Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem (mF2C)

Abstract

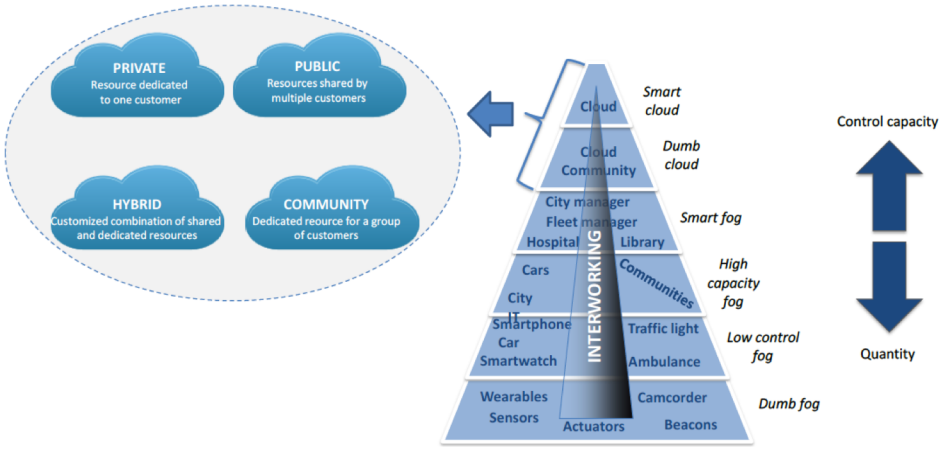
Fog computing brings cloud computing capabilities closer to the end-device and users, while enabling location-dependent resource allocation, low latency services, and extending significantly the IoT services portfolio as well as market and business opportunities in the cloud sector. With the number of devices exponentially growing globally, new cloud and fog models are expected to emerge, paving the way for shared, collaborative, extensible, mobile, volatile and dynamic compute, storage and network infrastructure. When put together, cloud and fog-computing create a new stack of resources, which we refer to as Fog-to-Cloud (F2C), creating the need for a new, open and coordinated management ecosystem. The mF2C proposal sets the goal of designing an open, secure, decentralized, multi-stakeholder management framework, including novel programming models, privacy and security, data storage techniques, service creation, brokerage solutions, SLA policies, and resource orchestration methods. This document outlines the architecture and main functionalities of the management framework designed in the H2020 mF2C project to coordinate the execution of services in this heterogeneous and distributed set of resources.

Introduction

The emergence of IoT –the networked connection of people, process, data and things – is expected to significantly increase the number of connected devices worldwide, from the billions of units we have today, to tens of billions of units expected to be deployed in the coming years. According to HIS [1] there were 27 billion of interconnected devices in 2017, while Cisco [2] predicts 50 billion by 2020. The same HIS report expects an annual growth of a 12% until 2030 to reach 125 billion of connected devices. At the same time, cloud service providers (Amazon AWS, Google Compute Engine, Microsoft Azure) are today **enabling** customers to quickly deploy a myriad of private and corporate services at comparably lower prices than buying and maintaining their own infrastructure. When combined, fog and cloud computing are undeniably setting standards in flexibility, cost, economy of scale, but also innovation in new services, devices and applications.

There is no doubt therefore that any future, enriched, smart scenario, is likely to deploy and benefit from the combined computing infrastructure, be it cloud, fog, or a combination of both. Figure 1 shows how today's ecosystem integrates centralized cloud infrastructure, with various levels (or layers) of dispersed elements starting with smaller scale clouds, some fog computing capabilities with various degrees of decision making and data processing capabilities (the stack of resources). We refer to the scenario shown in Figure 1 as a Fog-to-Cloud (F2C) system [3]. In this novel computing paradigm, 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 execution of tasks and computational offloading on the cloud.

Figure 1: Fog-to-cloud (F2C) layered structure: The stack of resources



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 resources, taking into consideration service requirements and user demands, in a paradigm-shifting scenario combining cloud and fog computing.

The F2C collaborative and coordinated computing ecosystem has been conceived to: i) efficiently and transparently use 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 (e.g., low latency, fast handover and connectivity of mobile applications), and; iii) pave the way to new business models in both cloud and smart devices sectors. Security and privacy are also addressed in a transversal fashion in F2C systems. The highly necessary features of security and privacy, throughout both cloud and fog, make the F2C system a highly interesting subject of future research and innovation. Last but not least, despite fog's potential, many devices are likely to suffer from resource constraints (limited storage, battery, compute power), which can lead to inability to satisfy service level agreements. Hence, a critical question here is how collaborative scenarios, based on resource sharing and clustering, can extend the concept of cloud provider to an unknown frontier.

In this whitepaper, we outline the main functionalities of the mF2C management framework to being developed in the mF2C project. This framework will manage the execution of applications and services in a coordinated way in this new computing ecosystem.

Challenges for an open and coordinated management of fog and cloud computing systems

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 resources, taking into consideration service requirements and user demands, in a paradigm-shifting scenario that combines cloud and fog computing. This ambitious objective may be divided into the following general challenges:

- **Manage a large, decentralized, heterogeneous, open, volatile, dynamic and non-trustable set of resources** (from cloud to the edge of the network), boosting and allowing an efficient and transparent utilization of the available distributed and heterogeneous resources 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 in fog), whose time in the market 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 that handles the distribution, parallelism and heterogeneity in the resources transparently to the application programmer. The framework is able to handle data regardless of its persistency by supporting a single and unified data model (e.g. data can be replicated "up" towards the cloud or sideways, with lower latency replicas being preferred).

- **Resources discovery and allocation**: Executing a service requiring different resources (fogs and/or cloud) to interact with 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 resources (in the fog(s) that it is responsible for) and then choosing those that can best meet the requirements of the service.
- **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 can 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 (device owners) willingness to do so, but giving users incentives (not necessarily financial) and transparency (they can see and control what they contribute).
- **Coordinated layer orchestration**: A coordinated orchestration is required to: i) generate an individual service workflow; ii) map the service workflow into 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 a service's individual functions are split into the different fog layers and mapped on different physical resources, and even dynamically managing schedules based on runtime conditions.
- **Semantic adaptation**: The semantic mapping between the attributes of a service and the capacities offered by 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 is undoubtedly inheriting most of the issues coming from the uncertainty of edge devices: they could be compromised, hacked and malicious, malfunctioning, etc. Particular challenges include preventing botnet attacks and implementing privacy controls for personally identifiable information.
- **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 deployment. 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 the business level.

With these ambitious challenges the project has been organized in two iterations, the first, iteration 1 (IT-1) from month 1 to month 18, and the second, iteration 2 (IT-2) from month 18 to month 36. A functional and working platform is expected at the end of both iterations. The advantage of this approach is that it gives the project enough time to design and implement a model for IT-1 which is sufficiently realistic that we can learn from it and reuse components, and use IT-2 to address any open issues. Additionally, it is expected that the IT-2 will further expand the functionalities of IT-1.

mF2C view

mF2C hierarchical approach: Agents, leaders and IoT

This section presents the global design of the mF2C architecture. The mF2C system proposes a coordinated management solution leveraging all existing and potentially available resources, from the edge up to the cloud, when executing a service. Figure 2a) shows examples of possible heterogeneous devices participating in the mF2C system; these devices are heterogeneous yet able to execute the services in a distributed and coordinated fashion. In order to manage this execution, we propose to organize these devices in a hierarchical architecture, as shown in Figure 2b), where the resources are categorized according to a certain policy and an mF2C agent entity deploys the management functionalities in every component within the system. We see how different layers are set (from layer 0 at cloud to Layer N+2 at the level closer to the edge) and the fact that an agent is deployed on all components. In fact, the agent software must in principle be installed all devices participating in the system; all devices thus become mF2C capable devices. However, in practice, only nly devices with sufficient capacity will have the mF2C agent installed. All information from other IoT devices without enough capacity, such as sensors and actuators (red balls in the figure) is gathered, processed and distributed by the agent connecting these IoT devices to the system.

Hierarchically, several devices are clustered under the control of one device that is defined as the leader. The policy to identify the clustering strategy and the leadership role is to be defined during the project, although characteristics such as distance and connectivity may be considered in a first approach.

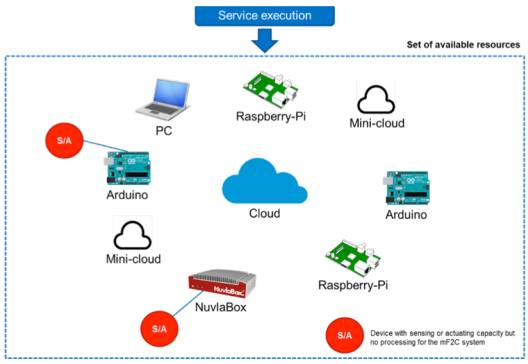
We assume that:

- Fog area or cluster is comprised of set of nodes, managed by a leader.
- Only one node acts as a leader in each fog area.
- Initially, only one leader backup node ((for robustness, acting when the leader fails), in each fog area, is considered.
- IoT devices can be connected to any of the agents in the mF2C system.

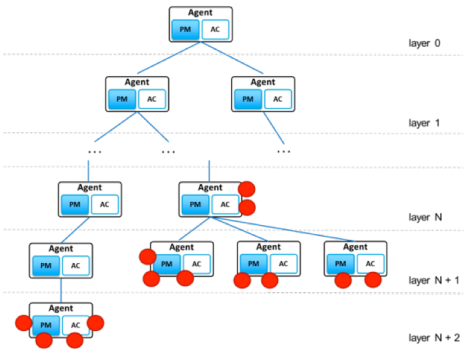
The whole set of agent management and control functionalities has been divided into two big blocks, the Platform Manager (PM), and the Agent Controller (AC), roughly as foreseen in the description of work, but with their names and roles changed slightly. In short, the PM provides high-level functionalities, 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, the Agent Controller (AC) functionalities have a more local scope, dealing with local resources and services. It is worth noting, that when the mF2C device acts as a normal agent (not being a leader) “local resource” means only its own resources (such as sensors connected to it, or its own memory), but when the mF2C device acts as a leader, “local resources” mean its own local resources plus the set of the resources of the whole fog area that it manages.

Figure 2 mF2C resources and architecture

a) Set of available resources



b) Layered and hierachical mF2C architecture



It is also important to explain which is the approach to managing services/tasks, and in particular which service/task is considered local. When a service/task is requested from any of the mF2C agents, always the responsibility of deciding if this task can be executed in that agent or delegated downward (to any of the agents in the area if the agent is a leader) or upward (to the higher hierarchical layer) is taken by the Platform Manager. If the task is delegated up or down, the communication is also done by the PMs of the agents. Only when the task is decided to be executed in the specific agent is the request passed to the Agent Controller, which will execute that task in the agent's local (own) resources. Thus, in conclusion, all the smartness to decide when and why forwarding tasks/services to higher/lower layers resides in the PM.

A final important remark refers to the databases containing the system information (resources, services and users). This information is distributed in the databases of all the mF2C agents in the system.

- A normal agent will contain information about itself and its connected IoT devices
- A leader will contain information about itself, the set of nodes in its fog area ("children"), and its connected IoT devices. Information about its "children" may be in an aggregated form. The aggregation policy is one of the topics under study in the project.
- The cloud agent will contain information (maybe in some cases aggregated) about all the devices in the mF2C system.
- Each mF2C agent will have a database with information about resources, services and users; and this database is shared by the Platform Manager (PM) and the Agent Controller (AC).

Finally, for IT-1 we made a set of assumptions, in order to simplify the set of functionalities to be implemented in IT-1.

- Only one cloud is considered
- Only three layers are considered (Layer 0: cloud, Layer 1: leaders and Layer 2: normal agents). IoT devices such as sensors, actuators, etc. are not per se a layer, but they are attached (and managed/controlled) by the any of the mF2C agents
- There is no horizontal communication among the mF2C agents, so service requests are only communicated vertically, i.e. between a leader and its children. Horizontal communication could still be implemented by relaying messages via the leader.
- Only one leader backup node is considered
- Only one node acts as leader at any given time
- Resource virtualization is not considered
- Mobility is only considered at the lowest layer (normal agents, e.g. mobile phone)

The mF2C agent: Platform Manager and Agent Controller

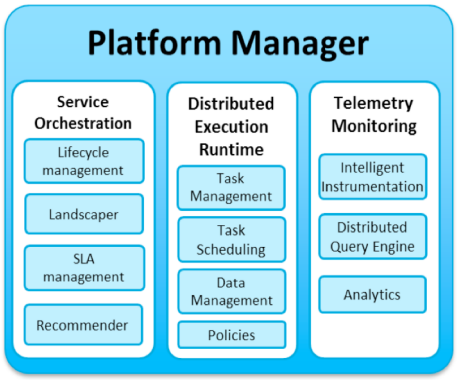
In this section, we detail the mF2C agent functionalities, divided into the set of functionalities of the Platform Manager (PM) and the set of functionalities of the Agent Controller (AC) as outlined above.

The Platform Manager (PM)

The Platform Manager is the block responsible for the orchestration of services based on the compute, storage, network and IoT resources and using a full-stack monitoring system, which receives telemetry data from different sources. This block also includes the distributed execution runtime, which coordinates the execution of end-user applications within the mF2C infrastructure. The Platform Manager can be seen as a global entity that works as a controller when it is managing agents in lower layers, and as a receiver of control data when it is being managed by agents from upper layers. The Platform Manager is divided into three main components according to these responsibilities: Service Orchestration, Distributed Execution Runtime (DER), and Telemetry.

The Service Orchestration is responsible for allocating the services to the most suitable resources. It is

Figure 3. Platform manager functional architecture



composed by the following components:

1. **Lifecycle management**: The Lifecycle Management component is responsible for managing the lifecycle of the applications to be executed by the mF2C infrastructure. This includes the initialization, the submission and the termination of these applications, among other operations.
2. **Landscaper**: The landscaper is intended to obtain a view of the whole mF2C infrastructure for fog/cloud infrastructures. The Landscaper must be able to query all physical elements of the mF2C infrastructure for a given mF2C area, that is led by a leader. This includes all the physical machines and parts of, e.g., CPU resources, storage, memory, etc. Metadata about these elements are also required, e.g., CPU core count, speed, cache size, etc. Each node stored in the Landscaper should have a mapping to any telemetry probes that are measuring performance on that machine; as well as a subscription to an event system to ensure any changes in physical or service layers are updated accordingly (service stop/start, machines added/removed to cluster)
3. **SLA management**: The SLA Management component is responsible for managing the SLAs between the parties involved in a service on the mF2C platform: the platform and the platform users. The component is in charge of generating, storing and observing the electronic documents that describe the expected service level of a service. The agreements contain functional and non-functional terms that describe the service being delivered.
4. **Recommender**: Prior to deploying a service, the Lifecycle Manager module will check with the Recommender for an appropriate recipe of suitable type of resources for this service (based on previous analysis). The recommender should match the characteristic of the service (obtained by means of the Service Categorization module) as well as the analytics from previous executions. To that end, the Recommender module must store the heuristics and models derived from the output of the Analytics module (in Telemetry component).

The Distributed Execution Runtime (DER) component receives the requests for the execution of the services/tasks and optimizing their execution on the available resources. As mentioned earlier, these local resources could be those of the agent where the application is started, or resources managed by agents of other levels of the architecture. The DER is composed of four different sub-components:

1. **Task management**: The DER considers applications composed by pieces of software encapsulated as methods called Core Elements (CE). The main purpose of the runtime toolkit is to orchestrate the execution of CE invocations (tasks), and thus to fully exploit the available computing resources. The Task Management component should receive the request for the execution of tasks from the Lifecycle Manager
2. **Task Scheduling**: The Task Scheduling component is in charge of distributing the tasks generated by the execution of the applications on the local resources selected by the Lifecycle Manager. In the case of resources belonging to another agent, the DER has to interact with the same component in other levels. It also needs to provide a means to get information on the finished tasks, and eventually get their results.
3. **Policies**: The Policies component is needed to support the runtime in the selection of the resources for the scheduling of tasks. The Policies component provides the list of resources to the Task Scheduling component. The current implementation of the runtime only supports the usage of resources (or resource providers) whose description is provided before the instantiation of the application. A basic requirement for the project is to allow the registration of additional resources dynamically, allowing the runtime to have an updated list of agents that can provide nodes for task execution.
4. **Data management**: The main responsibility of the Data Management in the PM is to keep the metadata of the objects that are stored by the Data Management component in the AC. The Data Management must react to the corresponding requests to get or update metadata. In case an agent is a leader, it must also be aware of its children

and the data they contain, so that the required replicas can be managed in such a way that data is accessible from the agent that will need it, which is not necessarily the one that created or updated it. This requirement is derived from the nature of the mF2C platform, where devices may lose connectivity, but they should be able to perform their functions even though they are isolated.

Finally, the Telemetry and Monitoring component analyses service performance on the infrastructure it is deployed on. The three main components are:

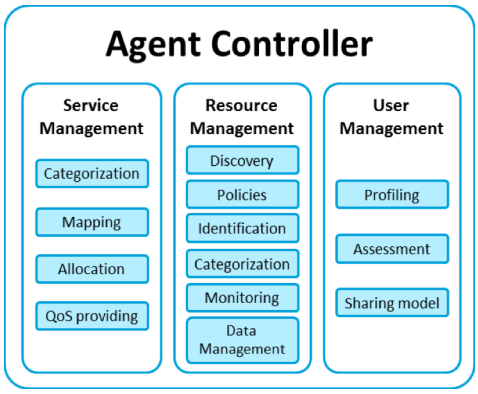
1. **Intelligent Instrumentation**: This component provides the telemetry collectors and aggregators of the metrics. A telemetry framework will perform the Instrumentation on the nodes of the mF2C cluster, measuring performance of key variables. The Intelligent Instrumentation module will monitor the telemetry metrics, e.g. setting the collection parameters of probes according to device constraints.
2. **Distributed Query Engine**. The Distributed Query Engine should provide a single API to facilitate the querying of all telemetry data captured. This abstraction layer reduces accessing telemetry data from multiple locations to a single source. Each probe will register with the query engine, notifying it of identity, metrics captured and publishing location; and also, it will register with the Intelligent Instrumentation module so that it can receive a notification to throttle its measurements and publishing frequency.
3. **The Analytics module** characterises service execution by mapping the service's deployment configuration against telemetry captured for those same nodes:
    a. **Deployment configuration**: the Analytics module needs to be able to query the deployment configuration for any given service that has been deployed by the Lifecycle Manager. This may be a currently executing service or a historical deployment. It is envisioned that the Landscaper will track service deployments over time, so should be able to take both a service identifier and a time window as parameters.
    b. **Telemetry**: For each node in the subgraph of the landscape returned, the Analytics module needs to query all Telemetry metrics relating to each node of the subgraph from the Distributed Query Engine. Each individual query should include a machine identifier and a time window.
    c. **Analysis**: A number of analytics algorithms will run against this data to derive heuristics and models will be stored in the Recommender system.

The Agent Controller (AC)

In the distributed and coordinated strategy proposed in mF2C, where services are executed in different devices, the PM includes the logic of the system, taking decisions based on a more global view. On the other hand, the AC has a more local scope, focusing on local resources, services and users. Regarding the services, the AC only controls and manages the services being executed in its own device. Finally, the AC also manages the preferences, roles, profile, etc., of the user (owner) of the device. Figure 4 shows the AC and its set of functionalities. The set of AC functionalities is split into three main blocks, Resources, Services, and Users Management.

The Resource Management component collects and manages local resources. However, due to the hierarchical nature of the proposed mF2C architecture, resources are grouped into clusters of devices, with one of these devices being the leader. For this reason and despite the local scope of the AC, in the case of a device being leader, its 'local view' includes its own resources and the resources of the

Figure 4 Agent Controller functional architecture



devices forming part of this cluster. It is also worth mentioning that, although the PM includes all the smartness of the system, in the agent entity, the database is unique and both the PM and the AC share it. For these two reasons, the hierarchical architecture and the shared database, one of the main responsabilities of the AC regarding the resources can be summarized as:

- Filling in the resource database, to be used by both PM and AC, with information about:
    - Own resources if the devices is part of the cluster but it is not a leader
    - Own resources and the resources of the "children" if the device is the leader of the cluster.

The six subcomponents of the Resource Management are:

1. **Discovery**. This subcomponent is in charge of discovering resources in a fog area managed by a leader. The discovery component should allow the leader to advertise its presence, as well as to allow the agents to detect a leader in their vicinity. In the case of a new agent in a fog area, and after the success of discovery process, the new agent will join the mF2C system, being part of that fog area.
2. **Policies**. The policies block will be a set of rules to be applied and used by different blocks in the Agent Controller. This set of rules could be changed by the PM, according to a high-level policy managed by the PM. Examples of these policies are: the clustering, the discovery (related to the frequency leader's advertisement), the leader selection, the backup selection, the protection and the resource aggregation policies.
3. **Identification**. The objectives of this module are to provide every device participating in the mF2C network with a globally unique ID, aimed at facilitating an unambiguous resource identity and to establish mechanism to update and/or revoke the resource ID. The identification component is divided into two sub-components, the registration and the identity management. While the registration takes place in a cloud server, the identity management is executed by the agent in the resource itself. During the registration phase the IDKey is generated and delivered to the user according with the chosen registration method. e.g. to the mF2C app on the user's phone). Thus, Identification can cover three different aspects: the unique identification of the device, the association of the device with the user (ownership) through the IDKey, and the cloud-issued credential which is issued to the agent controller. The latter allows the agent to establish trust across fogs, because the issuer of the credential resides in the cloud and is shared by the fogs.
4. **Categorization**. The basic objective of the Resource Categorization module is to provide the information about the resources. When running this module, the system is able to determine the hardware (storage, RAM, processor, etc.), power, software (operating system), security requirements (data, device, and network), attached components (webcam, Printer, scanner etc.), attached IoT information (sensors & actuators), and also information about resource behaviour and features (i.e. mobility). This information is stored in the resource's local database and, at a later stage, this information may be aggregated to be shared with the leader in higher layers.
5. **Monitoring**. The Monitoring module is 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. Each probe is required to perform 3 main functions:
    a. **Collect metrics**: Software probes must be able to capture metrics from hardware (both in-band and out-of-band), from any software source: host O/S, middleware and hosted application. Collectors should be able to query the output of other collectors as this could impact continued operation.
    b. **Process metrics**: Captured data should be passed through customised filters to perform a defined action on the data, e.g., generate average, standard deviation, etc.
    c. **Publish metrics**: Processed data should be published to defined destinations, e.g., file, database, message queue. The publish

module should be able to analyse this data prior to publishing so as to decide how much data to publish, e.g., publish averages, anomalies only, etc.

6. **Data management**. The data management functionality of the AC is in charge of allowing applications or other functionalities to store, retrieve, and delete data in mF2C. This data can be either the information needed to manage the platform resources, services and users, or the data needed or generated by the services themselves. For the data managed by the mF2C platform, this component is also in charge of managing the appropriate replicas of each piece of information, in such a way that data is accessible from the agent that will need it. Also, this component must react to the requests coming from the Data Management component in the PM regarding the data locality requirements from the Task Scheduling component, and the deployment of classes so that objects can be accessed. As mentioned earlier, a particular data challenge is privacy of personal identifiable information, so the first step is data categorisation, as mentioned in the security section below (note that this is different from the data resource categorisation of the Service Management). A more thorough implementation of data privacy (e.g. addressing any gaps in meeting the requirements of the General Data Protection Regulation GDPR [4]) should be available in IT-2.

The Service Management component is responsible for the orchestration of local services. The main functionalities of this block are: categorization, mapping, allocation and QoS provisioning. The service requests are decomposed into tasks in the Task Management block of the Platform Manager. After that, the Task Scheduler block decides where each individual task will be executed, and tasks are then deployed to the corresponding agent controller of each selected agent. The SM functionalities are:

1. **Categorization**: The categorization component receives a service request, and categorizes this service according to some defined attributes. The attributes defined in a first approach are CPU, Storage, Network, Memory, Priority, Time limit and Location.
2. **Mapping**: The mF2C control architecture leverages a distributed and hierarchical control topology intended to map service requests into the most suitable resources for a successful service execution. The service requests can be decomposed into tasks in the Task Management block of the Platform Manager, and the Task Scheduler block decides where each individual task will be executed, that is, the set of agents where the tasks will be deployed. Once the resource (Agent) is selected, the mapping consists of selecting the resources matching the requested task in the own resources of the agent (be they own agent resources or devices non-mF2C capable attached to it).
3. **Allocation**: This subcomponent is responsible for the optimal allocation of available resources in the agent to the various tasks requests, during the mapping process and in the runtime execution phase. In any case, all communication will go through the mapping component, and its unique functionality is to allocate available resources to the various requests, trying to meet security and privacy rules, cost models, while also guaranteeing overall optimal resources usage.
4. **QoS provisioning**. The QoS provisioning subcomponent is, unsurprisingly responsible for guaranteeing the required quality of service, according to the SLAs. For each service, it will contact the SLA Management block in the Platform Manager to get information about the expected service requirements. The QoS provisioning subcomponent will be contacted by the Lifecycle management component for each service query to get the characteristics of the candidate devices to execute the service. This information allows the QoS provisioning block to discard the unsuitable candidates. The parameters that should be defined will depend on the different services characteristics, but in a first approach the focus will be on the service execution time.

Finally, the User Management module is responsible for managing the profiling and the sharing model properties of the users who have access to the mF2C system and the applications running on top of it. This module is composed by three subcomponents, Profiling, Assessment and Sharing model, described as follows:

1. **Profiling**: A user profile is the collection of personal data related to a specific user (be it both, the user/owner of the device or the mF2C consumer client executing a service) and the digital representation of a person's identity. It can be also considered as a logical representation of a user model. The user profile information can be extended according to the user's preferences and behaviour. In a first approach, this information includes the user's identification key (if available), the user's email (if they choose to provide it), how the user joins mF2C (as a contributor, as a consumer, or both), and the parameters such as services the user is allowed to use, the services the user will allow to run in his/her devices, maximum number of services allowed to run in their devices, security levels defined for different data flows related to the user, etc.
2. **Assessment**: The User Management Assessment component is responsible for checking that the mF2C applications meet the sharing model and the profile properties defined by the device's user. The main functionalities are:
    a. Double check that the profiling properties are met
    b. Double check that the shareable resources constraints are met
    c. Send a warning to the Platform Manager if some constraint is violated.
3. **Sharing model**. The Sharing Model component is responsible for defining the resources that the device's user wants to share through (or with) the mF2C system. The functionalities included in this module are the following:
    a. Definition of the resources that will be available to the mF2C applications, such as memory, storage, etc.
    b. Definition of the rules or constraints in order to establish not only the amount of resources to be shared, but also the conditions under which these resources should be increased, decreased or not shared at all, such as maximum CPU, memory usage, battery or/and bandwidth limits, etc.
    c. Definition of reward mechanisms for these resource contributions, like some kind of service execution credits, economic rewards, etc.

mF2C security approach

Security Policy

Whenever a system is designed it must consider security as a key feature, addressing aspects related to data, connectivity and hardware interoperation. To that end, it is essential to define a security policy, which must guide implementers towards thinking on how security must be deployed.

mF2C system security

Table 1 lists the security requirements and functionalities we have identified as necessary in the different layers, mapped into each one of the functionalities expected from the mF2C agents.

After identifying the mapping of the agent functionalities into the security requirements, we may conclude that agents are not necessarily isolated entities and they, and their security functionalities, will be implemented/supported and deployed through an agent controller.

The security features can be implemented in the agent controller's functional blocks directly, or in the agent controller itself. The advantage is that each controller will have all the functionalities it might need, at the cost of increasing the size and possibly the computational requirements of the agent controller (all calculations are done locally).

Table 1 Security requirements in functional blocks in mF2C agents

	functional blocks	Security requirements												
		Authentication	Key distribution and management	Identity management	Access controls	Secure Communication	Data Security	Integrity	Availability	Confidentiality And privacy	Non-repudiation	Intrusion detection	Intrusion prevention	Secure Mobility
RESOURCES	Discovery	✓			✓	✓		✓	✓	✓		✓		✓
	Policies	✓		✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	Identification	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Categorization			✓		✓	✓	✓	✓	✓		✓	✓	
	Monitoring					✓	✓	✓	✓	✓		✓	✓	
	Data Management				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SERVICES	Categorization													
	Mapping													
	Allocation													
	QoS providing	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Service Runtime													
USER	Profiling				✓	✓	✓	✓	✓	✓	✓			
	SLA				✓	✓		✓	✓	✓	✓			
	Sharing Model				✓	✓	✓	✓	✓	✓	✓			

Use cases validated in mF2C

The mF2C project proposes a business innovative approach based on three different but complementary and incremental use cases to validate the concepts developed within the project, to show a wide spectrum of scenarios mF2C may substantially impact on.

Emergency Situation Management in Smart Cities (ESM)

Continuing into this century, society has supported a movement of people from rural areas to cities. Nowadays, more people live in urban environments than in rural ones. It is estimated that this process will not stop and within 20 years the urban population will be around 5 billion of people. The big challenges for the whole society will be related to resource management and mobility throughout these overcrowded environments. mF2C works in the development of a model for smart cities to facilitate the deployment of innovative services in highly valuable sectors (health, traffic control, entertainment, etc.) while also making the most out of the set of IoT devices deployed. However, for that evolution to occur an efficient coordination between all the actions is the key to have the maximum positive effect in the fulfilment of our objective, which is actually the main mF2C rationale. Thus, we propose to use a Fog to Cloud management (mf2c) to handle an emergency situation in smart cities, since mF2C allows an immediate, safe and reliable response to such an event.

One of the proposed services in this emergency management for smart cities is the detection of the collapse of a city construction, see Figure 5. There are areas in the world where seismic movements are very frequent. These natural phenomena can cause serious accidents due to the collapse of buildings. The severity of the accident can range from a simple movement of land without any notable consequence to a collapse with fatalities. Among this range of values, the performance of emergency services is vital to reduce the negative consequences that may occur as much as possible.

It will be necessary at all times to have different types of sensors that allow us to know the real state of the construction. In addition, it must be guaranteed that sensor data is being received at all times. To this end, mechanisms must be used to verify and restore communication between the sensors and the data collection center. Based on the values provided by the sensors, an interpretation system must decide the degree of action to be taken and activate the services corresponding to the said action.

To know what kind of services are the most appropriate to be activated, the mF2C system will provide the management between all the elements of the city. While the fog layer provides a rapid response to an emergency, the connection with the cloud allows optimizing the resources to be used based on the historical knowledge of similar situations.

Enriched Navigation Service

Sentinel is an IoT device consisting in different sensors currently applied in the navigation sector for vessel monitoring. There are 1,000 Sentinel devices mainly in the Adriatic Sea, aiming at improving quality of the sailor's journey. As of today, Sentinel devices work in an isolated way, hence losing the potential

Figure 5 Collapse building detection in a Smart City

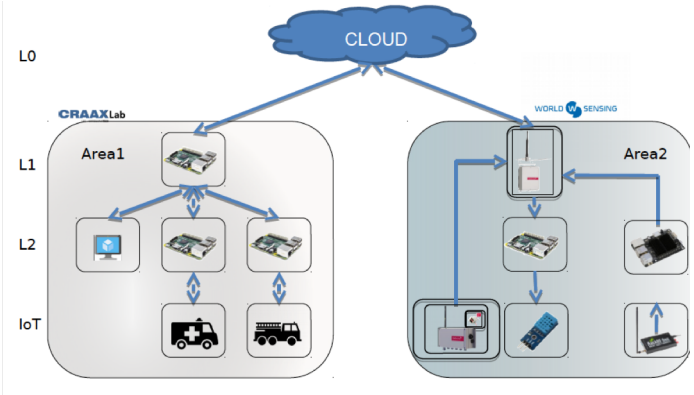
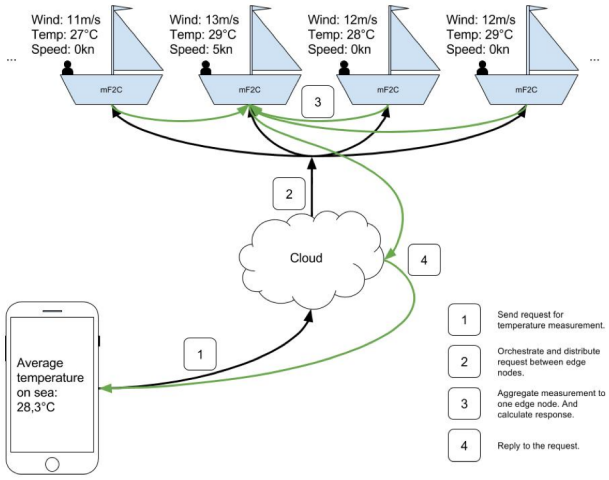


Figure 6 Example of Enriched Navigation Service



benefits brought by correlated data processing. The mF2C project can help on developing and easing the implementation of novel services enriching the Sentinel port-folio. However, most of the ideas for innovative services require the missing data correlation but also additional capacities such as interaction with external data, etc.

One of the proposed services, is illustrated in Figure 6, where an agent located on land or in the sea, wants to know the average temperature on sea:

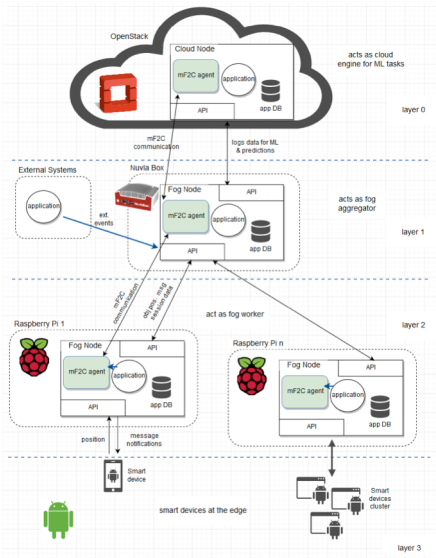
1. It sends the service request to the agent at cloud.
2. The PM of the agent at cloud decides which are the needed resources and allocates tasks in different ships (as explained in Figure 6)
3. One of the selected devices has the task of aggregating the data from itself and the data coming from the other devices, and sending it to the agent at cloud.
4. The agent at cloud replies to the request with the average temperature on the sea.
5. One of the main characteristics of the proposed services is the use of Wi-Fi or LORA for connecting between ships, and the use of 4G/5G for connecting to the agent in cloud.

Smart Fog-Hub Service

The main idea 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 value added services on top for proximity marketing, prediction of path/behaviour of consumers, and taking real time decisions.

Let us consider an airport as a small city in which a large number of people must spend a long period of time (more than 3 hours) walking through it. In this space, many services are offered to users such as shops, restaurants, relaxation areas, etc., which are distributed throughout the airport. Users can dedicate their waiting time to make use of these services but they must always be attentive to the time of boarding for their flight and to the location of the boarding gate. The uncertainty caused by not knowing the time a user needs to get from any point in the airport 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 at the time of the said boarding. We must also add the possible incidents caused by delays in flights or changes in the boarding gate. It is true that in some airports the estimated times to get from one area to another are indicated in the airport signage. But this information does not provide any reliable information that allows the user to spend their time waiting to make use of airport services in a calm and relaxed way, without having to constantly worry about their physical situation inside the airport and the time remaining before the boarding time. In addition, these times are affected by the different agglomerations of people that occur irregularly (departure of passengers from a flight, queues at boarding gates, luggage control, etc.) throughout the entire airport's passable space. This means that most users do not move out of a radius near the boarding gate during the last hour of boarding their flight.

Figure 7 Smart Fog-Hub Service architecture



Thus, many of the services offered at airports depend on the area where the boarding gate is located and, therefore, are limited to users from other areas. A solution that has been implemented is the replication of services in different areas to give maximum coverage to users. However, this solution is totally inefficient and does no fulfil its objective. In addition, only large commercial companies can afford to make such a replica.

Figure 7 shows the architecture overview of the proposed Smart Fog-Hub Service, with the following layers:

- L-0 Cloud: OpenStack engine for heavy processing, e.g., ML
- L-1 Fog Leaders: Fog aggregator for underlying layer, and interface with external systems for relevant events gathering, e.g., Admin Dashboard; Flight gate opening, last call, etc.
- L-2 Fog workers: Keep devices position, session-oriented information, dispatching of events, delivering notification of Points of Interest (PoI) in proximity
- L-3 user devices: Without an mF2C agent, using a specific android app, they connect to L-2 nodes using security protocol, according to a certain privacy policy. They must agree on terms of condition

The main features of the developed services will be:

- able to track the presence of people and objects in the field
- develop value added services for proximity marketing
- suggestions on best use of airport services
- prediction on path/behaviour of consumers

With these main features some of the specific services may be:

- Location of the boarding gate
- Time of boarding / departure of the flight
- Services offered by the airport
- Physical situation of available airport services
- Service selection:
    - Calculation of the actual time of access to the airport service
    - Time and place of offers / activities / performances offered by the service within the airport.
    - Guided path through the airport facilities to get to the preferred services.

Who mF2C benefits?

Following recommendations provided by ISO standards related to Cloud Computing [5], these actors and roles are classified in three categories: Customer, Provider and Intermediate/Partner Side.

Customer Side

- Application Developer, a technical person capable of developing a software application to be operated in an mF2C-powered installation.
- IoT Platform / Solution Provider, The provider of an end2end IoT specific solution or Platform candidate to consume as final user mF2C system and services.

Provider Side

- Fog Service Provider, the provider of a the provider of a single or multi-fog set of instances and services which comprise and include software, platform and/or infrastructure (compute, storage, and networking services) operated from geographically dispersed close to the edge clusters of resources.
- Cloud Service Provider, the provider of a single or multi-cloud set of services which comprise and include software, platform and/or infrastructure (compute, storage, and networking services) operated from traditional data-centers in private /public or hybrid forms in an "as-a- Service Model".
- Resource Contributor, the provider of a resource or a set of them which does not offer it in as-a- Service" but in a contributory model.

Intermediate/Partner Side

- Fog Equipment Provider, an equipment provider which sells Fog and Edge specific Hardware.
- Cloud Equipment Provider, an equipment provider which sells Cloud capable hardware, traditionally servers.
- Sensor Provider, Wireless Sensor Networks hardware manufacturers.
- Network provider, Network services provider or operator provided in isolation to additional services commonly tagged as Cloud-services.

It has to be noted that currently in the IoT, Fog (Edge) and Cloud markets, many variations and aggregations of these Actors can be found.

Conclusions

This paper is intended to bring light to the key issue of making the best of the scenario set by putting together cloud and fog resources. As widely accepted, cloud and fog are expected to collaborate so a key challenge comes up when setting a coordinated resources framework where services may be executed at cloud or fog only leveraging real resources characteristics and availability.

In this paper, we describe the mF2C initiative that presents an ongoing work intended to design and implement a coordinated management architecture, pushing for innovative solutions dealing with the set of foreseen challenges. These challenges are also included in the paper as well as tentative directions and trends, as active lines of work within the project, for a clear and comprehensive understanding. Finally, some use cases are also presented to illustrate the benefits that such a coordinated resources management may bring to real-world scenarios, supported by flagship companies in their individual sectors.

Certainly, being mF2C an active and ongoing effort supported by industrial and academic members, we are open to any collaboration that may help solve the already identified open challenges as well as any suggestion for testing, validation, etc. More information about the project as well as contact directions may be found in the links included in the next page.

References

[1] "The Internet of Things: A movement, not a market" HIS Markit, https://ihsmarkit.com/Info/1017/internet-of-things.html

[2] "How Many Internet Connections are in the World? Right. Now" Cisco Blogs by Karen Tilman. https://blogs.cisco.com/news/cisco-connections-counter

[3] X.Masip-Bruin, E.Marín-Tordera, A.Jukan, G.J.Ren, G.Tashakor, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud (F2C) computing systems", IEEE Wireless Communications Magazine, Vol. 23, Issue 5, October 2016.

[4] EU General Data Protection Regulation https://www.eugdpr.org/

[5] http://standards.iso.org/ittf/PubliclyAvailableStandards/c060545_ISO_IEC_17789_2014.zip

For more information

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

Deliverables: http://www.mf2c-project.eu/blog/press-room/deliverables/

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/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730929. Any dissemination of results here presented reflects only the consortium view. The Research Executive Agency is not responsible for any use that may be made of the information it contains.