certainly, an obvious trade-off is between the number of layers and the additional "complexity" of managing too many layers. Policies and strategies should be defined to properly accommodate this trade-off, defining the number of layers required, therefore creating a more vertical or more horizontal architecture depending on the specific scenario.

Leveraging the unique agent design from IT-1, in IT-2 there is distinction between a full-stack mF2C agent, an optimized mF2C microagent and an mF2C cloud Agent. The agent is the default one used by most devices within the architecture. The cloud agent is a slightly modified version of the standard agent adapted specifically for the cloud while the microagent is a simplified version of the agent designed to be used in constrained devices, such as those built on Raspberry Pi.

The cloud agent can be instantiated over one or multiple private or public clouds according to the specific requirements of the user. Figure 3 shows how each layer can accommodate multiple clusters of agents, having at least one leader and if possible one backup for resilience purposes. The policies for determining whether an agent can be leader or not, may include different rationales, for example resource capabilities, connectivity capacity, etc. Importantly, the microagent can be placed in any layer along the architecture, but cannot manage other agents, i.e. play the role of a leader. As previously outlined some sensors or actuators with no computing capabilities may be attached to any agent and will be managed as "entities" with a specific set of capacities and characteristics, through categorization and classification mechanisms.

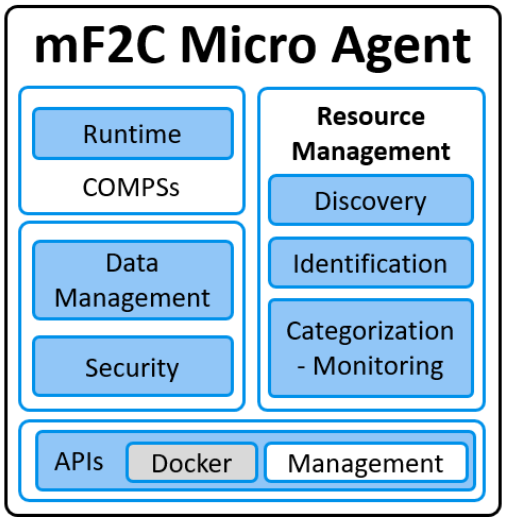

Figure 4 mF2C Microagent architecture

The final architecture proposed for the microagent is shown in Figure 4. It is worth emphasizing that the two main blocks, i.e., AC and PM, are not explicitly identified, since no management functionalities are expected for the microagent. All components in Figure 4 have an equivalent capabilities in the full agent architecture presented in Figure 2, which indicates their intended functionality, however this does not mean that the expected features are exactly the same, instead each component will be a simplified version of the corresponding component in the original agent.

Blocks Functionalities

The second key area of change from IT-1 to IT-2 resides in the set of functionalities allocated to each functional block, the AC and the PM. Figure 5 illustrates the key changes to the Agent architecture in IT-2 vs the architecture proposed for IT-1.

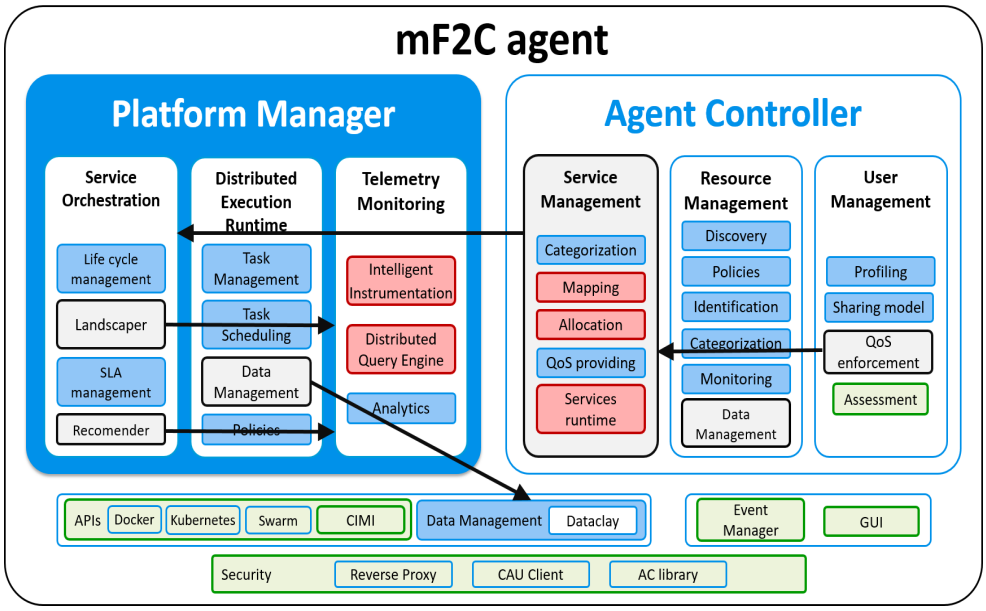

Figure 5 Agent architecture: Changes vs IT-1

The significant changes are additional components included in the Agent architecture, transposing some functionalities between AC and PM and separation of existing functionalities into discrete blocks. In short:

- The role of the Data Management as the component responsible for organizing all mF2C system data resources and providing an interface for accessing such data remains the same as in IT-1. However, while in these functionalities were split into two blocks in IT-1, one in PM and one in AC,

in the revised mF2C architecture it is positioned as an independent transversal component.

- During the IT-1 implementation phase we concluded that it was also important to include an event tracking module - **Event Manager** in the mF2C agent. This module represents a broker that will be used by each of the modules to publish/subscribe to events, e.g., service deployed, device added/removed, etc.
- Finally, the modules highlighted in red in Figure 5 were found to be obsolete and were deprecated from the architecture, while those blocks highlighted in are new modules implemented in IT-2. The black arrows represent changes in the original placement of some of the modules in PM and AC from IT-1 to IT-2.

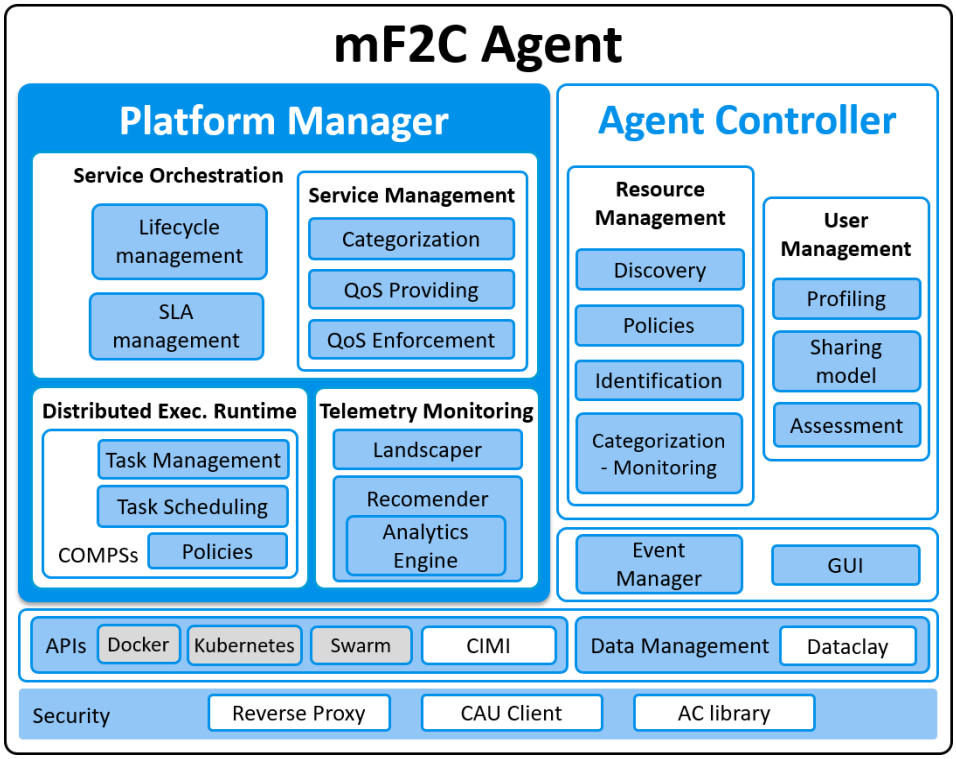

Figure 6 mF2C IT-2 Agent architecture

- The final architectural of the different agent blocks is shown in Figure 6. Indeed, Figure 6 shows the main building blocks that compose the agent entity: Platform Manager (PM), Agent Controller (AC), dataClay as a data management related component, Security block comprised of Control

Area Unit (CAU) Client, Reverse Proxy and AC library providing an agent with standard functionalities like cryptography, etc., an Event Manager, GUI and an API as an entry point

Architecture Specifications

In IT-2, we move from the architecture assumptions made for IT-1, necessary for the preliminary development, to the final mF2C architecture specifications as a stable and real Proof-of-Concept of the mF2C concept. These final specifications are more complex in nature, and are less restrictive than those previously considered in IT-1. The specifications also address the expected requirements associated with potential mF2C real world usage scenarios. The list of specifications captured in IT-1 where then split into three areas, namely architecture, agent and leader/backup selection process:

Architecture

- The mF2C architecture contains N logical layers, with N depending on the requirements of the scenario. When horizontal scalability is required, fewer layers are used, when vertical scalability is needed, more layers are created.
- There is no direct horizontal communication among agents in the same layer at the control level. However, in case of having multiple leader/backup agents per cluster, states will be synchronized among them in order to maintain statefulness within the system.
- Mobility requirements are considered.

Agents

- The Agent is instantiated in all devices with sufficiently capability.
- When the device is too constrained to run the full functionality agent, a lightweight microagent version is used instead.
- For the cloud, a slightly modified version of the agent is used, called the Cloud Agent.
- One or more agents can be clustered.
- Multiple clusters of agents can belong to the same layer.
- Each cluster has at least one leader, if there are agents dependent on it.

Leader/Backup Agent Election

- A cluster with more than one agent, will have will have designated node

configured as leader and one or more nodes configured as backup agents.

- All leader agents are able to manage service requests and their states are synchronized with the backups.
- Backup agents cannot manage service requests, but their states are synchronized with the leader agents.
- New policies are defined for the creation of clusters and selection of leader and backup agents.

Services

- Services are executed from the mF2C dashboard which is available from the interface of the agent. This GUI is accessible in all agents locally through the web browser and enables visualization of all services in the form a services portfolio that are reachable by the user logged in.
- Unlike IT-1, in IT-2 the set of categories for services and resources are not limited but are left to the discretion of the developer to define them.
- QoS functionalities are separated into two different modules: QoS providing, to report on past SLA violations, and QoS enforcing, to deploy solutions to meet QoS in runtime.

Evolving Challenges

Several challenges were identified at the time the mF2C project started. However, the unstoppable innovation in the envisioned project scenario, including cloud, edge, fog, IoT concepts and many different technologies, demands for a substantial tuning in the earlier defined challenges.

This section starts by revisiting the original set of challenges defined at the start of the project and, which have been continuously updated based on the technical and implementation challenges experienced over the lifespan of the project life.

Identification of Challenges

The set of challenges identified by the mF2C project at the beginning and considered mandatory in order to achieve the projects objectives include the following:

- **Manage a large, decentralized, heterogeneous, open, volatile, dynamic and non-trustable set of resources** (from cloud to the edge of the network), intended to supporting an efficient and transparent utilization of the available heterogeneous resources distributed at the edge.
- **Cloud/fogs identification**: An address, a label or a name must be linked to the resource (cloud and fog) in a secure and verifiable fashion. This is especially important when considering dynamic resources (especially fogs), whose time in the resource landscape is not pre-determined, constant or even guaranteed.
- **Transparently and optimally offload computations**, between fog and cloud computing systems, reallocating both resources and services, as well as executing services in parallel, addressing a solution based on a programming model handling distribution, parallelism and heterogeneity in the resources transparently to the application programmer, as well as able to handle data regardless of persistency by supporting a single and unified data model.
- **Resources discovery and allocation**: Executing a service requiring different resources (fogs and/or cloud) to interact each other will first require the proper selection, and in some cases discovery, of these resources. A management entity must be responsible for discovering the set of available resource (fogs) and then selecting the best match for the corresponding service requirements.

- **Incentivize users and devices to participate** in the paradigm through ease of operation, services customization, optimized performance as well as features of security and privacy. That is, how collaborative scenarios, based on resource sharing and clustering, extend the concept of cloud provider to an unknown frontier, creating innovative resource-rich proximate infrastructures near to the user, while remaining profitable. The concept is based on the contributory/volunteer computing, involving volunteered resources from users (resources' owners) willingness to do so.
- **Coordinated layer orchestration**: Coordinated orchestration is required to: i) generate an individual service workflow; ii) map the service workflow to the fog and cloud resources best suited for the service requested, and iii) coordinate the interactions among the different layers involved in the service execution.
- **Services execution scheduling**: Service scheduling is required to decide how the service's individual functions are split into the different fog layers and mapped onto different physical resources, and to dynamically manage schedules based on runtime conditions.
- **Semantic adaptation**: The semantic mapping between the attributes of a service and the capacities offered by a cloud and fog layers, include attributes such as static/dynamic infrastructure (whether the infrastructure is persistent in time or not), reliability (how reliable is the resource), time-to-leave (time to expected teardown), security and privacy properties, connectivity, to name a few.
- **The management of security and privacy** in F2C systems. The envisioned scenario inherits most of the issues emanating from edge devices uncertainty, with the corresponding unsolved challenges in the security and privacy arena.
- **Develop a dynamic business and market model** that can trigger new business growth opportunities. New players in the cloud and services sectors are expected to emerge in the near future, leveraging IoT deployments. It is clear that coordination is required when services are executed on resources hosted by different providers. Hence, new models of collaboration must be sought at a business level.

Technical Challenges

The challenges in the previous section have been updated over the lifespan of, transforming into the following challenges from a pure technical focus:

- Security by design provisioning in the highly distributed envisioned scenario, including highly constrained devices with dynamic behaviours.
- Strategies for resilience must be guaranteed to prevent service disruptions, as a result of systems failures or cyber-attacks. Although the high volatility of the envisioned scenario does not contribute to identification of a solution, the hierarchical architecture envisioned in mF2C may notably help in this endeavour.
- An agnostic management of a vast set of IoT devices must be also included in the solution, extending the current deployment towards a seamless approach where distinct systems may be abstracted from the mF2C system and managed according to their characteristics and features.
- QoS provisioning is achievable, however, QoS enforcing is not a simple undertaking, as it requires reassigning resources to services on the fly in order to maximize the chances of the associated SLA being met. This necessitates real time monitoring of the whole system, including resource availability and utilization as well as services performance. A functional capability which utilises the telemetry data to detect whether a service SLA will be met according to t current performance and in cases where it will not, to reassign new resources to services is also necessary.
- Aligned to the challenge above, accurate tracking of resource availability is critical in order not only to guarantee the expected QoS (easing also the deployment of a QoS enforcing strategy) but also to maximize resource utilization.
- Identifying patterns for systems mobility would also help minimize the resources discovery problems (false detection) and would also facilitate services offloading, leveraging prediction strategies based on mobility estimations.
- Defining a proper strategy for resources sharing, where device owners and potential clients may co-exist in a win-win scenario. This challenge would also include the business models and opportunities to sustain such a sharing paradigm.

- Designing strategies for optimal clustering of devices at the different layers of the architecture, not only at the bottom where IoT devices are attached to either agents or microagents but also to upper levels to decide how many levels should be considered and how many devices per leader.

Implementation Challenges

mF2C is acutely focused on delivering a final prototype of the proposed management solution. Significant efforts have been devoted to integrating the various components developed by the project and supporting technologies in order achieve the expected system functionalities, characteristics and features defined in the mF2C design phase. These integration efforts have acted as a feedback loop into the design phase particularly in the design of components within the AC and PM blocks and their interaction mechanisms. The following challenges may be added from an implementation perspective.

- Build an agent component including all functionalities all encapsulated in a light component which does not require significant hardware resources be deployed
- Build a microagent component to be supported by devices where the microagent is expected to run, i.e., highly constrained devices
- Develop a solution running on any environment (android, iOS, etc.)

Pilot Based Validation of mF2C and Corresponding KPIs

The validation strategy proposed for the mF2C project basically consists of deploying the mF2C management solution in real-world use case pilots and verifying t mF2C's benefits through well-defined KPIs.

The mF2C project proposes an incremental and business innovative approach based on three different but complementary and incremental use cases. These use cases will be used to validate the concepts developed within the project and to illustrate representative scenarios where mF2C may have substantial and beneficial impact on. The expected common benefits from a successful mF2C deployment are as follows:

- Reduced latency for latency sensitive services
- Intrinsic redundancy
- Easy access to sensors devices
- Readily available hardware
- DER (parallel execution)
- Optimization of resources and data computing (impacting QoS)
- Security (including data privacy and system security)

Figure 7 provides a mapping of these benefits to corresponding KPI's.

	Delays (Latency vs turnover)	Reliability	QoS	CAPEX OPEX	Security
Increased coverage		X	X	X	-
Reduced latency	X	X	X		-
Intrinsic redundancy	X	X	X	X	-
Sensor Access (easy deployment)				X	-
Readily available hardware				X	-
DER (// execution)	X		X		-
Optimisation of ressources	X			X	-
Optimisation of computing	X		X	X	-
Security					X

Figure 7 mF2C validation KPIs

Emergency Situation Management

The pilot for this use case was defined by mF2Cpartner Worldsensing. The use case is deployed and being validated on the smart city testbed hosted at the CRAAX Lab at Universitat Politecnica de Catalunya (UOC). The key concept is to set a real-time response to any emergency in a smart city context, particularly addressing the case of smart construction. Thus, the pilot monitors a building and when needed (building collapse, etc.) the system runs the corresponding set of actions as defined to react to a specified event. Figure 8 shows the topology of the use case including the different elements considered.

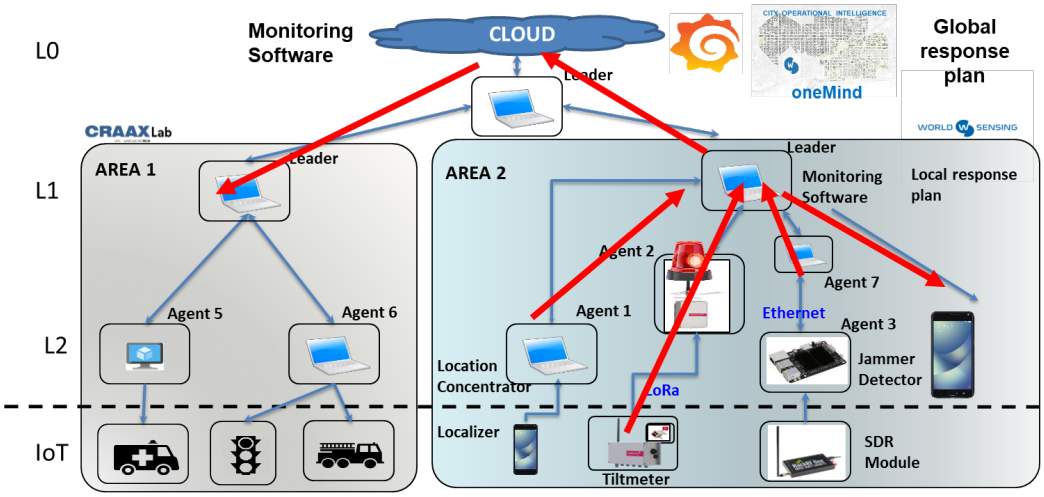

Figure 8 Validation scenario for ESM pilot

The set of KPIs considered in order to validate mF2C in this pilot are:

- **Delay**: Service delay decreased by 24%
- **Service Reliability**: Services achieve close to 100% reliability.
- **QoS**: Increase Vodafone offer of QoS of 93% by 7%, to 99%. This can be achieved due to intrinsic redundancy with the mF2C architecture, in case of cloud connection failure the service can be ran locally.

Smart Boat

This use case deployed by XLAB, focuses on enhancing the functional of remote monitoring in boats allowing them to become smarter through the deployment of the mF2C technology and better connectivity, enabling the development of enriched smart services. Figure 9 shows the topological of this use case.

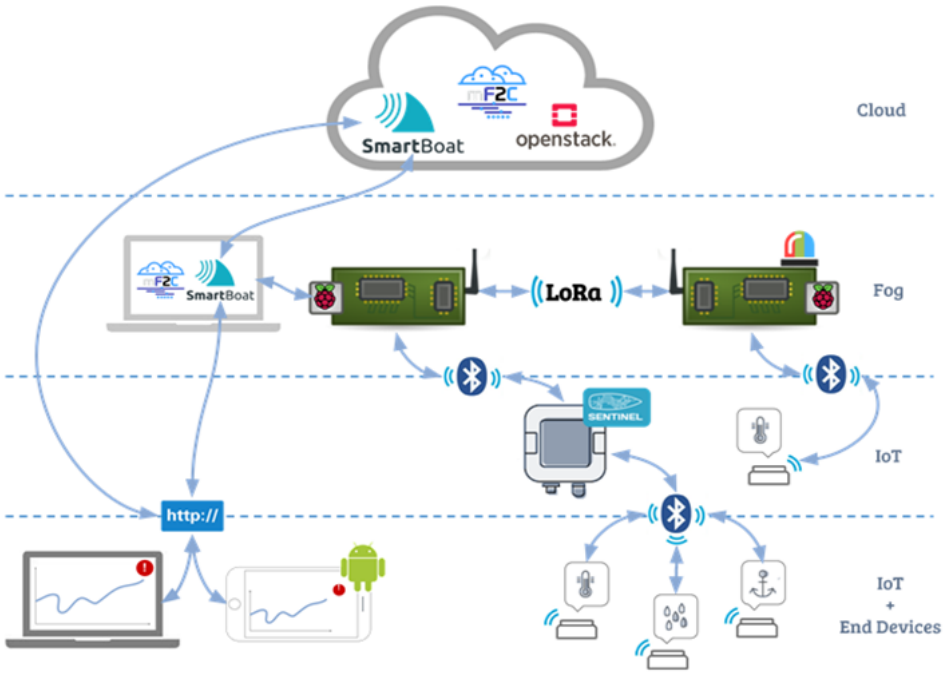

Figure 9 Example of Smart Boat topology

In order to show the benefits of a potential deployment of the mF2C technology in this pilot the following KPIs will be assessed, to determine if substantial improvement can be achieved:

- **Delay**, measured in terms of mobile device-cloud, mobile device-fog and assuming fog still works in areas without coverage
- **Expanded coverage** (20%) with a cloud independent solution where functionalities remain active even in dark zones.
- **Safety at sea**, fog P2P communication allows getting notifications in dark zones with no network coverage
- **Safety communications** enabling chatting with nearby boats sailing through the oceans without network coverage

Smart Fog-Hub Service

Engineering will support this pilot located at Elmas Airport in Cagliari utilising mF2C's technology. The core idea of this use case is to setup hubs in the airport capable of tracking the presence of people and other objects in order to provide value added services on top for proximity marketing, prediction of path/behaviour

of consumers, and taking real time decisions. Figure 10 shows the envisioned scenario for the use case.

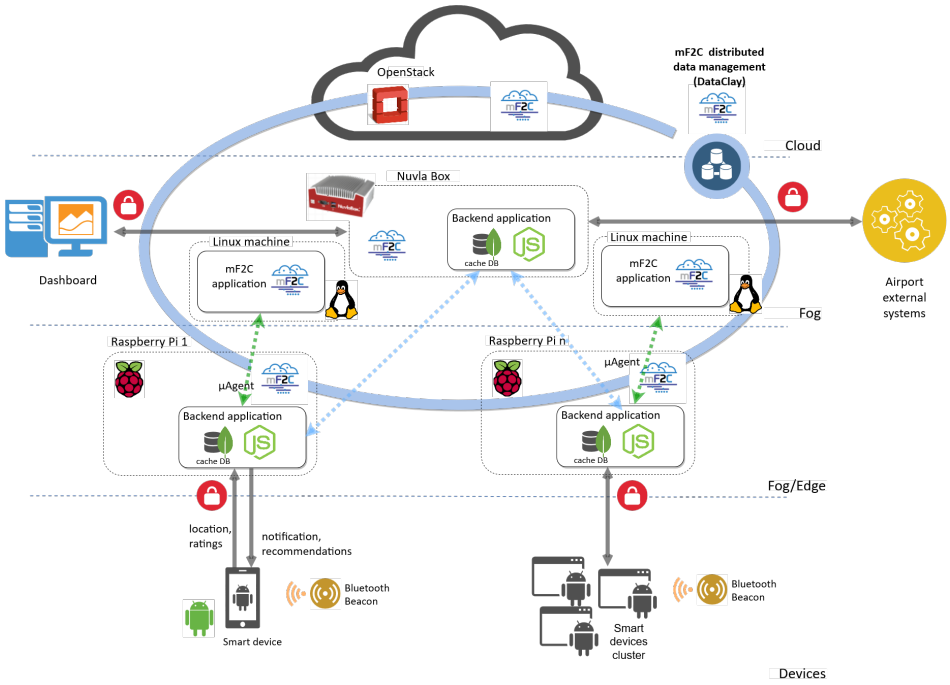

Figure 10 Smart Fog-Hub Service architecture

- The following set of KPIs will be considered to analyse how a successful deployment of mF2C technology will benefit the proposed service:
- **Response Time**: the end-user application requires real time response, compare responses in an mF2C environment versus responses in a cloud only environment, with a decrease time of 15%.
- **QoS and Resiliency**: the smartphone app is based on continuous data communication, intrinsic redundancy of the mF2C architecture guarantee better use of bandwidth and resilience capabilities.
- **Data Locality and Regulatory Compliance**: processing of personal data is done at the edge using security/privacy by design mF2C features, thus fulfilling the GDPR constraints

Lessons Learnt

The collaborations and interactions within the mF2C consortium have been a highly enriching experience for the partners, resulting a variety of valuable insights and key lessons learnt. The following is a summary of these lessons:

- Baseline technologies: Integration of different state-of-art technologies into a common solution is necessary in order to appropriate align with current trends. Indeed, any novel development must be fed by current technologies. However, the cost of such effort must be measured and minimized in order to achieve realistic timelines, particularly when dealing with the integration with a high number distinct blocks and components which comprise any complex system.
- Interdisciplinary team: Composing teams with different knowledge, expertise and backgrounds is a must in order to address the diverse set of challenges encompassed within the project. However, a time period should be defined in the early stages of the project in order agree on common working approaches and to agree on the main concepts based on a standardised and common understanding among the consortium members. This necessary “alignment” time prevents future issues and misunderstandings among the consortium as the project matures preventing conflicts and lost time.
- Real real-time is difficult: Some issues in the current deployment make real-time responses challenging. The design should appropriately accommodate the trade-off between real-time and accuracy, mainly when managing aspects related to discovery management (devices entering/leaving or real resources availability).
- Basic security by design deployment. The project was ambitious in proposing security by design as a clear target in the architectural design. However, the project’s implementation adopts this design approach in a lightweight manner. Although some strategies have been deployed with this design philosophy in mind many aspects remain yet to be addressed in order to fully embrace this objective.
- Start implementation/testing/integration earlier: Managing different technologies and deployment from different teams needs early integration efforts. This was not fully realised by the project not even through the two iterations that were already proposed to reduce the delay between the design and the integration phases.

- The implementation of independent microservices, with clear REST-based definitions for the interfaces between components and data shared between them, has facilitated parallelisation of progress without dependencies on the completion of other components. However, despite facilitating development and interoperability, the overhead added by the REST protocol is not always suited for near real-time communication.
- The Agent footprint is around 6 GB in size with the microagent being less than 1 GB. Fitting all functionalities into a light software release is critical to make the system viable. However, this has been a continuous effort, especially when dealing with the microagent component. Indeed, the microagent is designed to be deployed on low power devices, so much less than 1GB is expected for memory consumption.
- Deployment in Android is not feasible. The current deployment using Docker imposes some limitations on Android utilization. Solutions must be thought to either change the implementation environment or assume the existing limitations.
- Most operating systems do not allow access and manipulation of the 802.11 wireless protocol (used in the discovery mechanism), so mF2C has been released only for Linux, which works on clusters, laptops, and Raspberry Pi.
- The types of devices where mF2C can run is limited by the fact that certain Wi-Fi network cards, either built-in or external dongles, do not support broadcasting customized information, what is mandatory for the proposed discovery strategy.
- Finding the killer app is a real challenge! Indeed, the coordinated management is envisioned as a technological solution contributing to optimized performance for many different applications, particularly for those requiring real-time parallel task execution and large data consumption. This is the right approach for mF2C rather than a specific, single or unique application.

Conclusions

This paper provides a summary of the efforts within the EU H2020 mF2C project in relation to the design and implementation of a management solution for the F2C ecosystem. The challenges yet to be solved are described together with the key lessons learnt by the project. The paper includes the set of improvements carried out by the project from its initial iteration (IT-1) to the final one (IT-2), highlighting and justifying specific design decisions which had notable impact on the mF2C system architecture.

The paper enriches the challenges earlier identified for the project, benefiting from the experience obtained in integrating the components and validation the preliminary Proof-of-Concept (PoC) in IT-1 as well as in the final integration in IT-2 towards the final mF2C delivery. Certainly, some lessons learnt are also included to highlight what should not be repeated. The paper also includes a short overview of the pilots used to validate the project outcomes, with a clear description of the set of KPIs included for each use case to demonstrate the value-add of the mF2C solution.

References

[1] H. Gupta et al. “SDFog: A software defined computing architecture for QoS aware service orchestration over edge devices,” https://arxiv.org/pdf/1609.01190.pdf

[2] T. Coughlin, “Convergence through the cloud-to-thing consortium,” IEEE Consum. Electr. Mag., vol.6, no.3, pp. 14-17, 2017.

[3] X. Masip et al., “Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud (F2C) computing systems,” IEEE Wireless Commun. Mag., Oct. 2016.

For more information

Webpage: http://www.mf2c-project.eu/

Twitter:
https://twitter.com/mF2C_project?ref_src=twsrc%5Etfw&ref_url=http%3A%2F%2Fwww.mf2c-project.eu%2F

LinkedIn: https://www.linkedin.com/in/mf2c-project-22b4ba139/