2019

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Control resilience in a F2C scenario

MASTER IN INNOVATION AND RESEARCH IN INFORMATICS (MIRI) – MASTER THESIS

Computer Networks and Distributed Systems

ALEJANDRO JURNET BOLARIN

THESIS TUTOR: BEATRIZ OTERO CALVIÑO

THESIS SUPERVISOR: XAVI MASIP BRUIN

Department of Computer Architecture (DAC)

THESIS DEFENSE DATE: **01-05/07/2019**

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

ABSTRACT (1000 WORDS)

With the actual Internet-of-Things (IoT) trend and the early adoption of 5G technologies, we are foreseeing an explosion on the number of mobile devices and “things”. As the amount of connected gadgets in the extreme of the network is growing, network traffic and infrastructure will also evolve to provide better connectivity and new paradigms. For instance, as the data is generated at the edge of the network, services can be deployed near to the data source to reduce delays and decentralize the execution, impacting on the overall energy consumption and efficiency of the resources. This trend supposes new challenges in terms of mobility of the devices in the network, big data storage and management, network dimensioning and capability, and energy consumption or environmental impact between other issues.

Edge computing is already proposing scenarios and solutions to some problems, and the progression of the Fog-to-Cloud (F2C) concept provides the integration of mobile devices with cloud infrastructure in a single platform. F2C proposes the coordination of devices at the edge up to the cloud following a hierarchical model on control communications, coordinated by devices elected dynamically. Yet, mobility supposes a serious concern regarding the availability and reliability of the system to execute services with an enhanced Quality-of-Service (QoS). The control on the different clusters of devices at the edge, in a F2C scenario, supposes a major concern in terms of protection and recovery in case of failure.

As an example, the execution of an emergency detection and resolution service is a perfect use case of a critical real-time service that can be executed in a F2C context. Let’s suppose a scenario with a car crash in the middle of a metropolitan and intelligent city, where immediate actions should be taken. Being able to directly change street lights to redirect the traffic out of the site of the emergency, collect direct data from the persons injured and send it directly to the hospital, or direct coordination between emergency vehicles and normal automobiles are great benefits that can improve the actual emergency procedure and potentially save more lives.

That amount of communications and actions require the coordination of such amount of heterogeneous devices, that is very difficult to reestablish any static policy or procedure. In fact, if the element controlling the execution of the service fails due a connectivity error with some of the devices participating in the operation, the service cannot be executed properly if no recovery mechanisms is activated with a very low delay, and restarting the service from scratch should be not an option. Protecting the service starts with the protection of the infrastructure.

Control Resilience in a F2C scenario project proposes a novel architecture and a proof-of-concept implementation to provide resilience of this structures, and the set of rules applied to them. The provided solution of the project, called Control Resilience Management (CRM), can be deployed as a standalone component or integrated into any F2C application to help with the reliability issues and profit from other capacities. Its main components focus on the protection of the control elements on the clusters or also called areas, and the incorporation and spreading of rules into them. As F2C is also a new and evolving technology, the architecture of the CRM is designed to be expandable or modified to fit in a variety of different scenarios and architectures.

The CRM module is able to surveil a control element and overtake its responsibilities if a failure is detected, by providing a strategy and architecture to constantly be ready and change the fastest as possible. Additionally, if the current policies of the system enable that the controller element should be changed when some specific parameter is no longer meetup or by a manual decision of the system manager, the controller (or also known as Leader in this project), is replaced by another selected device. Moreover, the set of rules applied to the area and enforced by the Leader are replaceable and expandable thanks to the design and mechanisms provided in the CRM. Furthermore, to make the CRM independent to any third-party component or static configuration of the creation and management of the area topology (set of devices that are managed by the Leader), it also includes a discovery mechanism that provide dynamicity and control when deployed alone. Finally, to allow the integration of the CRM into another F2C application, a specific component is designed to allow such behavior and provide newer functionalities and enhanced mechanisms.

The developed prototype of the design of the project has been validated with a set of individual tests, on specific scenarios and situations that are specially created as the baseline that the project should accomplish. Such tests have been executed in devices with different resource and computing capabilities, as it is found in a real scenario. Moreover, the project has been also validated integrating the solution into a real F2C application called mF2C agent, actually in its final development phase, to fulfil the required resilience functionalities on the proposed architecture. The documentation and implementation of an API in the CRM allows an easy integration and management of the solution without entering into the code, as it also provides a web frontend that allow the creation and dispatch of message requests to the module.

CRM is conceived to be a building block from where it can be expanded or used, contributing to the creation and evolution of the F2C systems. The open code and design allow the customization of its internal procedures and mechanisms, the addition of newer functionalities. Moreover, the project also proposes some further steps that can be done in the future, and provide a new whole set of functionalities that may improve the overall system.

Keywords:

Edge Computing, Fog Computing, Resilience, Protection Mechanisms, Fog-to-Cloud, F2C, Policies, Leader Election, Leader Protection, Policies Distribution

01.INTRODUCTION

Nowadays, society is living on an evolving world full of new technologies and improvements. From the agricultural revolution and the industrial revolution to the creation of the Internet, mankind has never been so close and connected. In fact, since the beginning of the Internet, we have transitioned from the fixed Internet and websites to the current Internet of Things (IoT) and the forthcoming Internet of Everything (IoE) [1] paradigms. IoT changed the relationship with humans and machines [2], putting the focus of the data sources on the "things" rather on the people. In other words, today machines generate more information that humans have generated ever.

Data availability is a great deal, as new services emerge and society evolves. Unfortunately, data is not information, therefore requiring some processing to become useful information –data are useless if is not processed. Services and applications transform data into meaningful information. Specifically, this huge amount of data (in the order of Petabytes or even Zettabytes) is distributed along a diverse collection of devices, sensors and databases, with different syntax and different ways to access. Moreover, data is usually located at the extreme of the network, along with sensors and other devices, and commonly sent to Cloud infrastructures to be stored and processed.

Mobility is another concern on the actual paradigm. In 2017, the amount of wired devices was estimated to be around 48% and predicted to be drastically reduced to 29% by 2022 [3]. Moreover, the data traffic will increase astronomically up to 77.5 Exabytes per month on 2022 worldwide only in mobile communications.

Traditional cloud solutions are becoming inefficient to handle this picture. But solutions are already there.

1.1. What is fog/edge computing

The concept of fog computing first appeared in 2012, coined by Cisco as an extension of the cloud computing paradigm (i.e. Platform-as-a-Service (PaaS)) down to the edge of the network [4]. The main rationale behind fog computing is rooted on the benefits brought in by deploying IT capacities close to both data sources and users, in terms of low latency, security, optimal performance and proximity (geographic distribution), to existing but also innovative services yet to come. For instance, a wind farm [5] can benefit from this platform to optimize the operation thanks to the low and predictable latency between turbines, data collection from source or near sensors, and in-place actions.

Previous to fog computing, there were two similar areas that also use the resources at the edge of the network to provide new services and are worth to mention: Sensor Networks and Vehicular Networks.

Motivated by the unstoppable evolution in electronics, it is currently possible to create small low-cost low-power sensors capable of collecting data and sending the data to another device. Sensor Networks [6] are groups of large number of sensors interconnected between them in a non-stable network topology, prone to failures and very limited on their capabilities. These platforms allow to deploy a large amount of sensors preconfigured to sense and send data across the entire platform to a point where can be stored, referred to as *sink*. An illustrative example is shown in Fig. 1.

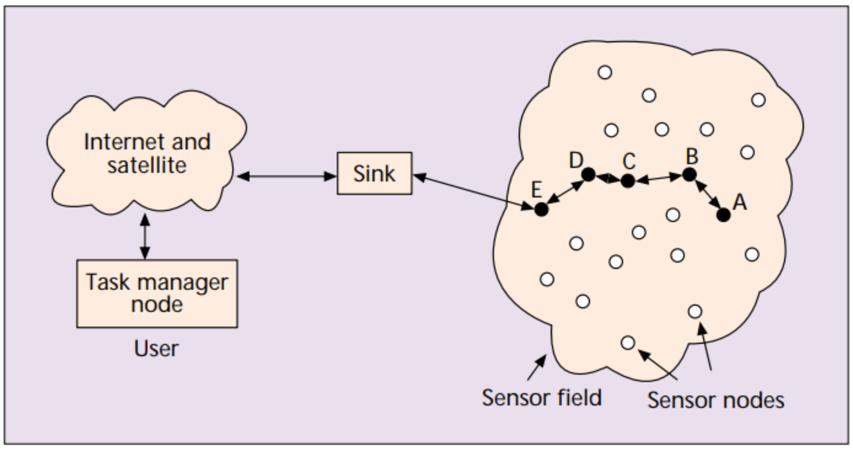


Fig. 1 Sensor network architecture example[1]

One difference with fog computing is that Sensor Networks' primary goal is to provide a platform to collect data from cheap, easy to deploy sensors without a heavy setup. On the other hand, fog computing is concerned on the data processing and storage required to execute services, leveraging the data collected by the Sensor Network.

Vehicular networks, or Vehicle-to-Vehicle (V2V) communications, share most of the challenges and objectives of fog computing, but oriented on car driving applications [7]. For instance, availability, mobility or security are some of the main challenges to solve. One illustrative use case is the cooperative collision warning [8] along vehicles on the road, improving safety on the roads. Here the service is executed on each car and the different actions to be done are transmitted to other cars directly.

The next step in V2V communications leverages 5G technologies turning into the Vehicle-to-Everything (V2X) [9]. V2X is the connection of the vehicle to any other capable device or "thing", such as pedestrians, infrastructures, or other vehicles, that can sense real-time information to be transmitted to the vehicle.

It is worth mentioning the similarities between fog computing and edge computing. Existing similarities lead to the use of the two paradigms to describe similar concepts. There are some efforts discussing about the similarities and differences between them both (see Cisco report in [10], describing fog computing as the standard and edge computing as the concept, the definitions proposed by the OpenFog Consortium in [11] or the discussion in [12]). In the scope of this project, both terms are used to define the concept of bringing intelligence to the extreme of the network, also known as edge, and use all the available resources to collect data and execute services.

And is with this definition that we can see the main differences from all the other technologies. Edge computing does not care about the communication standards between devices or specific applications, but provide a general infrastructure using the available and closer resources to execute nonspecific applications or services.

[1] Figure extracted from [6]

However, even cloud computing and edge computing are different paradigms that does not work under the same context, the use of both opens up the possibility to a new scenario where edge computing can be more extensible and mobile.

1.2. Fog-to-Cloud Paradigm

The use of cloud computing in connection with edge computing coins the definition of the Fog-to-Cloud (F2C) concept [13]. The proposed F2C architecture consists in the slicing of the network into layers. Each layer is formed by devices with similar characteristics or by proximity [14]. Moreover, in each layer, there are also divisions of blocks called Areas or clusters, that are managed by some controlled element.

In Fig. 2, an illustrative example of a F2C scenario is provided where, on top of everything, there is the cloud datacenter that connects to all other areas on the layer bellow. On the left part of the figure, we have a city scenario with high capable and static devices, such as buildings or factories and on the right side there is an airport. Each scenario has its peculiarities, like the amount of mobility expected or the type of services and data.

For each area, it is necessary a controller element responsible for coordinating and managing the cluster. The control communications follow a hierarchical structure, even inside each area. Controllers are the element at the top of each area with the exception of the cloud that is at the bottom of the upper layer (i.e. layer 0), allowing to have multiple datacenters as the cloud computing allows.

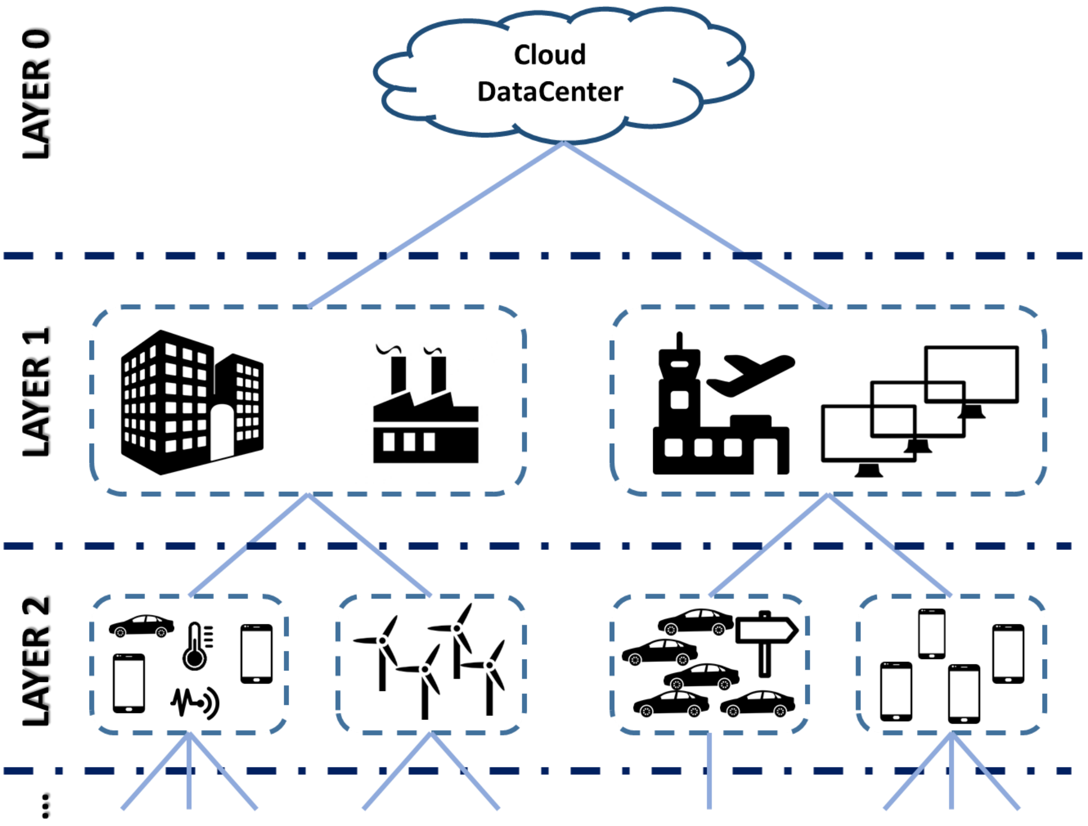


Fig. 2 A F2C scenario and architecture example

The F2C architecture does not specify a limit on the number of layers. Conversely, for each added layer, the granularity of the system increases, leading to execute a service in multiple areas and, at the end, using a superior layer for the coordination. As the goal is to maximize Quality of Service (QoS) to be able to execute low-latency real-time services, it is necessary to reduce at its maximum all the delay that is possible.

Edge computing solves some of the problems that cloud actually has, but there are still challenges that cannot overcome. For instance, the mobility feature, a key aspect of the edge context, leads to the volatility of the infrastructure and constant changes that make the execution of a distributed service a laborious task.

Another analogous issue is the reliability of the system in such scenario. The topology may be changing continuously, meaning that devices are constantly entering or leaving the system in an unpredictable or non-deterministic way. Battery consumption, lack of connectivity, or device shutdown are some of the causes.

Controller elements need to be reliable to avoid the loss of control and the system need the sufficient mechanisms to assure that, in case of failure, there is a process that can recover the control.

1.3. Real Time Services

5G technology [15], proposed on 2013 – 2014, is an evolution of the actual 3G/4G mobile technologies, to address the current challenges on that field and the increasing amount of mobile devices. 5G is designed to provide communication transportation in scenarios with hundreds of machines per base station, increase of data rate by several orders of magnitude, reduction of delay, lower energy consumption, etc.

The deployment of 5G infrastructure is planned to be done beyond 2020, in some places even before. For instance, the European Union has proposed that all the state members deploy 5G networks towards a large scale introduction on 2020 and align all the roadmaps to fulfil this objective [16].

Operators, companies and institutions are already proposing use cases where this technology is a key feature, and most of them are related to the real-time concept. As an example, the Government of Catalonia at the 2019 Mobile World Congress Edition presented the “Ambulance of the Future” [17], a connected ambulance that uses the 5G technology. It has two cameras that stream the image directly to the hospital where the doctor can advise the professionals at the ambulance.

Related to the F2C scenario, there is a huge need on provide top-notch service allocation algorithms to reduce to its minimum all the infrastructural delays and focus only on time execution [18]. There are also other concerns such as the localization of the data and IoTs, to decrease communication delays and network usage. Different and adaptive strategies can reduce the overall time and making possible to be deployed with a better behavior.

More services will appear, requiring to be executed in a distributed fashion way, closer to the data source or receiver, non-prone to failures and connected to multiple devices on different locations.

1.4. Problem Definition and Motivation

As the 5G technology will provide a large set of different devices and IoTs interconnected with faster and ubiquitous networks, edge computing and F2C solutions will take an important role in service deployment.

Among all the different challenges that F2C architecture proposes to solve, mobility and reliability of the system are important ones, as fit pretty well on the present paradigm. In terms of mobility, there are different issues and necessary mechanisms to be provided. It also happens with the reliability. In specific, the coordination of the areas supposes one of the big issues to answer in terms of protection and dynamicity.

To make the F2C system reliable and preserve the mobility feature, there is a need to provide different sort of mechanisms that provide protection procedures in case of failure. Moreover, due to the changing context of the devices, there is also a need to create recovery procedures. These two concepts can be merged into one called resilience.

The formal definition of the resilience concept is "*the ability of a substance to return to its usual shape after being bent, stretched, or pressed*"[2]. Applied to the project domain, it refers to the ability of the area or the system to return to a stable state and working as nothing has happened. Archiving this propriety in a non-stable heterogeneous system is a great challenge, but so are the benefits.

The lack of existent deployed solutions in this specific context is a great opportunity to contribute with some proposals and put another brick in the creation of this architecture. Specifically, the lack of consolidated solutions that provide resilience on F2C scenarios in a dynamic and customizable way. By accomplishing this, the F2C architecture can increase the overall QoS and be even more suitable for critical real-time services such as emergency management operations or self-driven cars. The goal of this project is to address this specific problem and develop a solution that can be applied in such scenarios, focusing on the provision of resilience of the areas, and the rules that manage the overall system.

1.4. Structure of the Project

In accordance to the previous explained, the project is focused on developing a theoretical proposal and a proof-of-concept solution (as explained in detail in the objectives at the next section). The project, structured accordingly, follows the following schema:

The structure is divided in four main chunks. First, a presentation of the objectives and the state of the art of the scenario is presented in sections 2 and 3. Second, the overall project context and proposed architecture, provided in section 4. Third, the main designed components of the project: Leader Protection and Policies, in sections 5, and 6 respectively. Fourth, the description of the solution implementation and the provided validation, presented on sections 7 and 8. Finally, some further extensions and conclusions of the project are provided on sections 9 and 10.

[2] Definition of "resilience" from the Cambridge Advanced Learner's Dictionary & Thesaurus © Cambridge University Press

02.OBJECTIVES

One key aspect in a F2C scenario is the control of the different areas, following a hierarchical control plane structure, where each area has a node that coordinates all the control messages. F2C scenarios are designed to allow real-time services, that require high resilience level and support the mobility of the devices.

The scope of the project is to design and develop an application or module that allows the management of the resilience with protection and recovery mechanisms in the F2C context, especially focused on the edge. The design of the application should provide a clean and extensible architecture, compliant with the F2C paradigm, while the development of the application should be a proof-of-concept of the proposed design, with the key features implemented and validated.

Another point that is related with the protection is the election of the control elements in each area or cluster and finally the ability to spread rules or policies into the system.

To validate this project, besides the testing and validation of the standalone features of the application, one of the objectives is to design and integrate the result into a real F2C application or device. The mF2C European project, explained in the next section, is the selected use case to integrate the result and validate that the project works in a real F2C application.

In summary, the list of objectives of this project is the following:

- Designing and implementing a closed solution that provides resilience over a F2C architecture, with a main focus on the edge. This solution should work as a standalone module that can be used without other applications or be integrated.

- Providing the design and proof-of-concept implementation of protection mechanisms for the controller device, also called Leader.

- Providing the design and proof-of-concept implementation of the different policies, related with the resilience of the F2C scenario.

- Validating the project into a real F2C application, integrating the solution and fulfilling the expected functionalities.

03.STATE OF THE ART

Distributed systems models are diverse on architecture and points of view. Some models focus on the data distribution while others on distributed execution. The F2C paradigm aims to execute services in a distributed and reliable way, on a heterogeneous and mobile scenario.

To understand better the context and some related technologies, this section gives some of the background where the project is based.

3.1. Fog-to-Cloud

Nowadays, F2C is a proposal yet in a development state, but still there is not an extended commercial version. Some institutions and companies are founding projects and research initiatives to create a F2C application or system and try to bring closer companies that adopt this technology.

For instance, the European Union in the frame of the Horizon 2020 research founding program, has a budget of around 3.000 million euros on the Information and Communication Technologies area [19], destined on projects oriented on 5G technologies, improving the digital market, artificial intelligence, cloud computing, IoT, big data, digital transformation, etc...

More related to the F2C scenario, there are two projects that are worth to mention and analyze, that define the F2C application and scenario architecture.

3.1.1. mF2C

"*Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem*", often named the **mF2C project**[3], is a European Project founded under the Horizon 2020 framework [20]. The goal of the project, inside the F2C context, is to design 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*". Aside from the design, there is also the objective to develop a proof-of-work system and platform, tested and validated with real-life use cases, and to set the groundwork for a distributed system architecture.

This project started on early 2017 and is planned to be finished on end 2019. As an ongoing project, there are still some work in progress and newer definitions, but the main architecture is completely defined.

The mF2C proposed scenario [21] [22] is a layered hierarchical structure from the cloud to the edge, where in each layer the devices participating on the system are clustered into areas and managed from another device in a superior layer. All the devices with the mF2C application installed are called Agents, that can have different roles. For instance, a node managing an area is acting as the leader of that area, so it can be called Leader Agent or Leader. A graphical example of the proposed architecture is showed in Fig. 3, where all the blue boxes represent Agents with different roles depending on the location. The backup is defined as the protection

[3] Official Website: https://www.mf2c-project.eu/

element of the Leader, and the microagent is a reduced light version of the agent with less capabilities and functionalities.

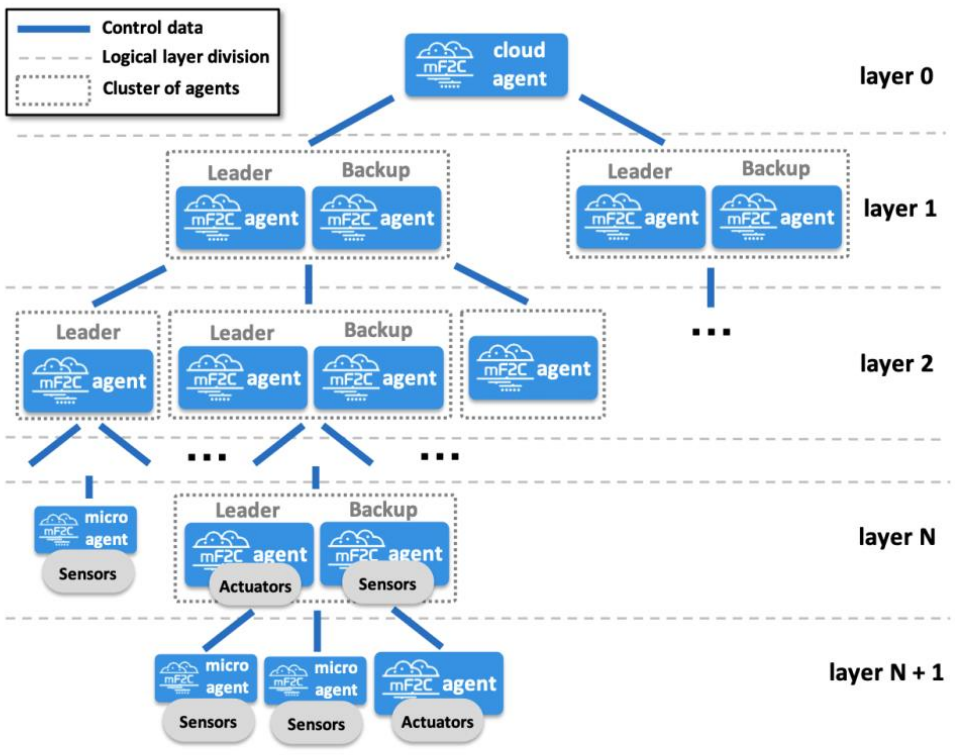


Fig. 3 The mF2C architecture

The internal architecture of the mF2C Agent is sub-divided into two big blocks: The Agent Controller, in charge of the control of the device itself and the agents bellow due to the hierarchical view of the system; and the Platform Manager, in charge of the logic of the system regarding the service execution and orchestration. A more detailed schema of the architecture is presented in Fig. 4.

The Agent architecture follows a modular approach, divided in blocks depending on the responsibility. As transversal components, the agent interface and the security are present in all the design.

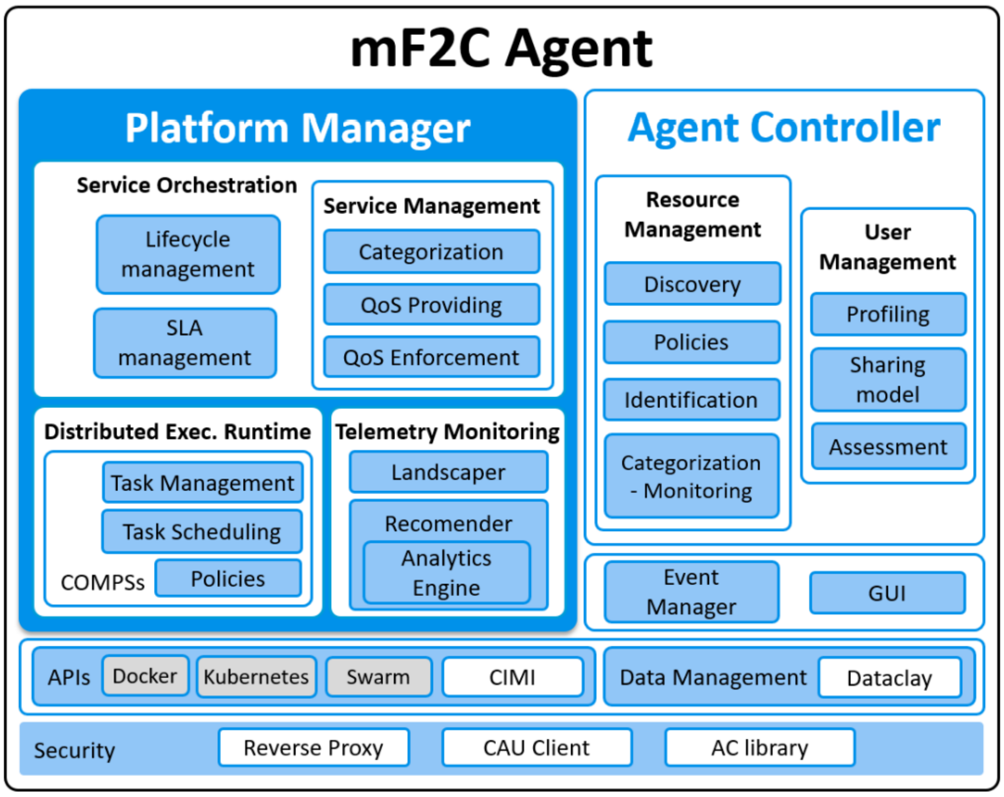


Fig. 4 mF2C Agent architecture

The mF2C Agent is an open-source application and can be downloaded[4] and used by anyone. Moreover, the agent is designed to work as a Docker service, making easier the deployment and setting up the scenario.

3.1.2. OpenFog Reference Architecture

The OpenFog Consortium[5] (recently integrated into the Industrial Internet Forum[6]) is an organization formed by different organizations and companies, some of them key on the definition of the fog computing. The main aim is to create "*an open reference architecture for fog computing, build operational models and testbeds, define and advance technology, educate the market and promote business development through a thriving OpenFog ecosystem*". In other words, the focus is to provide an open framework that defines what fog computing should be and the interaction with the cloud [23].

In specific, this synchronization between the fog and the cloud on services, called cross-fog applications, follow the F2C principle of using both of the best benefits that edge and cloud can provide on the system.

The resultant architecture from this board of members is called the OpenFog Reference Architecture (OFRA). This architecture follows a set of principles or so-called *pillars* that set the

[4] Github Repository: https://github.com/mF2C/mF2C
[5] OpenFog Official Website: https://www.openfogconsortium.org/
[6] Industrial IOnternet Forum: https://www.iiconsortium.org/cambridge/index.htm

general lines of the design of the architecture. In Fig. 5, an example of a possible type of scenario is provided.

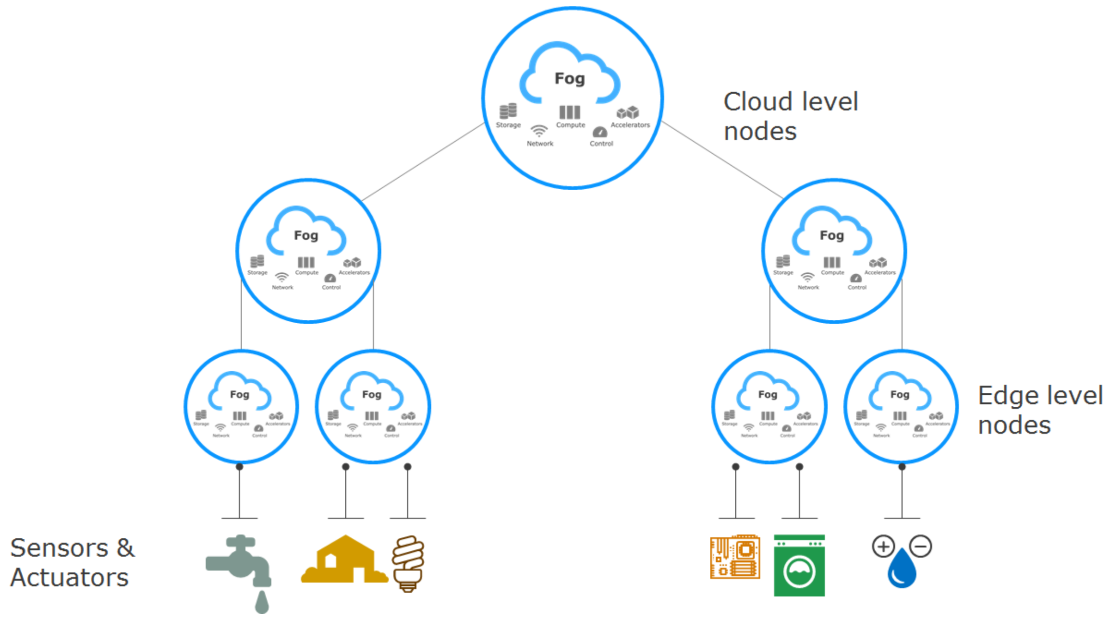


Fig. 5 The OFRA scenario architecture example

The scenario follows the F2C slicing and hierarchical principle, with the nodes at the bottom near to the data sources and the higher ones closer to the cloud. Another interesting point is that, as the mF2C, the OFRA proposes the deployment of the same application in all the nodes of the architecture.

Without going any further, as it is not the propose of this section or project to make a research work on the proposals of both projects, it's easy to see that there is a trend on both projects on how the F2C should be developed. Both follow the same approach and scenario architecture, with some variation on the internal architecture and different points of view on applications and hardware involvement.

However, it is clear that mobility and reliability challenges are an important issue to solve and both projects have some components or ideas, where the solution of this project can bring benefits.

3.2. Resilience in Distributed Systems

The project focuses on designing an innovative solution aimed at providing resilience leveraging a novel architecture rather than integrating an existing one, given that not many solutions exist on this context. Alternatively, there are similar scenarios or solutions that can be used as a reference for the design proposal of the project.

On this subsection, there are two illustrative cases related with the resilience on distributed systems.

3.2.2. Controller Election

The election of the controller in peer-to-peer (P2P) networks is a research topic from long time ago. Innumerable amount of proposals have been submitted on different collaborative ways to elect a controller from a group of interconnected nodes.

For instance, some of the most influential algorithms, Bully Algorithm and Token Ring Algorithm [24], are based on the communication between nodes following a specific procedure. Accordingly, all nodes participate on this election process to archive a consensus and are equally electable.

However, in a F2C scenario, this type of election is not possible. As the architecture is defined to be hierarchical on the control messages between devices, if the controller element is not present, there is no possibility to establish a communication. Moreover, controller elements may be defined by the system manager and not all devices may be suitable for that duty.

Some of the election algorithms have rules on how the election is done (e.g. the device with the higher ID is the coordinator) or some other discretionary mechanism to avoid conflict.

A new mechanism should be designed to fit into the F2C model, to provide election of a Leader and in case of failure, the selection of a newer one without conflicts. Some lessons can be applied from these algorithms and used.

3.2.3. Zookeeper

Zookeeper [25] is a commercial solution that provides high reliable distributed coordination among a set of devices. The architecture consists on a set of nodes (or servers in ZooKeeper notation) that are reachable and clients can make requests. Data is stored in an atomic and reliable way along all the other active servers and the load requests are automatically balanced.

This solution is especially popular in cloud environments, as services does not need to concern about the race condition or high availability [26]. To see an example, in Fig. 6 there is a system working with ZooKeeper, with five servers clustered and eight clients making requests spread among all the servers.

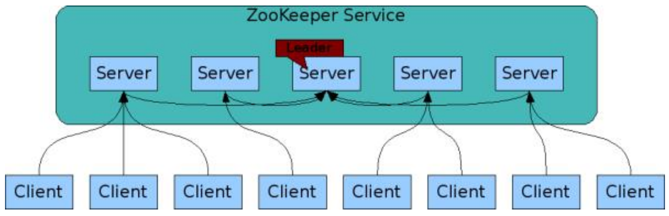


Fig. 6 ZooKeeper architecture example[7]

ZooKeeper provides an interface to facilitate the entry point of the clients, the atomic data management and load balancing of the nodes. However, it requires all the servers to create a cluster and share all the information.

The implications on the F2C architecture, especially at the edge, are that more than one device shares all the information implying that each area have more than one Leader that can attend requests from Agents. Another counterpart is the loss of specific control over the Leader selection process and other modules involved. However, the design of ZooKeeper and its use on the cloud cannot be discarded.

[7] Original picture from [26]

04.GLOBAL FUNCTIONAL BLOCKS DESIGN

One of the first things to do on the project is the design itself. Previous to the protection mechanisms and the policies definitions, it is important to understand the scenario and the functional architecture of the project. In this section, first the scenario is presented, followed by the design challenges to face and finally presenting the proposed architecture with its functional blocks.

4.1. System Architecture

Before the definition of the functional blocks, it is necessary to define a scenario. As F2C is the concept and only stablishes some guidelines, it is important to specify the architecture of the system that the project is using. Although, the design of the project does not rely on a specific architecture, as it should be transparent on any specific scenario configuration.

For this project, the scenario uses as reference the mF2C architecture (see section 3.1.1. mF2C) for its definition, but with some modifications. Specifically, the scenario only has three layers: Cloud layer, Leader layer and edge devices or normal agents layer. Regarding the backups, the proposed design allows to set dynamically the amount of active ones, but in normal conditions the project will use only one as the main backup.

The control element, called *Leader* in our F2C scenario, has similar functionalities and responsibilities as the Coordinator or Cluster Head in distributed systems or clustering. On the context of the project, these names are used to describe the device in charge of the management of an area and their use should be understood as a synonym. Also, the Backup is the device that provides protection of the area (specified with more detail in the design of the Leader Protection in next section), not to be confused with a literal data replica or database. The terminology used has the same meaning as the one at the mF2C project.

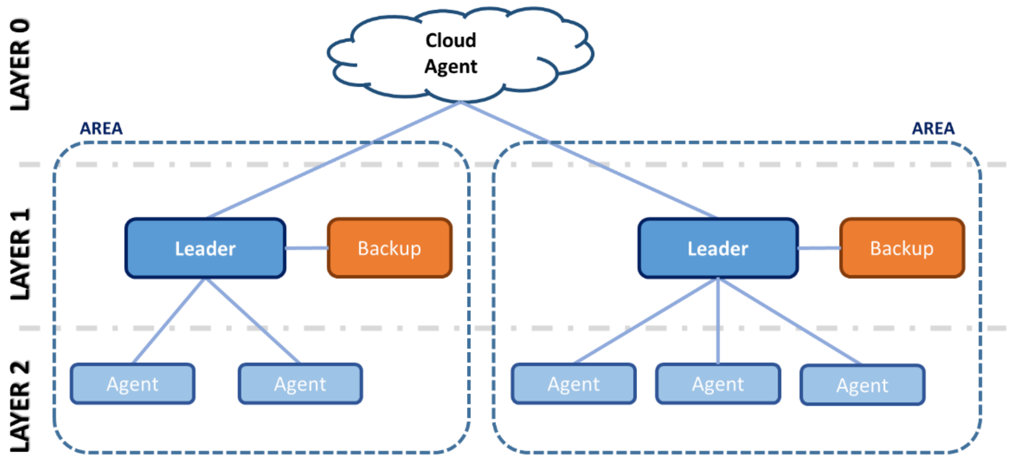


Fig. 7 An architecture example of the project

To better understand, a graphical architectural example is provided in Fig. 7. The cloud, at the upper layer, controls and synchronizes all the leaders between areas. On the second layer, Leaders manage the area and a Backup is active. Finally, on the bottom layer, the rest of agents of the system.

The scope of the project is focused on the reliability of the edge part of the architecture. That means that the cloud is not crucial for the design as it already has its own resilience mechanisms. This architecture can scale horizontally as many devices and areas the system manager wants to create.

To setup this architecture, the system only needs a Leader to start clustering devices into an area.

Leader Election

The election of a device capable to manage all the control messages in the area is an important step. In fact, before protecting the control of the cluster or area, we need first to assure that the control element that is in charge is suitable for that duty.

The resilience of the area and the protection of the Leader, explained in Section 5, require the leader election to select a leader or a backup inside the protection and reelection mechanisms of the leader.

Leader Responsibilities:

The main finality expected for a F2C system is the execution of services with an improvement of the performance and using the resources at the edge. The coordination elements for each area are expected to manage, execute and return the result of service requests.

As the service execution is the primary goal, all the devices in the system are expected to participate on this procedure. Focusing on the Leader, there are some added responsibilities:

- **Service Management and Orchestration**: The leader must coordinate the execution of a service distributed in more than one device in its area, as a consequence of the lack of horizontal communication between devices of the area at the deployment and finalization phases.
- **Resource Mapping**: In order to send services to be executed, the leader must know the actual state of the area, the available resources and the number and location of services executing.
- **Resource Management**: The management of the area can be done exclusively from the Leader. The Leader has to maintain the topology of the area, decide who is allowed to be part of, the policies applied to, and other management tasks.
- **Resource Aggregation:** The resources and data observed from above should be like a big orb of combined information about the available assets.
- **SLA and Monitoring:** Check that the service is behaving correctly and being compliant with the service terms.
- **Security:** As the Leader controls the access of the devices and service executions, it can directly analyze to check for suspicious behaviors and block devices from the area.

Leader Requirements

To fulfill all the responsibilities, a Leader requires a sufficient amount of available resources. For each scenario, the set of requirements may be different. Moreover, the edge paradigm describes such scenarios as heterogeneous and changing over the time, making inviable the static default setting of policies or requirements.

The evaluation of the required amount of resources for the Leader depends on the specific implementation of the system and is out of the scope of this project. However, there are two main group of requirements that can be modeled, incorporated into the following policies:

- **Leader Mandatory Requirements (LMR):** Set of requirements defined in the system that a device needs to fulfill in order to be eligible as Leader.
- **Leader Discretionary Requirements (LDR):** Set of requirements defined in the system that give a score to the device to sort along all other devices.

This two set of requirements can include a set of different characteristics such as CPU usage over time, virtual memory available, storage capacity, mobility, remaining battery, etc.

LMR and LDR are included into the **Leader Election Policies (LEP)**, defined in Section 6.

Selection Process

The selection process is the mechanism that implements the leader election defined by the current policies of the system. This procedure is divided into two main categories:

- **Selection at the device startup**: Each time the device starts the application, the algorithm tries to find a leader or become one, following the Leader Election Policies.
- **Selection at running time**: If the leader fails or the leader is no more suitable, a new leader is selected. These mechanisms are defined in Section 5.

4.2. Design Challenges

To design the internal architecture of the project, there are some challenges that are present on the whole design process, which are worth to specify.

- First, the design should be modular and isolated from other components. The provided architecture must be a closed block that can run standalone, with its own required mechanisms to provide the expected functionality.
- Second, the design should be able to be integrated into the mF2C agent using the same architecture. A clean and specific interface must be present into the architecture to facilitate this task.
- Third, the design should be scalable to allow added functionalities, especially on the policies definition to allow new specifications.
- Fourth, related to the context proprieties, the design should work in a mobile and non-stable scenario.

The required main functionalities of the project are the resilience of the system and the spread of policies. The resultant design should provide solutions for both finalities and overcome the specified challenges.

4.3. Functional Blocks Architecture

As one of the objectives of the project is to provide a closed solution, one first step before the individual design of the components is to present the full picture of the project internal architecture. That includes how the different functional blocks are placed together inside the solution.

The **Control Resilience Management (CRM)** application or module is the resultant designed architecture of the project. The design follows a modular schema based on the expected responsibilities and functionalities. The CRM is designed to be used as a standalone application that provides resilience and policies distribution along a set of specific devices and scenario, or as an integrated module inside another application that provides some or all the expected functionalities to the system.

The overall design consists in four principal sub-modules, the core, and the API. A graphical representation of the internal architecture can be found in Fig. 8.

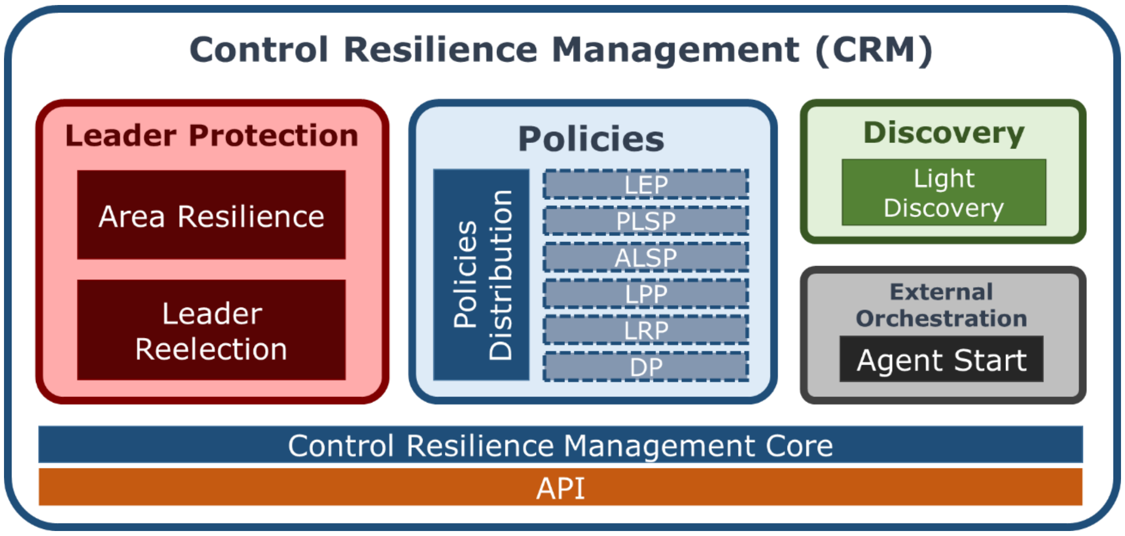


Fig. 8 Control Resilience Management design architecture

Each sub-module has a set of different components, isolating the specific responsibilities into smaller pieces. The four sub-modules are:

- **Leader Protection:** Contains all the mechanisms in charge of the area resilience and the leader reelection.
- **Policies:** Contains the policies distribution mechanism and all the defined group of policies of the system.
- **Discovery**: In charge of the topology management and acquisition.
- **External Orchestration**: Orchestration and integration between other external modules that require coordination or actions from the CRM module. This sub-module allows the integration of the CRM into other systems.

The design and implementation of the sub-modules is explained with more detail in the following sections. In specific, the Leader Protection and the Policies proposed design and functionalities due to the complexity inside of them, are explained in separated sections.

All the sub-modules are designed following the isolation principle, meaning that are completely independent and can be substituted for others if the interface specification is followed. The CRM itself follows the same principle, making possible this dual-behavior between standalone application or module. The API specification is designed to act as a frontend for the system manager to set-up the configurations or the interface of other modules to the CRM integration.

Discovery

The discovery is a sub-module in charge of the provision of the topology. This module makes possible the deployment of the CRM without the manual insertion of the devices that conform the cluster.

The discovery protocol is based on the broadcasting of messages over the current attached network and the reply of the agents that receive the messages to the Leader with the own resource information embedded. Each time that the Leader receives a message, the topological database stores or updates the device.

The approach used on the design of this protocol is a simplified procedure of [27] without the use of the 802.11 protocol. This makes that only devices that are connected to the same network as the Leader can be discovered.

This sub-module can be deactivated if other external mechanism supplies the CRM with the topology information. For instance, the mF2C has a module that already provides a better way of discovering agents.

External Orchestration

The External Orchestration is the sub-module in charge of the integration and coordination of the CRM into another application as a module. This means that the design and implementation of this sub-module is specific for each application and added functionalities.

For instance, for the integration with the mF2C, the CRM substitutes the Policies block of the mF2C architecture, as shown in Fig. 9.

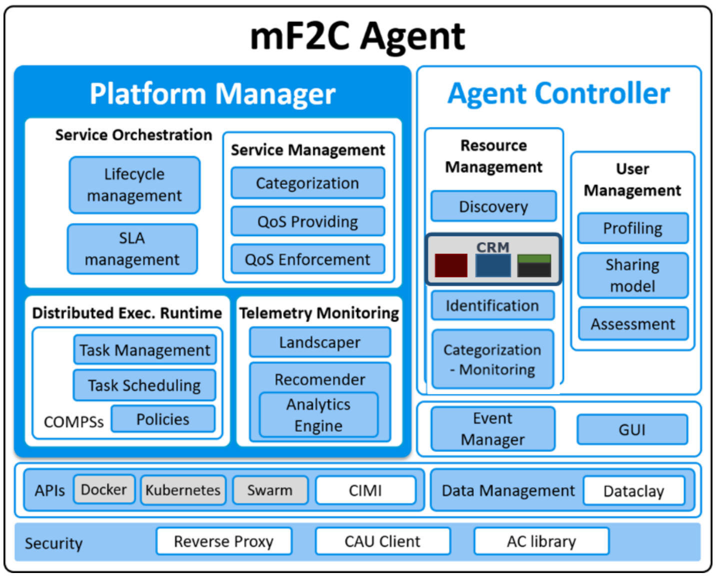


Fig. 9 Integration of the CRM into the mF2C Agent

The integration of the CRM into the mF2C, explained in detail on section 7.6. Illustrative Use Case: mF2C Integration, implies that for instance, the coordination of the startup procedure and the role control of the agent is performed by this sub-module inside the CRM, as mF2C defined these responsibilities in the architecture.

The design of the Agent Start module, illustrated in Fig. 10, follows the unique code with different functionalities principle of the CRM general design. Modules inside the mF2C application require different triggers depending the actual role.

The flow starts requesting the Identification module to generate the unique deviceID of the Agent. Once the information is received, the Passive Leader Promotion (PLP) mechanism decides if the device is set to start as Leader or normal Agent. If PLP is not set or result in a normal start, the Automatic Leader Promotion (ALP) starts. ALP determine that if a Leader is not successfully discovered in a defined number of attempts, the device if capable becomes a Leader. Once the Leader of the area is detected, the agent proceeds with some authentication steps and finally starts the Categorization module, the Area Resilience module (located inside the CRM) and other required interactions with the mF2C API.

If the device has become a Leader, a set of specific procedures must be done, including the switch of the Discovery module into Beaconing mode and Leader Authentication procedures with the mF2C architecture.

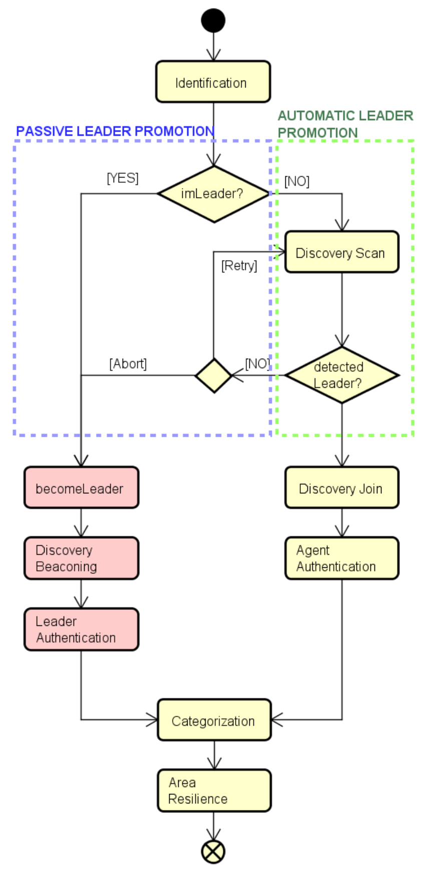


Fig. 10 Agent Start flow design

In case of a Leader failure, promotion or reelection, the Agent Start flow is designed to be executed many times. The PLP and ALP mechanisms with the provided information and policies are able to decide the role of the Agent.

Finally, as the design is specific for each project, there are only some guidelines specified for that sub-module in order to keep the design principles:

- This sub-module shall not interfere with the other existent modules in a way that can make crash the correct working of the application.
- The sub-module has to be isolated from the others and it must provide the specific interfaces to interact with.
- This sub-module, like the others, is under the interface of the CRM API and its core, and it cannot define any new external interface.
- If there is a need to interact with other modules, the trigger should be sent to the API rather than to the sub-module itself.
- API specifications can be changed to cover new required messages and integration with the API.

05.DESIGN OF A LEADER PROTECTION STRATEGY

Protecting the Leader in a F2C scenario represents the protection of all the areas bellow the cloud, as the Leader is the control element for each one of them. Cloud already has its own developed and successful mechanisms to assure a high availability and resource escalation to satisfy the requests. Edge is completely different from cloud, as it is a heterogeneous scenario with flexible availability, requiring new mechanisms to protect such control elements in a practical, faster, dynamic and reliable way.

For each area at the edge, a coordinator device is needed. For real-time services, a high-availability and high-reliability are key features. For instance, services concerning emergency scenarios such as catastrophe management, terrorism threats, climatic events, environmental hazards… have some common points: high amount of devices at the edge that have data to send and receive, services that need that data and to be executed on the available and near resources.

For example, the OpenFog Consortium (see section 3.1.2. OpenFog Reference A), proposes as one of their use cases the healthcare patient monitoring [28]. Using the data proximity and scalability agility edge features, real-time services can benefit from and increase the overall performance. It’s easy to see why is important to protect the area of this scenario.

One critical part to protect the leader is the election itself, as an awful decision could lead to a malfunction of the area and the loss of control. However, even if we are able to predict and select the best suitable device for that duty, devices are mobile.

Mobility is one of the causes that make the edge dynamic and changes the topology continuously. Another factor is energy consumption and non-plugged devices, leading to disconnections of devices while services are executing.

For this and other reasons, the edge needs some mechanisms that safeguard the control of the area by the device that has the control over it in runtime. This project proposes two main mechanisms that suites that definition: The protection of the area by adding devices that oversee the correct behavior of the Leader and that can, at any given time, overtake the responsibility when an error occurs; and the replacement of the Leader to prevent error and replace the current control element with one more suitable.

This section explains in depth this two different and coexistent mechanisms. In section 7.2, the implementation of them is explained in detail.

5.1. Area Resilience

Resilience in the project context refers to the ability to keep the system working in the event of the loss of a controller element. Especially, the resilience focuses on events related to malfunctions and unexpected behaviors. For example, loss of connectivity, out of battery, unexpected crash…

The resilience of the area depends on the reliability of the Leader and the capability of the system to keep a functional Leader coordinating the cluster. As the context is the execution of time sensitive services, is very important to prepare the scenario for this kind of events, or in other words, implement a proactive strategy instead of a reactive one.

On the reactive approach, electing a new Leader when a failure is detected requires an enormous amount of time, with the inconvenience of synchronizing all the devices along them with different detection periods. F2C aims to create a hierarchical architecture, where all the control communications are through the Leader, following a vertical line from the cloud to the bottom of the edge. This approach leads to a non-scalable amount of connections for each device (most of them are expected to have minimal computing power), more network messages and power consumption.

On the other hand, a proactive approach can consist on a “backup” device that will overtake the Leader responsibilities in case of a failure detection. Other devices will detect that a new device is now the Leader of the area and only proceed with the connection procedures. This mechanism can be repeated anytime if there are devices capable to be “backups”.

Given the benefits and the type of scenario, the approach selected for this project is the proactive one. On the mF2C project, all the devices have the same mF2C application, but different responsibilities, or in other words, depending on the responsibility there is a different role. Following this approach, the “backup” is a role of the device in our system, like the Leader but with different duties.

5.1.1. Backup Election and Keepalive Protocol

The backup election is the process triggered by the Leader to select a capable device. The capable decision proceeds from the Leader Election Policies (LEP) specifications. The Leader is continuously self-checking if the number of active backups is satisfactory and selecting newer ones if is not enough.

The Backup Election starts with the acquisition of the topology. The topology is a list of devices that belong to the same area and is stored in each Leader. In addition, the topology can contain the available resources of the device or other categorization data, used to select the best backup.

The lack of resource information or related policies will fall into a random election. Although, if the scenario requires a most precise and limited election, the LEP policies and a Leader Selection may be used.

As a normal device (or Agent in the mF2C context), if by policies is considered a capable device, it waits until an election message is sent by the Leader. When transitioning to the backup state, some internal procedures may be triggered, as some components act on a different behavior depending on the role. Then the backup uses the self-defined Keepalive Protocol, explained bellow. As the backup checks the Leader, it gets also some information on the response from the Leader.

When a Leader is detected down by one of the backups, the one with a lower preference number starts the transition from backup to leader. If this procedure fails, the next backup with a lower preference number starts the procedure again, if there are more than one backup. Alternatively, the Automatic Leader Promotion (ALP) may elect a new Leader if any backup is not successful on the procedure, if the policy is setup.

The flow in Fig. 11 illustrate the design of the area resilience flow for all the devices. The monitorLeader and monitorBackup states are the ones when the Keepalive Protocol is in action, and are the core functionality of this module.

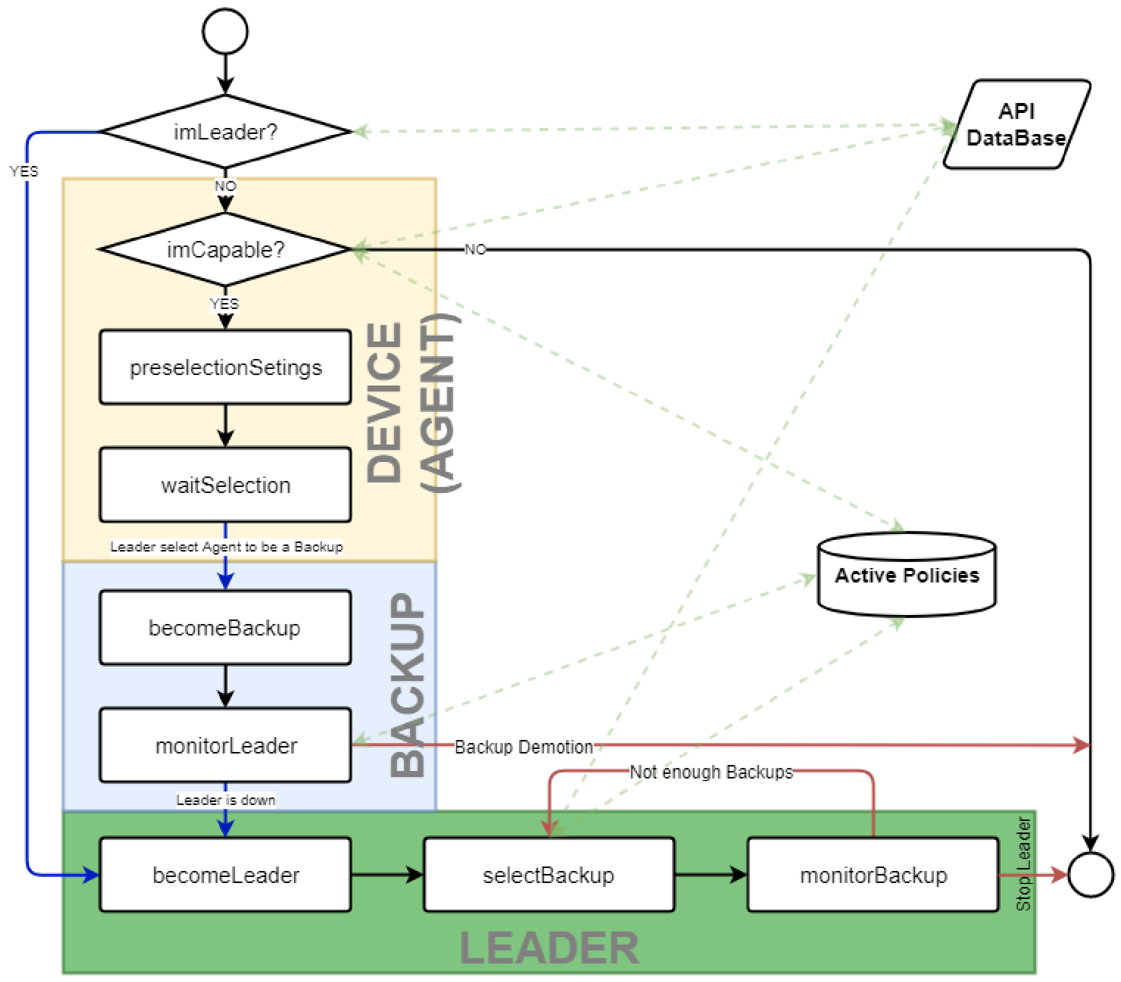


Fig. 11 Area Resilience Design Flow

The Keepalive Protocol is defined by a challenge message sent from the Backup to the Leader and a response back from the Leader. Defined by a policy, if a threshold is overcome, either the Leader or the Backup are considered out of the area and some action needs to be done. Those protocol messages need to be send periodically, where the period is defined by the policy.

The content of the challenge message must contain the identification of the device and may contain other information, as the protocol is designed to be extensible depending on the context. On the other side, the response message must contain the identification of the leader and the preference of the backup assigned by the leader and, same as the challenge message, it can be extended depending on the required functionality, as explained in next subsection.

Finally, the backups can be demoted, returning to the normal state (or normal Agent in mF2C terminology). This can happen at any time due to a reelection of the Leader (as explained in 5.2. Leader Reelection) or triggered by policy. In any case, the decision it can be only taken by the Leader or the device itself in case of disconnection or change of area.

5.1.2. Control Data Replication

As the new Leader is establishing, the topology information and services status are lost if no data backup or replication is made. This means that the Leader needs to regenerate all the information and services that are in execution are lost and relaunched. To minimize the overall time between a Leader failure and the area restoration, a control data replication strategy may be necessary in some scenarios.

Mainly, there are two ways of transmitting the information: Direct transfer between the leader and other selected devices or using a distributed database.

Direct Transfer

Direct transfer does not exclude the use of a database, but is designed to send specific information to a set of specific devices at some intervals of time, and managed by each device. The direct transfer is designed to work along the keepalive protocol, on the reply from the leader to the backup as a piggybank information.

This approach has four different strategies of data replication between the leader and the backups. Depending on the scenario and context, the impact of the performance and available resources may be factors to decide between one of them, as the preliminary study provided in 8.4. Dynamic Policies study shows.

Zero-Knowledge

Properly, the Zero-Knowledge (ZK) strategy does not send any control data to other devices. This strategy specifies that the backup regenerate all the control information from scratch and any executing services are lost.

To illustrate the architecture and procedure, a simple scenario is provided in Fig. 12. In basal state, the topology is already setup and the Leader has already a backup. The area information is stored into the Leader and updated each time that a change is detected, without any notification to the backup device.

On the second state of the figure, the Leader has failed and the Backup has already take the leader responsibilities, but without any area information. When the new Leader has the complete topology, it can reelect a new Backup and attend to service requests.

Finally, in the third state, the area is stable again and all the area information is coherent with the actual situation.

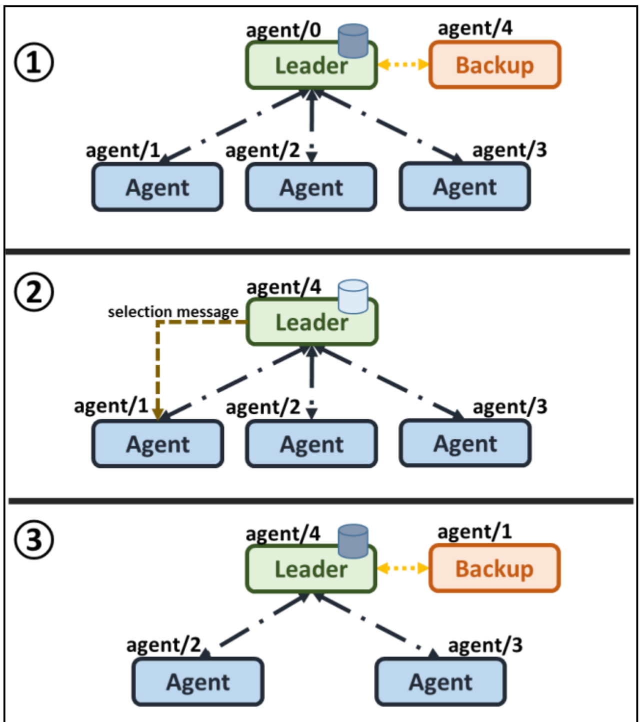


Fig. 12 Zero-Knowledge architecture and procedure.

In terms of design, this strategy is the most basic one, as only requires to restart the topology acquisition from scratch.

Keep-updating

The Keep-updating (KU) strategy synchronizes any detected change in the stored data with the backups. This sync interval is defined by the Leader Protection Policies (LPP). On leader failure, the Backup already has the last available information of the area, making possible to recover the state of the services and resume the activity. However, as there is a sync period, exists the possibility of discrepancies between the real state of the area and the stored state in the Backup.

The consequence may be the consideration of devices that are no longer present on the area or services that are not considered. In any case, those cases will eventually be revolved or discarded as the state of the area gets revaluated or services executions be discarded.

Using the same scenario than the previous strategy, in Fig. 13, the KU strategy is represented. The principal changes are the apparition in the Backup of the area information (represented with the blue cylinder) and the green arrow of the data transfer.

As the Backup already has all the area information, the selection can start immediately at the second stage. The area information is up to date at the third stage and synchronized with the Backup.

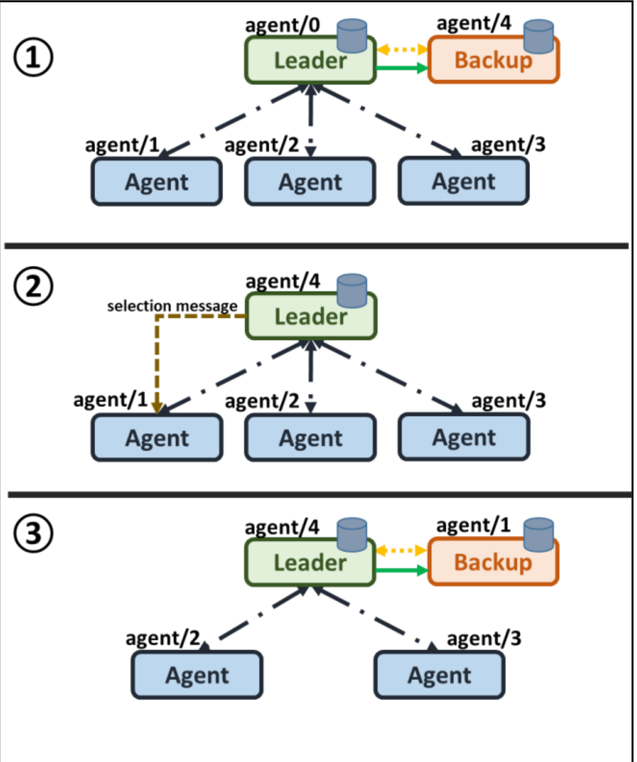


Fig. 13 Keep-Updating architecture and procedure.

External Keep-updating

The External Keep-Updating (EKU) or High-layer Download (HLD) strategy uses the same principle of the Keep-updating strategy, but outsourcing the storage of the information to an external device. This device is not necessary inside of the area or the system, more like an external repository.

In Fig. 14, with the same scenario proposed earlier, now the information is stored locally into the Leader and externally. When a change in the area is stored, a copy is sent to the external repository, with the same sync procedures previously explained.

When the Leader fails in the second stage, the Backup retrieves the stored information and select a new Backup of the area.

The same errors exposed in the previous strategy may occur. However, this strategy makes easy the synchronization in a scenario with multiple backups.

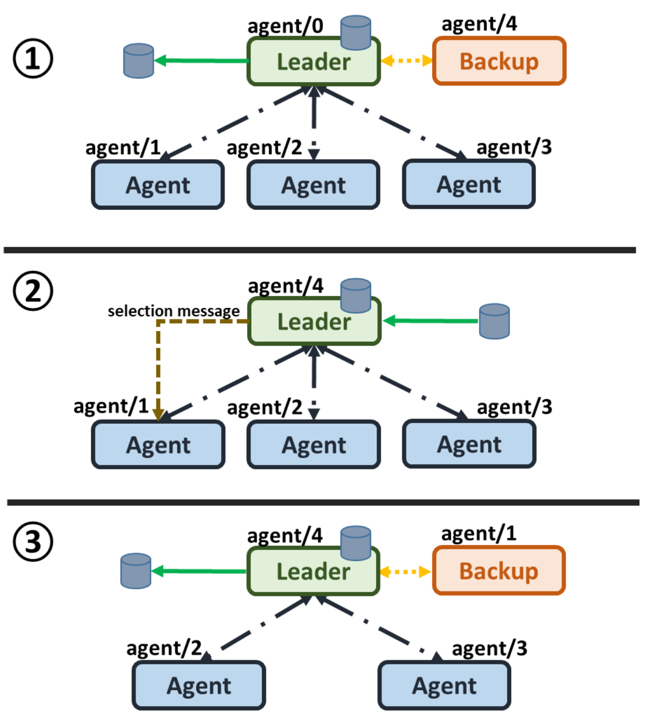


Fig. 14 External Keep-Updating architecture and procedure.

Cloud External Keep-updating

The Cloud External Keep-Updating (CEKU) strategy is a little extension where all the areas store their information into the cloud. Nevertheless, this strategy only uses the cloud as a repository, and the cloud should not access to that information to take any decision. The cloud vision must use the aggregated information of the areas and not their individual devices, as the Leaders of each area are the responsible of the resource management and service orchestration.

Distributed Database

Using a distributed database to synchronize all the data along all the nodes is another solution that can rely on trusted products already in the market. Using this mechanism, the area resilience component redistributes the responsibility of the control data transference to the database and rely on its own coherence and synchronization mechanisms.

For this project, there are two different options to be considered: A distributed Cassandra Database or the object oriented distributed data store Dataclay. However, there are other suitable options that are nowadays raising like the Apache Hadoop [29] for big-data distributed computing.

Cassandra Database

Cassandra [30] is a distributed database designed to manage large amounts of data and devices spread in location on different clusters or datacenters. The benefits of Cassandra are the performance enhancement, hierarchical namespaces, and high scalability and availability.

One use case that show the potential of this technology is presented in [30], where Facebook implemented a inbox search for the expected 100 millions of users and billions of requests, across all the globe.

Casandra is a good candidate for the F2C scenarios, as a tested and scalable solution. However, the Casandra structure does not support full mobility of their structures or transactions. Regarding the mobility, Cassandra is designed to operate in cloud computing environments, with a fixed set of datacenters and protection mechanisms in case of link failure and inconsistencies. In a scenario where devices are expected to have a high level of mobility, it can be exhaustive and consume a huge amount of network bandwidth.

Dataclay

Dataclay [31] is a distributed object oriented database, used in the mF2C project as the database for each agent. It provides the storage of data in classes with their methods and open management among different organizations.

The interesting property used on the F2C scenario is the synchronization between nodes, done in a hierarchical way. This property allows the control data synchronization between agents, transparent to the agent development and using the already secure and authenticated communications provided by the database.

On the mF2C integration of the project, Dataclay is not accessed directly from the modules, but instead using the CIMI interface, explained on Section 7.6. Illustrative Use Case: mF2C Integration.

5.2. Leader Reelection

Leader Reelection is another brick in the protection of the system and follows a proactive approach. As the Leader can be elected manually following the PLSP policies and the PLP mechanism, and automatically if ALSP and ALP are set in case of failure (see sections External Orchestration and Leader Selection Policies), a mechanism that can replace a Leader by other device without involving a failure or startup is necessary.

However, as the Leader is the coordinator element of the area, by definition should not be any other device except in the upper layer that can provoke a change of the current area rather than the Leader itself. This leads to the consequence that the only allowed device to make a Leader reelection is the Leader of the area. Still, the architecture can be modified to allow another behavior in some specific contexts. For instance, an area below another area (or a Leader managed by another Leader), the management of the election could be externally accomplished, being this especially useful on highly mobile scenarios.

Another aspect to take into account is the active backups of the area. As far as the Leader is demoted by the reelection of the new Leader, the Backups will promote to become a Leader as the resilience mechanism specifies. A previous demotion of the backups during the reelection procedure is necessary. Although, if the Leader does not have any active backup, in case of a reelection failure, the area may loss all the control until a new Leader arises.

In summary, the reelection mechanism should be a procedure that involves and is executed by the Leader of the affected area, allowing to receive the trigger from outside of the device. The backup should be notified and reelected as well but keep the resilience of the area in acceptable levels. At the end, the effect on the area should be minimal.

The designed reelection mechanisms use a trick to select a new Leader while having an active Backup. As the selected device has to become the Leader when the actual one is not acting anymore, the new device can use part of the resilience mechanisms to be promoted as one.

To graphically explain the designed procedure, a flow of the submodule is presented in Fig. 15.

First, the reelection starts with a proposed device to become the new Leader. As explained above, the receiver of such request is the Leader, which coordinates the full procedure until the demotion. The Leader checks if the device is already an active backup. If not, the device is promoted as one.

Second, the new backup is selected as the preferred one in case of Leader failure. Third, all the other backups that are active proceed to be demoted by the leader.

Finally, if all the procedure has been successful, the leader demotes itself to a normal agent and the new Leader be promoted and ready.

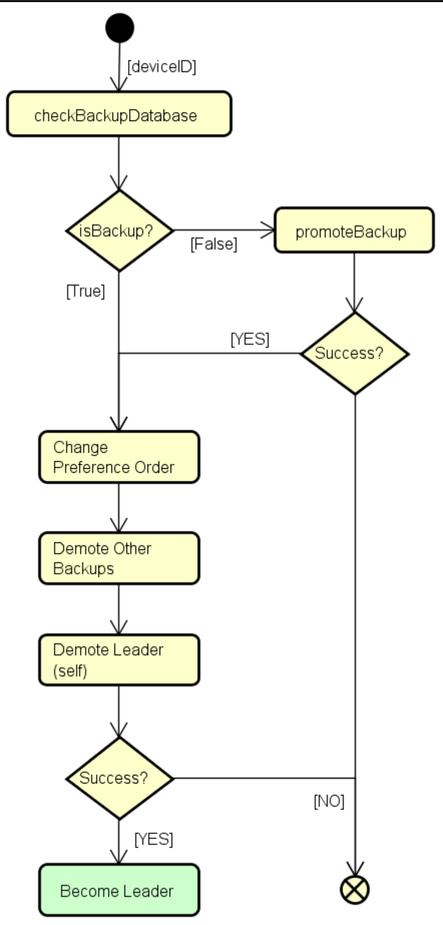


Fig. 15 Leader Reelection designed flow

The new reelected Leader will follow the normal startup or takeover procedure, as specified in Section 5.1. Area Resilience. If failure happens at any time, the resilience mechanisms assure that the area is protected. However, protection may fail in a critical zone between the demotion of the Leader, as the new Leader can fail and no other backup is locking for it. This will lead to the delay of having a Leader controlling the area and the loss of information depending on the control replication strategy used.

06.POLICIES DESIGN

The policies are the rules that affect directly the behavior of all the devices in the system. These rules usually are static and universal for all the devices or applications. As an example, in the cloud, the service manager can set the rules for the resource escalation, load balance and permissions. However, in a heterogeneous and changing scenario as the edge is, a more fined and customizable way to control is preferable.

Moreover, the policies are based on the necessities of the system and the application context. In other words, if there are no mechanisms that use the policies, it makes no sense to define them.

Policies sub-module design include a module called Policies Distribution, in charge of the replication and reception of new policies, with the subsequent treatment and addition into the current active ones.

On this section, the proposed policies for the resilience of the area itself and the mechanisms that use them are explained. Furthermore, the proposed mechanism to spread the rules is also provided and a brief definition on what the project considers as dynamic policies.

6.1. Rule Based Systems

For each system, may be a different set of policies and grouped by responsibility or functional block. This leads to distinguish between two different groups of policies: First, the policies directly involved on the resilience of the area, and second, the added policies that are not involved on the resilience.

For the first ones, in Table 1, for each submodule or a transversal functionality (e.g. Leader election), there is a set of policies.

	Policy	**Mechanism**
Leader Election	Leader Election Policies (LEP) • Leader Mandatory Policies (LMP) • Leader Discretionary Policies (LDP)	Leader Selection (LS) *(inside the Agent start for the mF2C)*
Leader Selection	Passive Leader Selection Policies (PLSP)	Passive Leader Promotion (PLP)
	Automatic Leader Selection Policies (ALSP)	Automatic Leader Promotion (ALP)
Area Protection	Leader Protection Policies (LPP)	Area Resilience (AR)
	Leader Reelection Policies (LRP)	Leader Reelection (LR)
Distr. Policies	Distribution Policies (DP)	Policies Distribution (PD)

Table 1 Relation between Policies and Mechanisms

Leader Election Policies

Set of policies related to the election process of a leader, sub-divided into two groups:

- ***Leader Mandatory Policies (LMP)***: Policies that require a set specific amount of resources or capabilities to be electable as Leader. It also includes policies related to the mechanism associated to these policies.

- ***Leader Discretionary Policies (LDP)*****:** Policies that specify the best amount of resources and optional capabilities to be ranked on a higher position to be a Leader. As in LMP, it also includes policies related to the mechanism associated.

The implementation of these policies is specific for each system, as the responsibility to select a Leader (as explained in the next point), affects the behavior of the components. However, the Area Resilience sub-module uses these policies to determine if the device is capable to participate into the resilience process (see 5.1.1. Backup Election and Keepalive Protocol).

Leader Selection Policies

Set of policies related to the selection of the leader itself. Using the Leader Election Policies or other specifications, it defines the rules regarding the selection process itself (i.e. become a Leader). Sub-divided into two set of policies:

- ***Passive Leader Selection Policies (PLSP):*** Policies that specify the conditions for when a device becomes a Leader by a static parameter without requiring an external change. For example, if the system manager establishes the condition that a specific device should act as a Leader of the area, only if it fits the requirements to be a Leader, the device will be the Leader at the startup.

- ***Automatic Leader Selection Policies (ALSP):*** Policies that specify the conditions for when a device becomes a Leader by an automatic trigger when a condition is reached or some external change. For example, if the device tries to reach a Leader but does not succeed, if is capable and the policy is defined, it becomes the new Leader.

On the CRM design, the Leader selection is located inside the External Orchestration (see 7.6. Illustrative Use Case: mF2C Integration), as is a coordinated process with other modules and has an impact on the behavior of the whole device and application. A detailed explanation is provided in next section.

Area Protection Policies

Set of policies related to the protection mechanisms of the Leader and the area. For instance, there are two main sub-groups corresponding to the two main mechanisms:

- ***Leader Protection Policies (LPP):*** Policies that specifies the parameters related to the protection of the Leader and the backup strategy. Some of the proposed policies are the following:
    - Minimum number of backups required for the area.
    - Maximum number of backups required for the area.
    - Time-to-Live (TTL) for the backups to be considered alive w/o any communication.
    - Maximum retry reconnection attempts between the Backup and the Leader before takeover.
    - Period of time between backup selection processes (if backups missing).
    - Keepalive challenge messages period.
    - Decrease TTL speed rate.

- ***Leader Reelection Policies (LRP):*** Policies that specifies the parameters related to the reelection of the Leader and the takeover procedure. For instance, the policy that allows this procedure to be enabled.

On the CRM design, the mechanism that directly uses the LPP policies is the Area Resilience while the LRP policies are used by the Leader Reelection.

Distribution Policies

Set of policies related to the distribution of the policies across the system. As an example, synchronization intervals and the activation of the periodic transmission.

Besides the proposed policies, the Policy sub-module is designed to be extensible to any new mechanism, internal or external to the CRM application/module. That means that any new policy can be added to the existent ones or a whole new group of them, without any specific modification of the existing design.

6.2. Dynamic Policies

As explained before, the definition of dynamic policies on the edge can potentially enhance the overall performance and reduce service delays. In such heterogeneous and changing scenario, customized rules for each device is a preferable solution.

The design of the policies consists on the definition of a name for each new rule added and a value that the mechanism uses. As external mechanisms may access to that information, the design of the module must not define any static schema. Given that, it is allowed to add any new rule to the existent groups without any code modification. Although, the addition on a new group require some code specifications and implementation following the proposed design and interfaces, explained with more detail on 7.3.2. Specific Policies Definition.

The CRM by itself does not have any specific mechanism to evaluate service performance. However, other modules can use or define new policies inside and use the proposed mechanism to distribute the rules over the area.

6.3. Replication of the Policies

Another related point with the dynamicity of the policies is the ability to spread new rules on the system. To do that, the Leader has the view of the area and the control over it to enforce the policies.

Policies Distribution module design contains a mechanism in charge of formatting and send the current selected policies to the targeted Agent, and also a mechanism to receive new policies. This replication can be triggered at the Leader to spread the rules over the area, or to a specific agent externally without the spread into the area.

07.BLOCKS IMPLEMENTATION AND INTEGRATION

In this section, the implementation for each one of the components and the integration with the mF2C is presented. Moreover, a summary of the technologies and methodology used and the reasons behind these decisions is also provided.

The code can be found in the GitHub public repository of the project[8] under the Apache-2.0 license. The code on this repository has been imported from a private one, used for the early development of the project.

7.1. Technology and Methodology

The programming language used to implement this project is **Python** [32], specifically the version 3.7. The reason behind this decision is that Python is a very powerful tool for fast prototyping and cross platform development, ideally for testing and to deploy the solution in multiple devices and operative systems, without specific compilation problems.

For the deployment of the solution, the CRM module provides a Docker image (that can be built using the source code or downloaded from DockerHub). **Docker** [33] is a *platform-as-a-service* (PaaS) platform that provide the ability to package and run any application in a virtual isolated environment called container [34] in all supported operative systems without changes in the application. The use of this tool allows to develop, test and deploy any solution in an easy and scalable way. One strong point on using Docker is that the consumption of resources in comparison to a virtual machine is way more reduced.

For the API development, the project uses **Swagger UI** [35] to represent in a graphical and interactive way the module interface. The specific library implementing Swagger and the REST server used for the project implementation is **Flask-RESTPlus** [36]. This library also allows the creation of a graphical documentation from the implemented code. More on that in section 7.5. API and CRM Core.

Other libraries used are **logging** [37] and **colorlog** [38] to provide useful and compressible logs of the application, and **psutil** [39] as a cross-platform library for obtaining the system and process information.

The used methodology to organize the project is **Kanban** [40] with the help of the **Trello** [41] application tool. The main philosophy of the project is to provide modular incremental pieces to archive the final product and change over time. Following that line, all the code is versioned using **Git** [42] to record all changes and **GitHub** [43] public release repository or **GitLab** [44] for private early development repository, in order to store the code.

Moreover, for the integration inside the mF2C project, other tools used to work in the project are **Slack** [45] for communications; **Waffle** [46] and **GitHub Projects** [47] for task planning; distribution and documentation; and **Travis CI** [48] as a continuous development/integration tool. The mF2C project organization follow the **Scram** [49] methodology with weekly meetings and sprints of two weeks for development.

[8] Repository address: https://github.com/ALEJANDROJ19/ControlResilienceManagement

7.2. Leader Protection

The Leader Protection sub-module is the core of the resilience that the CRM brings to the area. Following the design, there are two main modules inside: The Area Resilience and the Leader Reelection.

7.3.1. Area Resilience

The Area Resilience implementation follow the general approach described in the design, where the module is the same regardless of the actual role of the Agent. The implications in terms of the development are that a common structure should be created with the necessary subroutines to address the specific responsibilities for each role.

The area resilience implementation consists on a primary thread or common flow where all the agents regardless of the role execute. On this routine, the agent self-evaluates if is capable to be a Leader and if it is started as Leader. If the device does not have the capability to be a Leader, the execution of the area resilience is resumed. On the other side, if is capable, some preliminary operations are executed and the agent enters into a wait state to be selected as Backup by the Leader. This procedure is represented in a diagram in Fig. 11.

If the agent started as Leader (or Backup becomes Leader), the flow directly skips to the leader flow execution. This flow consists of two threads: One that is continuously checking if there are sufficient backups according to the active policy, and the other one that is checking if backups are alive.

As the Keepalive designed protocol consists on a constant stream of requests to the Leader, each time that a Backup challenges the Leader, the Leader reset the TTL of the Backup. If no challenge message is received from the Backup, the Leader removes the Backup from the internal database.

A more detailed view can be found on Fig. 16. The backup promotion thread and the backup keepalive procedure are represented as the two big loops of the diagram. Messages 1 and 2 in the diagram are common for all the agents. In case of backup promotion failure (message 5), the leader tries with the next device on the list.

When a Leader fails, the Backup in charge of the takeover starts the procedure, as shown in Fig. 17. Here there are three devices on the topology: one Leader, one active Backup and a normal agent. When the failure is detected by the Backup, the KeepAlive thread (message 1) changes to the leader pre-configuration and backup selection procedure (message 3), while the backup watcher (also named Keeper) is started. The remaining agent then is selected to be the new active backup and the area resume its current activity (messages 6 to 9 plus keepalive on message 11). If Light Discovery is present in CRM, the module is also triggered to start the beaconing process (message 4).

Depending on the active policy regarding the control data replication, there is another thread in charge. For instance, in the CRM standalone implementation, the KU is piggybacked into the KeepAlive response message, but by default the ZK is enforced.

Another implementation aspect of the module is the interface specification to preserve the isolation design principle. In Table 2, all the interface methods of the module are described.

Area Resilience Interfaces		
Function Name	**Arguments**	**Functionality**
start	***deviceID <string>:*** self-ID of the agent.	Start the Area Resilience threads and flows.
stop		Stop all the threads and current activity on the module.
promotedToBackup	***leaderIP <string>:*** IP of the Leader.	Promote from normal agent to Backup.
deleteBackup	***deviceID <string>:*** ID of the backup.	Remove an active Backup from the database. *(Only on the Leader)*
addBackup	***deviceID <string>:*** ID of the new backup. ***deviceIP <string>:*** IP of the new backup. ***priority <integer>:*** Assigned priority.	Add a device as an active Backup. *(Only on the Leader)*
getBackupDatabase		Get a copy of the actual state of the active backups in the area. *(Only on the Leader)*
imBackup		Get if the agent is in Backup state in the Area Resilience.
imLeader		Get if the agent is in Leader state in the Area Resilience.
receive_keepalive	***deviceID <string>:*** ID of the backup sending the Keepalive. ***payload <string>:*** Payload of the challenge message *(optional)*	Receive a Keepalive from an active Backup. *(Only on the Leader)*

Table 2 Area Resilience interface specification

All the different messages used to communicate between CRM APIs follow the REST definition. The requests are sent via TCP, with the payload formatted in JSON and all the packet encoded with the HTTP definitions (e.g. headers, status codes, request headers…). The internal calls to the module interface, that are not sent to the API, are done in code and does not use any network message. However, the module interface must be followed, or an exception may raise.

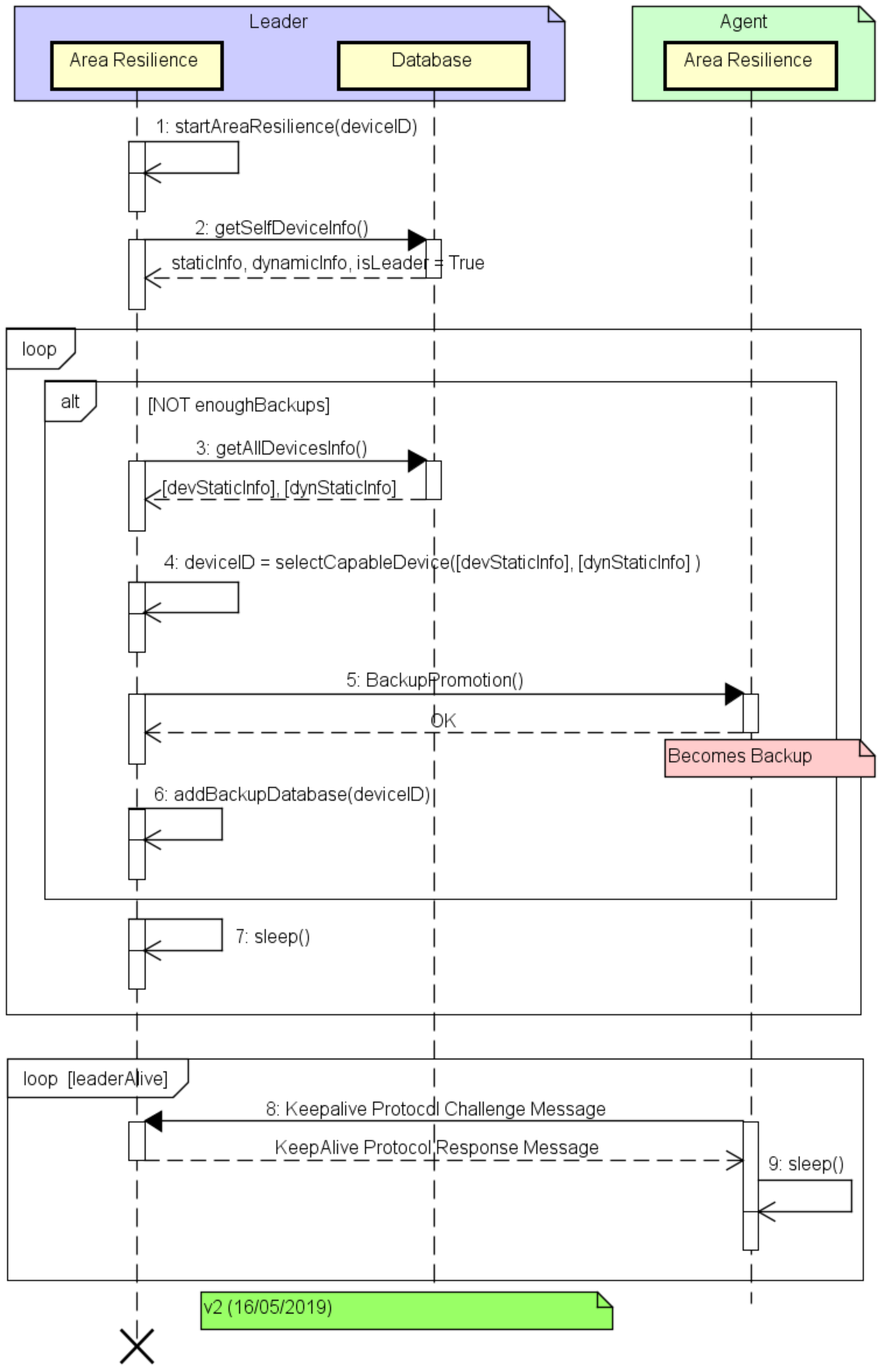

Leader
Area Resilience
Database
Agent
Area Resilience
1: startAreaResilience(deviceID)
2: getSelfDeviceInfo()
staticInfo, dynamicInfo, isLeader = True
loop
alt
[NOT enoughBackups]
3: getAllDevicesInfo()
[devStaticInfo], [dynStaticInfo]
4: deviceID = selectCapableDevice([devStaticInfo], [dynStaticInfo])
5: BackupPromotion()
OK
Becomes Backup
6: addBackupDatabase(deviceID)
7: sleep()
loop [leaderAlive]
8: Keepalive Protocol Challenge Message
KeepAlive Protocol Response Message
9: sleep()
v2 (16/05/2019)

Fig. 16 Area Resilience selecting and managing Backups (Leader view)

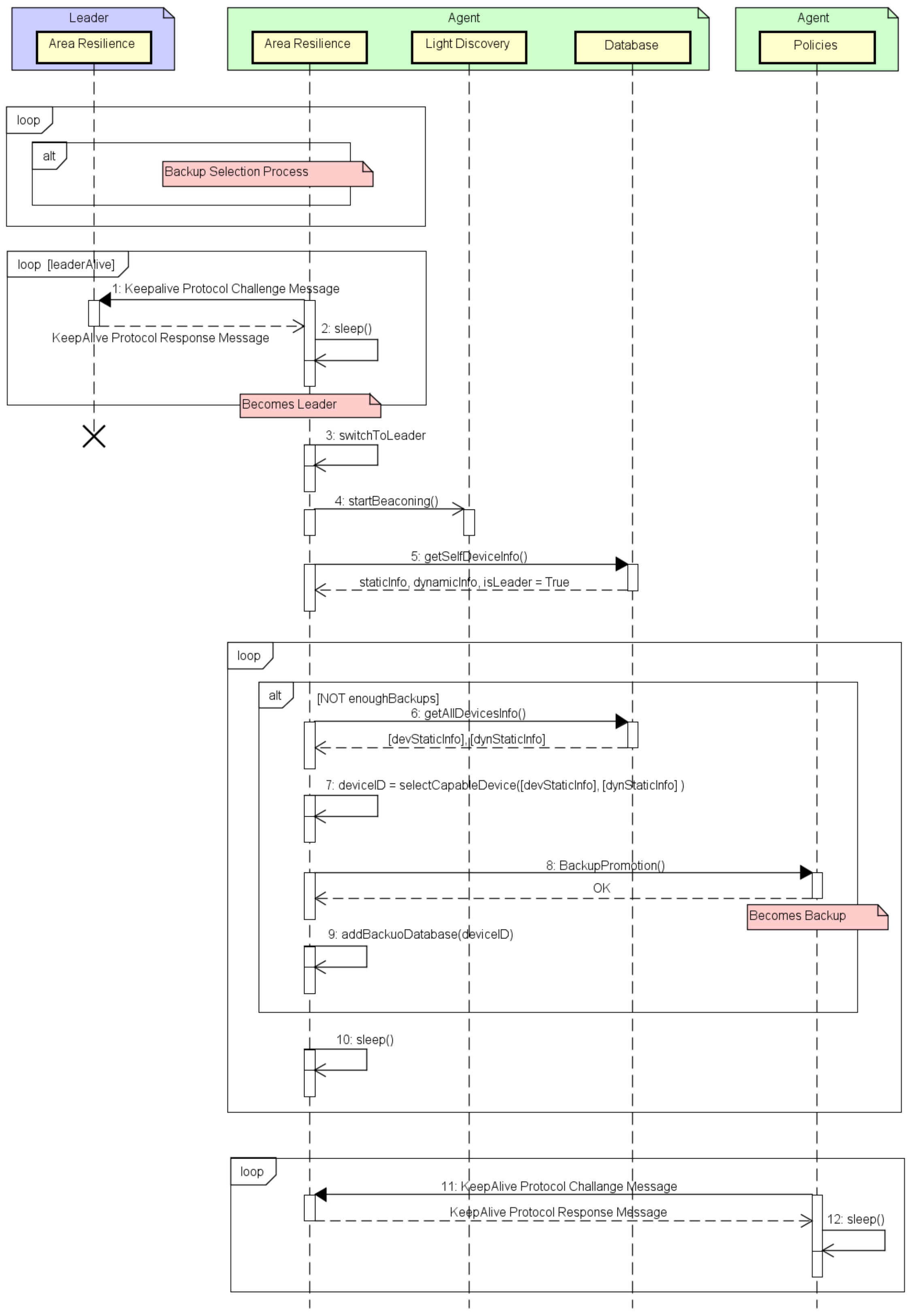

Leader
Area Resilience
Agent
Area Resilience
Light Discovery
Database
Agent
Policies
loop
alt
Backup Selection Process
loop [leaderAlive]
1: Keepalive Protocol Challenge Message
KeepAlive Protocol Response Message
2: sleep()
Becomes Leader
3: switchToLeader
4: startBeaconing()
5: getSelfDeviceInfo()
staticInfo, dynamicInfo, isLeader = True
loop
alt
[NOT enoughBackups]
6: getAllDevicesInfo()
[devStaticInfo], [dynStaticInfo]
7: deviceID = selectCapableDevice([devStaticInfo], [dynStaticInfo])
8: BackupPromotion()
OK
Becomes Backup
9: addBackuoDatabase(deviceID)
10: sleep()
loop
11: KeepAlive Protocol Challange Message
KeepAlive Protocol Response Message
12: sleep()

Fig. 17 Area Resilience Leader Failure

7.3.2. Leader Reelection

The Leader reelection implementation uses some of the area resilience functionalities and interfaces to work. The execution of the reelection consists on a single thread that execute the designed reelection flow.

In Fig. 18, a reelection process is triggered on the Leader to select agent/4 as the new Leader. As the selected agent is not an active Backup, is selected with preference 0 (the highest one). Then, if the selection has been successful, the active backup is demoted and finally the leader is self-demoted. The active backup, the agent that has been reelected to be the Leader, is promoted to be the new Leader.

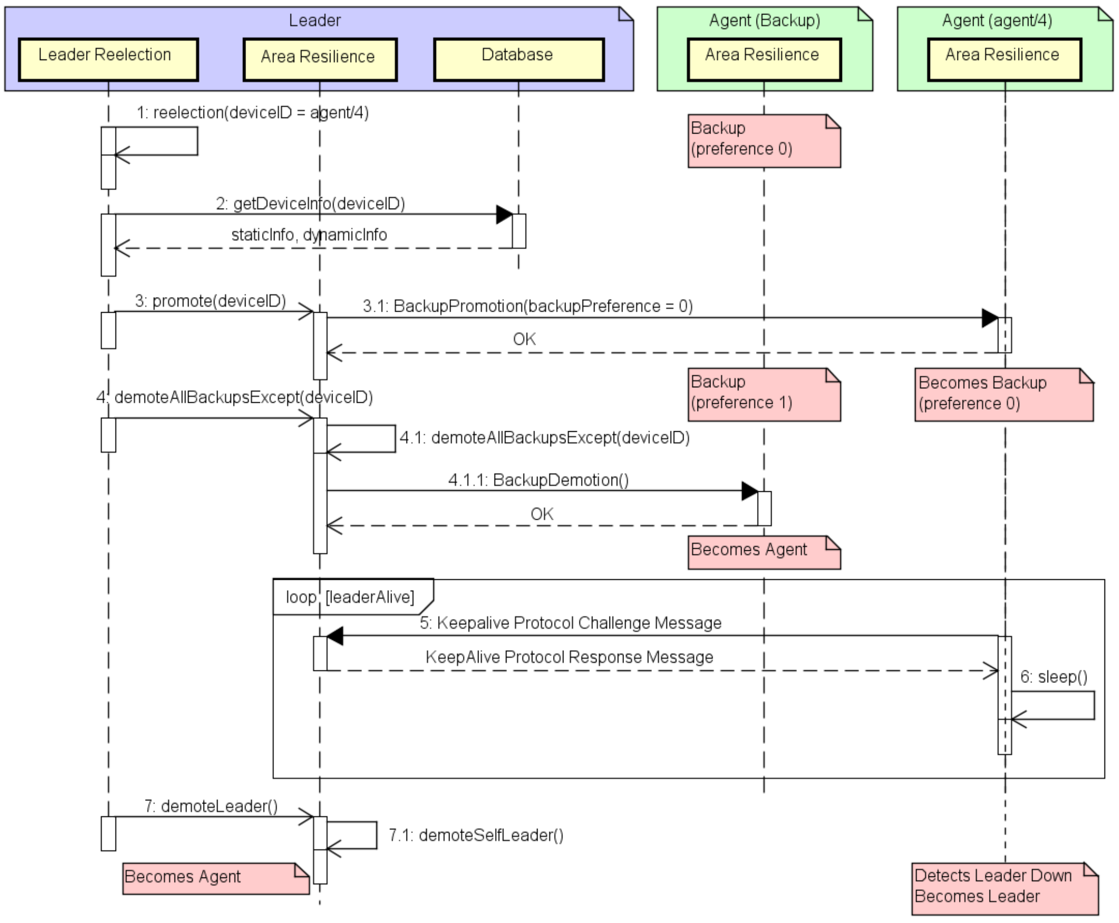


Fig. 18 Leader Reelection implementation and integration diagram

The interface to this module consists on only one that triggers the whole procedure, checking that the parameters are correct and the device is a Leader, as described in Table 3.

Area Resilience Interfaces		
Function Name	**Arguments**	**Functionality**
reelection	***areares <AreaResilience>:*** Module object reference. ***deviceID <string>:*** ID of the new Leader ***deviceIP <string>:*** IP of the new Leader	Start the Leader Reelection thread and flow. The specified agent enters into the reelection process and become the new Leader if successful.

Table 3 Leader Reelection interface specification

7.3. Policies

As the design of the policies specifies an open and extensible framework, that make possible the addition of new policies and change the current values, the implementation follow the same lines. The policies sub-module of the CRM has two different blocks:

- Policies Distribution: Module in charge of the replication and dynamicity of the policies.
- Specific Policies: Set of *modules* that specify a group of policies to be applied over a mechanism.

The implementation of the policies distribution and the general schema for the policies is discussed on next subsections.

7.3.1. Policies Distribution

The distribution of the policies consists on a sequential mechanism that send a copy of the current policies at the Leader to the selected agents of the area. The used channel for the communication is via TCP REST messages to the API of the targeted Agent.

The payload of the message must follow the model specified in the API: a JSON that contain the key of the policy to change and the value must contain another JSON. This inner-JSON is a dictionary where each key contains the name of the policy to change and the new value. The payload importation is made in such way that all the keys are optional. In other words, there is not the need to send all the policies to change only one.

In Fig. 19, there is an scenario with a Leader and two agents. The Leader receives a trigger to change some specific policy (message 1) and its applied after its check (message 1.1). A normal agent it can also receive the same trigger (message 2) and does the same process. Then the Leader receives the trigger to spread the policies on the area (message 3) and proceeds to send the policies to the agents (messages 3.1). These triggers can be send by any module, internally or externally to the Policies Distribution.

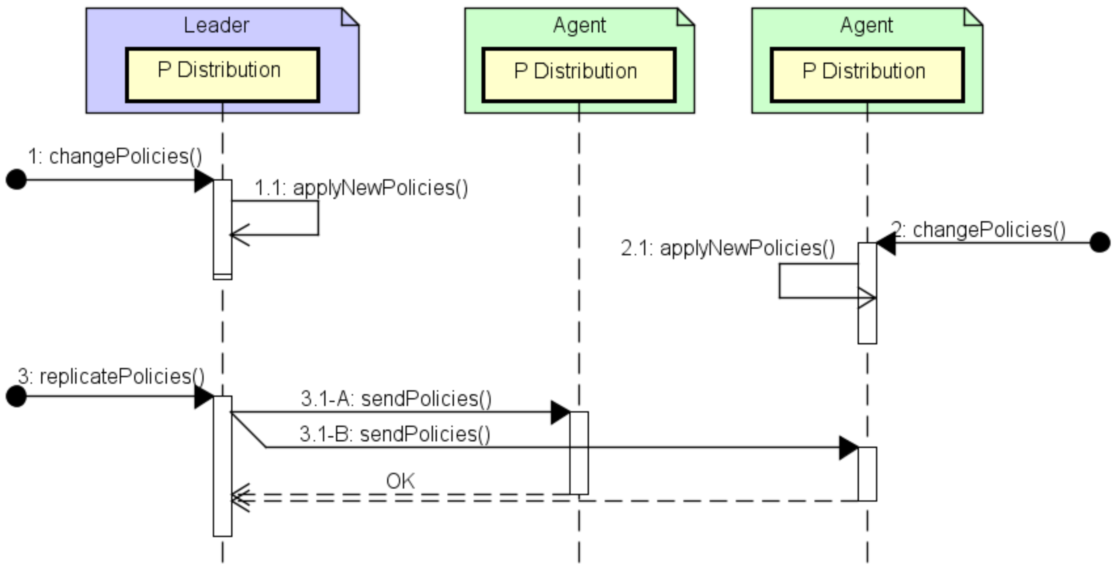


Fig. 19 Policies Distribution transfer diagram

Policy Distribution Interface		
Function Name	**Arguments**	**Functionality**
getPolicies		Return all the current policies in the Agent.
receivePolicies	***payload <dict>:*** Dictionary composed of a set of keys with the name of the policy group and the value in JSON format.	Set the new policies in the Agent.
distributePolicies	***listIPs <list<string>>:*** List with the IPs of the targeted agents.	Send all the current policies to the agents on the listIPs

Table 4 Policy Distribution interface specification

7.3.2. Specific Policies Definition

For each group of policies (e.g. Leader Protection Policies, Leader Election Policies, etc…), there is a common implementation that each one of them must follow, and use the same interface. For the definition of the policies themselves, there are no restrictions as well as the values are accepted by the JSON standard.

The policy definition consists on a standard interface and a dictionary where the policies should be stored by defining a unique key and a valid value. Optionally the name of the keys can also be defined to make easier the integration with the mechanisms and other modules. The interface specification can be found in Table 5.

Policy General Interface Definition		
Function Name	**Arguments**	**Functionality**
get	***key <string>:*** Name of the policy. ***default <T>:*** Value in case of policy not set. T refers to the type of the returned value.	Return the value for the specified policy.
get_json		Return the JSON with all the policies.
set_json	***json <string>:*** JSON with a key and value for each policy to modify or add.	Set or add the new policies to the system.

Table 5 Policy General Interface Definition

7.4. Discovery

As described in the design, the discovery sub-module of the CRM is a basic topology builder/scanner plus a little resource categorizer of the device. All the implementation is inside a unique module called Light Discovery. As the main design lines, this module can be substituted by a more elaborated one or completely removed if an external one take the responsibility.

7.4.1. Light Discovery

Light Discovery module is a single thread flow that can be either beaconing in the network the Leader messages or scan them as a normal Agent. In Fig. 20, the whole procedure can be observed. As a normal agent, the flow starts each time a beacon (message 1) is received. Then the agent first stores the leader ID inside the beacon and the IP of the device (step 1.1), and proceeds to run the resource categorization function (step 1.2). Once the information is correctly generated, a request is sent to the Leader via the API (message 1.3) with all the information inside. The Leader update the database with the discovered device (step 1.3.1) and returns a positive response.

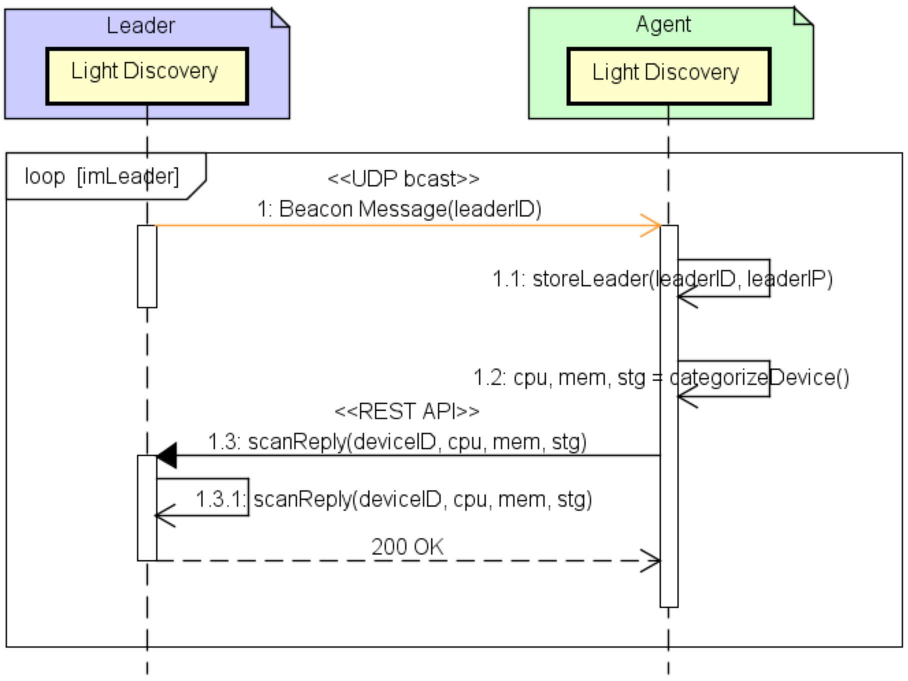

Leader
Light Discovery
Agent
Light Discovery
loop [imLeader]
<<UDP bcast>>
1: Beacon Message(leaderID)
1.1: storeLeader(leaderID, leaderIP)
1.2: cpu, mem, stg = categorizeDevice()
<<REST API>>
1.3: scanReply(deviceID, cpu, mem, stg)
1.3.1: scanReply(deviceID, cpu, mem, stg)
200 OK

Fig. 20 Light Discovery communication diagram

The beacon message strategy implemented is the broadcast of UDP datagrams addressed to the broadcast network addresses. Each Light Discovery module inside the CRM of the Agents has an UDP server that is waiting for the beacons. For each beacon received, a new reply is generated. This UDP server does not go through the API. However, the reply follows the normal procedure as all the other messages.

The beaconing flow only starts when the agent becomes a Leader while the scanning flow only when the Agent is normal or an active Backup. In any circumstance, both modes can be active, or have two Leaders in the same network.

The interface defined for this module, specified in Table 6, does not include the beacon entry (i.e. the scan server) as is an entry defined for this specific Discovery design and implementation, and others will not necessarily need such mechanism, while the interface is required by the CRM to work properly.

Light Discovery Interfaces		
Function Name	**Arguments**	**Functionality**
startBeaconing		Start the Light Discovery module in beaconing mode.
stopBeaconing		Stop the Light Discovery beaconing.
startScanning		Start the Light Discovery module in scanning mode.
stopScanning		Stop the Light Discovery scanning.
recv_reply	***payload <dict>:*** Device information from the beacon scan reply. ***deviceIP <string>:*** IP of the device replying.	Receive reply from Agent and add/modify the information to the database. *(Only on the Leader)*
get_topology		Get a copy of the current topology of the area.
leaderIP		IP of the detected Leader.
leaderID		ID of the detected Leader.

Table 6 Light Discovery interface specification

7.5. API and CRM Core

The interface to the CRM module is the API. Here, all the allowed operations from outside can be performed in an isolated and controlled way. The API is implemented to be RESTful service, or in other words, all the requests must follow the REST standard architecture plus the API specifications on the interface.

The webserver has been implemented using a modified version of Flask, which contains more support for REST API documentation and Swagger for the graphical representation. These modules allow the creation of the documentation and graphical interface, without any specific module development.

Adding an endpoint to the API is simple. First, a class must be created where the API object is defined. On top of the class definition, is necessary to specify the route of the API endpoint. Second, inside of the class, the HTTP method used for the request is specified creating a class method with the method as the function name. Third, specify using the API object the expected received models, expected result codes and documentation for each one of them. Finally, inside the method, the interface operations. An example of the Keepalive endpoint is provided in Fig. 21.

All the returned codes in the API follow the HTTP response status codes standard convention [50], a three-digit number that specify the result of the request. For each response in the API, there is an associated code. Most of the replies are completely accurate with the code meaning, but others are more specific. However, the documentation of the API removes any possible doubt, as all the replies must have a description. As a general rule, the first digit is the important one, as defines the class of response. The API only use codes contained into the 2xx (Successful) and 4xx (Client error) classes, as the default 5xx (Server error) for when the API is not available.

```
@pl.route('/keepalive')
class keepalive(Resource):
    """Keepalive entrypoint"""
    @pl.doc('post_keepalive')
    @pl.expect(keepalive_model)
    @pl.marshal_with(keepalive_reply_model, code=200)
    @pl.response(200, 'Leader alive')
    @pl.response(403, 'Agent not authorized (Not recognized as backup)')
    @pl.response(405, 'Device is not a Leader')
    def post(self):
        """Keepalive entrypoint for Leader"""
        if not arearesilience.imLeader():
            # It's not like I don't want you to send me messages or anything, b-baka!
            return {
                   'deviceID': agentstart.deviceID,
                   'backupPriority': arearesilience.PRIORITY_ON_FAILURE}, 405

        correct, priority = arearesilience.receive_keepalive(api.payload['deviceID'])
        if correct:
            # Authorized
            return {'deviceID': agentstart.deviceID, 'backupPriority': priority}, 200
        else:
            # Not Authorized
            return {'deviceID': agentstart.deviceID, 'backupPriority': priority}, 403
```

Fig. 21 API Keepalive endpoint code example

Another important component of the CRM module is the core itself. It contains all the common operations (log formatting, parameter definitions, etc…) and the creation of the sub-module objects. Working together with the API interface, the module interoperability and lifecycle is controlled.

To represent the API, Fig. 22 show all the implemented and active endpoints of the API, explained in more detail in Table 7.

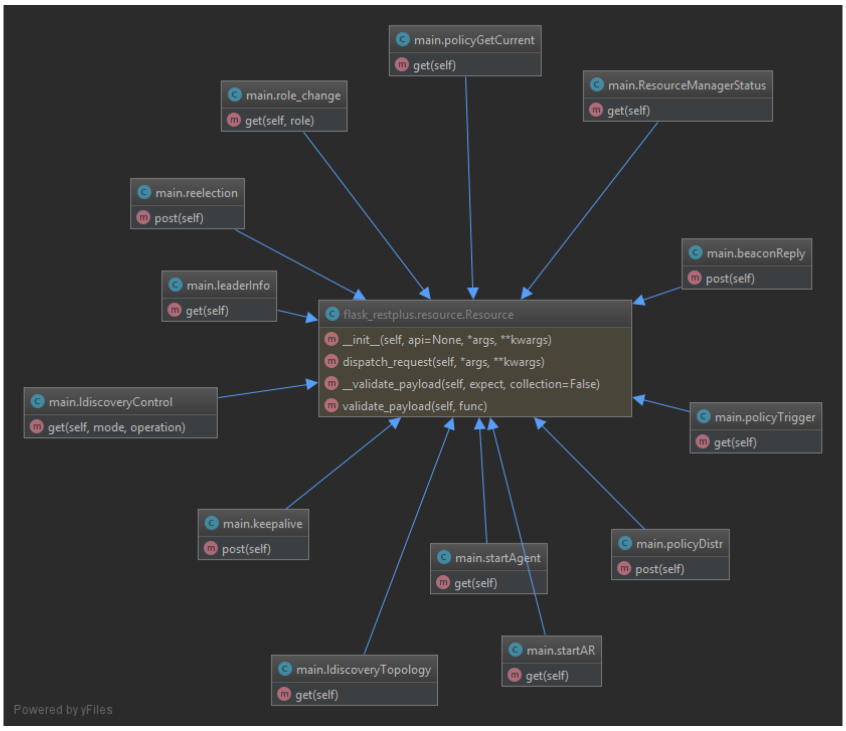


Fig. 22 API and Core integration graph

Keepalive	
Description	Keepalive entrypoint for Leader. Backups send message to this address and check if the Leader is alive. Only registered backups are allowed to send keepalives, others will be rejected.
Endpoint	**POST** /crm-api/keepalive
Payload	`{"deviceID": "agent/1234"}`
Responses	• **200** - Success • **403** - Agent not authorized • **405** - Device is not a Leader • **Response Payload:** `{ "deviceID": "leader/1234", "backupPriority": 0 }`

Leader Info	
Description	Check if the agent is a Leader or Backup.
Endpoint	**GET** /crm-api /leaderinfo
Payload	-
Responses	• **200** - Success • **Response Payload:** `{ "imLeader": false, "imBackup": false }`

Reelection	
Description	Send a message to trigger the reelection process. The specified agent will be the reelected Leader if it accepts.
Endpoint	**POST** /crm-api/reelection
Payload	`{ "deviceID": "agent/1234" }`
Responses	• **200** - Reelection Successful • **401** - The Agent is not authorized to trigger the reelection • **403** - Reelection failed • **404** - Device not found or IP not available • **Response Payload:** `{ "imLeader": false, "imBackup": false }`

Start Area Resilience	
Description	Starts the Area Resilience module
Endpoint	**GET** /crm-api/startAreaResilience
Payload	-
Responses	• **200** - Started • **403** - Already Started

Start Agent (External Orchestration)	
Description	Starts the Agent Start module.
Endpoint	**GET** /crm-api/startAgent
Payload	-
Responses	• **200** - Started • **403** - Already Started

Role Change	
Description	Change the agent from current role to specified one (leader, backup or agent).

Endpoint	**GET** /crm-api/roleChange/{role}
Parameters	**role <string>:** - agent - leader - backup
Payload	-
Responses	• **200** - Successful • **403** - Not Successful • **404** - Role not found
Light Discovery Module control	
Description	Management of the Light Discovery module status and mode.
Endpoint	**GET** /ld/control/{mode}/{operation}
Parameters	**mode <string>:** - beacon - scan **operation <string>:** - start - stop
Payload	-
Responses	• **200** - Successful operation • **403** - Not Successful operation • **404** - Mode/Operation not found
Beacon Reply	
Description	Reception of the reply to a received beacon.
Endpoint	**POST** /ld/beaconReply

Payload	`{`
	`  "deviceID": "string",`
	`  "deviceIP": "string",`
	`  "cpu_cores": 0,`
	`  "mem_avail": 0,`
	`  "stg_avail": 0`
	`}`

Responses	• **200** - Device added/modified on the topology • **400** - Error on beacon reply message
Get current Topology	
Description	Get the current topology of the Area (Leaders only).
Endpoint	**GET** /ld/topology
Payload	-
Responses	• **200** - Successful operation • **Response Payload:** `{“topology”: [(“agent/1234”, 127.0.0.1)]}`
Distribute Policies	
Description	Distribute the current policies to the attached devices.
Endpoint	**GET** /crm-api/PoliciesDistributionTrigger
Payload	-
Responses	• **200** - Trigger accepted

Set new active Policies	
Description	Define new active policies for the Agent.
Endpoint	**POST** /crm-api/receiveNewPolicies
Payload	`{ "LMR": "string", "LDR": "string", "PLSP": "string", "ALSP": "string", "LPP": "string", "LRP": "string", "DP": "string" }`
Responses	• **200** - Policies correctly received • **400** - Message malformation
Get current active Policies	
Description	Retrieve the active policies on the device.
Endpoint	**GET** /crm-api/getCurrentPolicies
Payload	-
Responses	• **200** - Policies correctly received • **Response Payload:** `{ "LMR": "string", "LDR": "string", "PLSP": "string", "ALSP": "string", "LPP": "string", "LRP": "string", "DP": "string" }`

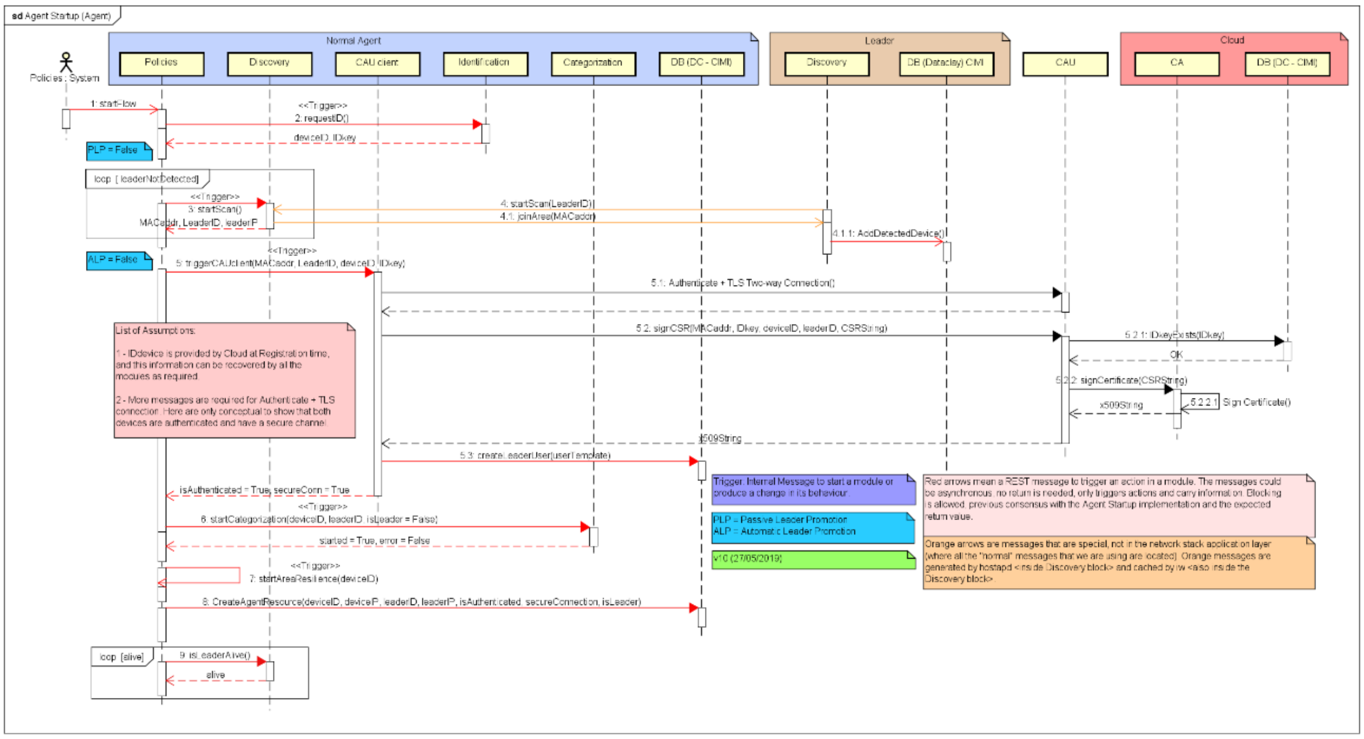
Table 7 API endpoint specification

Besides the utility itself of the API as the interface for the CRM and the webserver to attend the REST requests, the API also provides an interactive website. The documentation of the API is displayed into the website, automatically updated from the definitions in the code. Moreover, the API provide for each endpoint a test feature to send messages directly from it, with the correct format and protocol. The results are displayed, along with the HTTP headers, code and returned payload.

7.6. Illustrative Use Case: mF2C Integration

Finally, the integration of the CRM into the mF2C agent application is presented here with all the required additions and changes on the project. As displayed in Fig. 9, the CRM module can substitute the already Policies module defined on the Agent. In order to accomplish that, the External Orchestration sub-module is in charge of the integration and fulfil the missing functionalities, explained in detail on section 4.3. Functional Blocks Architecture. On this section, the specific integration of the CRM into the mF2C is explained.

All internal modules of the mF2C Agent are deployed inside Docker containers, attached to the same docker network using the docker-compose receipt. That implies that all internal communications are done directly using the docker name resolver while external communications are made using the API, preserving the network isolation. CRM is already designed and provide a Dockerfile to be deployed using Docker, meaning that no big changes are required. The required communications, implemented inside the Agent Startup module are graphically displayed in Fig. 23, using the mF2C module names and architecture.



sd Agent Startup (Agent)
Normal Agent
Policies
Discovery
CAU client
Identification
Categorization
DB (DC - CIMI)
Leader
Discovery
DB (Dataclay) CIMI
CAU
Cloud
CA
DB (DC - CIMI)
Policies : System
1: startFlow
<<Trigger>>
2: requestID()
deviceID, IDkey
PLP = False
loop [leaderNotDetected]
<<Trigger>>
3: startScan()
MACaddr, LeaderID, leaderIP
4: startScan(LeaderID)
4.1: joinArea(MACaddr)
4.1.1: AddDetectedDevice()
ALP = False
<<Trigger>>
5: triggerCAUclient(MACaddr, LeaderID, deviceID, IDkey)
5.1: Authenticate + TLS Two-way Connection()
5.2: signCSR(MACaddr, IDkey, deviceID, leaderID, CSRString)
5.2.1: IDkeyExists(IDkey)
OK
5.2.2: signCertificate(CSRString)
5.2.2.1 Sign Certificate()
x509String
x509String
5.3: createLeaderUser(userTemplate)
isAuthenticated = True, secureConn = True
<<Trigger>>
6: startCategorization(deviceID, leaderID, isLeader = False)
started = True, error = False
<<Trigger>>
7: startAreaResilience(deviceID)
8: CreateAgentResource(deviceID, deviceIP, leaderID, leaderIP, isAuthenticated, secureConnection, isLeader)
loop [alive]
9: isLeaderAlive()
alive
v10 (27/05/2019)

List of Assumptions:

1 - IDdevice is provided by Cloud at Registration time, and this information can be recovered by all the modules as required.

2 - More messages are required for Authenticate + TLS connection. Here are only conceptual to show that both devices are authenticated and have a secure channel.

Trigger: Internal Message to start a module or produce a change in its behaviour.

PLP = Passive Leader Promotion
ALP = Automatic Leader Promotion

Red arrows mean a REST message to trigger an action in a module. The messages could be asynchronous, no return is needed, only triggers actions and carry information. Blocking is allowed, previous consensus with the Agent Startup implementation and the expected return value.

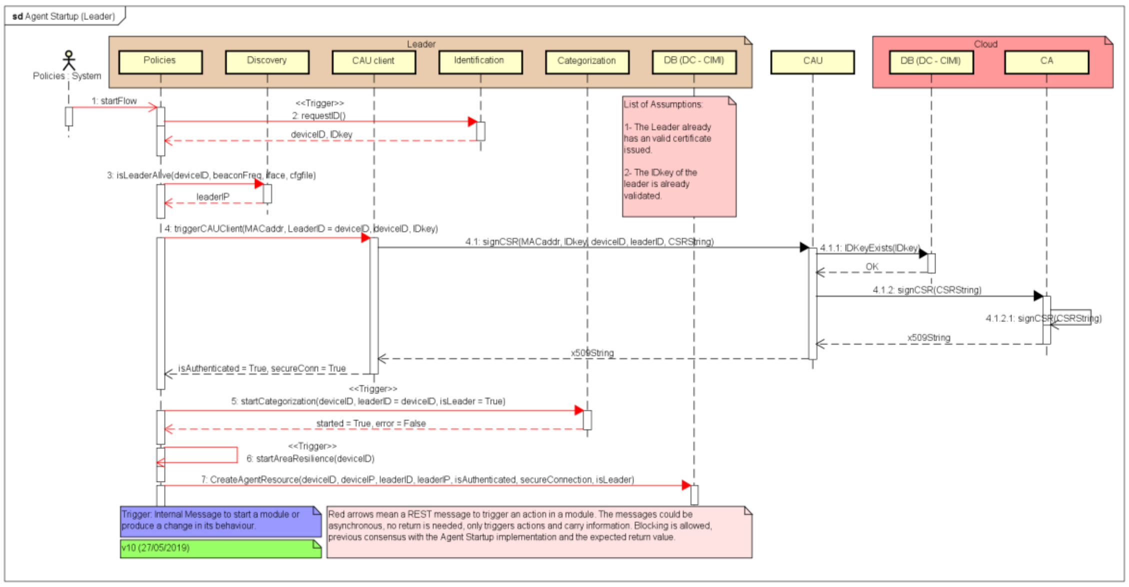
Orange arrows are messages that are special, not in the network stack application layer (where all the "normal" messages that we are using are located). Orange messages are generated by hostapd <inside Discovery block> and cached by iw <also inside the Discovery block>.

Fig. 23 mF2C Agent Start integration (Agent)

Policies module, or Agent Start on the CRM architecture, receives the request to start the module (message 1) and sends the trigger to Identification module (message 2), returning the deviceID and IDkey used for identification of the agent and authentication procedures. For the sake of the explanation, the PLP mechanism return that the device is not set to be a Leader. Discovery is triggered to start scanning for a Leader in the vicinity (message 3). The discovery procedure is different from the one explained in the CRM design. Discovery on the leader send 802.11 wireless beacons (message 4), and if the agent detects one of them, establishes a wireless connection (message 4.1). If the connection is successful, the Discovery module returns the detected ID of the leader and other internal parameters.

Once this is done, the authentication mechanism is triggered (messages 5.x), categorization module is started (message 6) and the Area Resilience inside is triggered as well (message 7). Finally, internal received values of the overall operation are updated into the Agent database using the common interface CIMI (message 8), and the agent keeps checking if the Leader is still broadcasting.

A similar procedure is done with the Leader flow, shown in Fig. 24. The same trigger is sent to the Agent Start (message 1), running the Identification request (message 2) and startup of the Discovery module as Leader (message 3). The specific authentication procedures are done in the CAU client module (message 4) and finally, the categorization and area resilience modules are triggered (messages 5,6), storing all the procedure results into the database (message 7).


sd Agent Startup (Leader)
Leader
Policies : System
Policies
Discovery
CAU client
Identification
Categorization
DB (DC - CIMI)
CAU
Cloud
DB (DC - CIMI)
CA
1: startFlow
<<Trigger>>
2: requestID()
deviceID, IDkey
3: isLeaderAlive(deviceID, beaconFreq, iface, cfgfile)
leaderIP
4: triggerCAUClient(MACaddr, LeaderID = deviceID, deviceID, IDkey)
4.1: signCSR(MACaddr, IDkey, deviceID, leaderID, CSRString)
4.1.1: IDKeyExists(IDkey)
OK
4.1.2: signCSR(CSRString)
4.1.2.1: signCSR(CSRString)
x509String
x509String
isAuthenticated = True, secureConn = True
<<Trigger>>
5: startCategorization(deviceID, leaderID = deviceID, isLeader = True)
started = True, error = False
<<Trigger>>
6: startAreaResilience(deviceID)
7: CreateAgentResource(deviceID, deviceIP, leaderID, leaderIP, isAuthenticated, secureConnection, isLeader)
v10 (27/05/2019)

List of Assumptions:

1- The Leader already has an valid certificate issued.

2- The IDkey of the leader is already validated.

Trigger: Internal Message to start a module or produce a change in its behaviour.

Red arrows mean a REST message to trigger an action in a module. The messages could be asynchronous, no return is needed, only triggers actions and carry information. Blocking is allowed, previous consensus with the Agent Startup implementation and the expected return value.

Fig. 24 mF2C Agent Start integration (Leader)

The implementation of both flows inside the Agent Start module uses a single-threaded strategy, as if any of the steps fails, the agent must be stopped. Although, for testing purposes, if the debug flag is set at the deployment of the module, if a module fails the whole process is not interrupted but notified. This thread can be switched depending if the agent is acting as normal or leader. However, being an active Backup does not have any implication on the execution of the flow, rather than the incompatibility with the Leader flow.

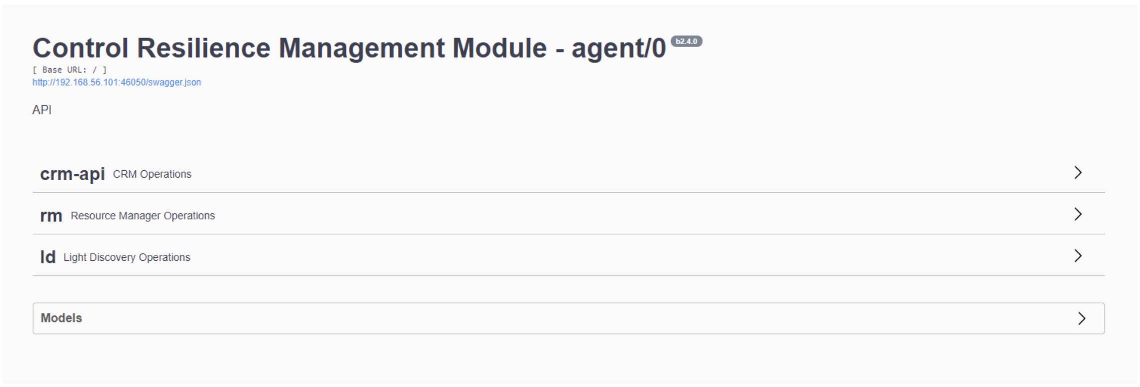
Same as the other CRM components, calls to the modules interfaces are done using specified APIs and using REST standards. CRM API has been modified to provide the necessary information to other mF2C components, such as the actual state of the agent or current policies.

08.PROJECT RESULTS AND VALIDATION

On this section, individual results of the project objectives are displayed. In specific, the resilience objective and the policies distribution are evaluated with a set of tests. Moreover, an analytical experiment of applying different policies and the effect in the area is presented. Finally, the validation of the project into the mF2C project is also demonstrated.

8.1. API GUI

One of the first results of the project is the graphical part of the CRM application. As designed and implemented, the CRM API also provides a website with the documentation and interactive request builder. The actual look of the website can be seen in Fig. 25.



Control Resilience Management Module - agent/0 b2.4.0
[Base URL: /]
http://192.168.56.101:46050/swagger.json

API

crm-api CRM Operations
rm Resource Manager Operations
ld Light Discovery Operations

Models

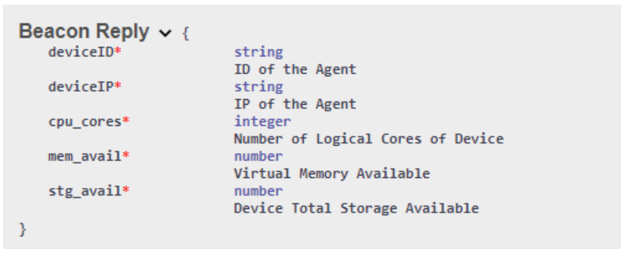
Fig. 25 API CRM website

The website is divided into two main sections: All the endpoints grouped into the defined categories, and the defined payload models associated for each endpoint.

Starting with the first section, an example with the Keepalive endpoint is provided in Fig. 27. For each endpoint, there is a documentation of the required parameters and the possible results. For instance, Keepalive require a specific JSON payload that contain the deviceID of the sender. At the return, if successful, the received payload follows the specified model.

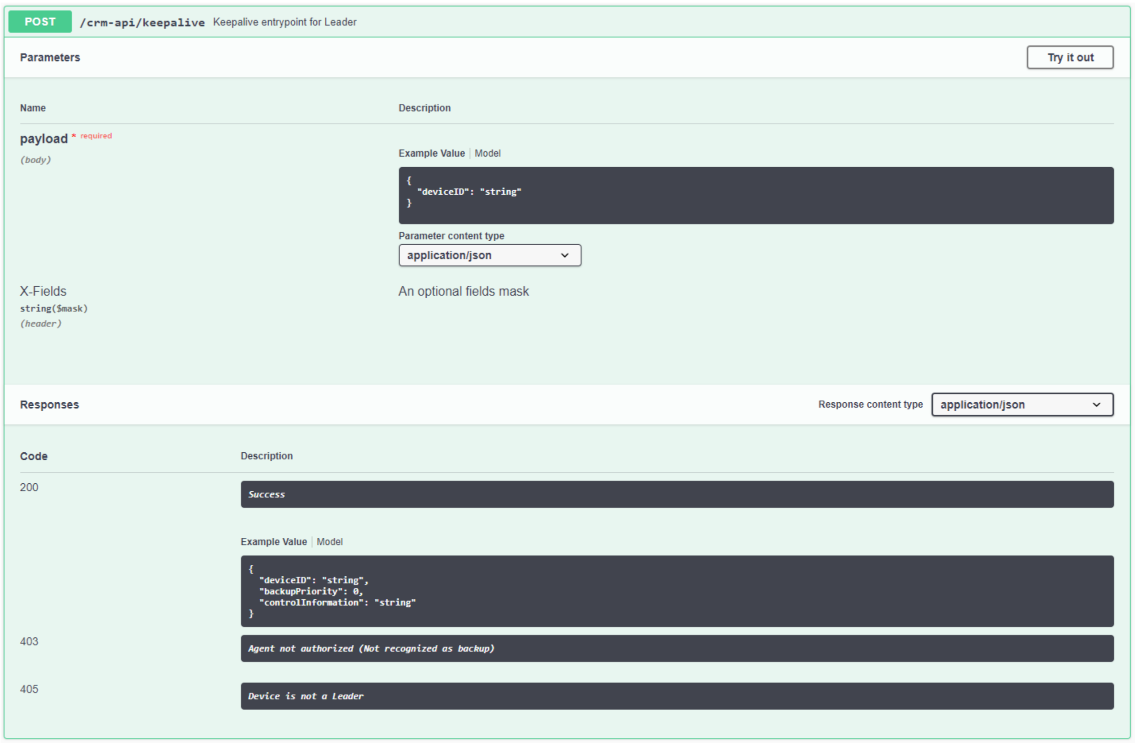
The endpoint can be tested or send real commands using the "Try it out" button in each endpoint. For example, in Fig. 28, the current active Policies are received when the command is executed. On the response part, the curl command and URL used are displayed and at the bottom the result, with the status code and the returned payload.

For the model specification, the specified model for the beacon reply is shown in Fig. 26. Each required key field (represented with a red asterisk), has an associated value type. Moreover, an example can be provided into the API documentation as reference.



```
Beacon Reply ⌄ {
    deviceID*              string
                           ID of the Agent
    deviceIP*              string
                           IP of the Agent
    cpu_cores*             integer
                           Number of Logical Cores of Device
    mem_avail*             number
                           Virtual Memory Available
    stg_avail*             number
                           Device Total Storage Available
}
```

Fig. 26 Beacon reply payload model



POST `/crm-api/keepalive` Keepalive entrypoint for Leader

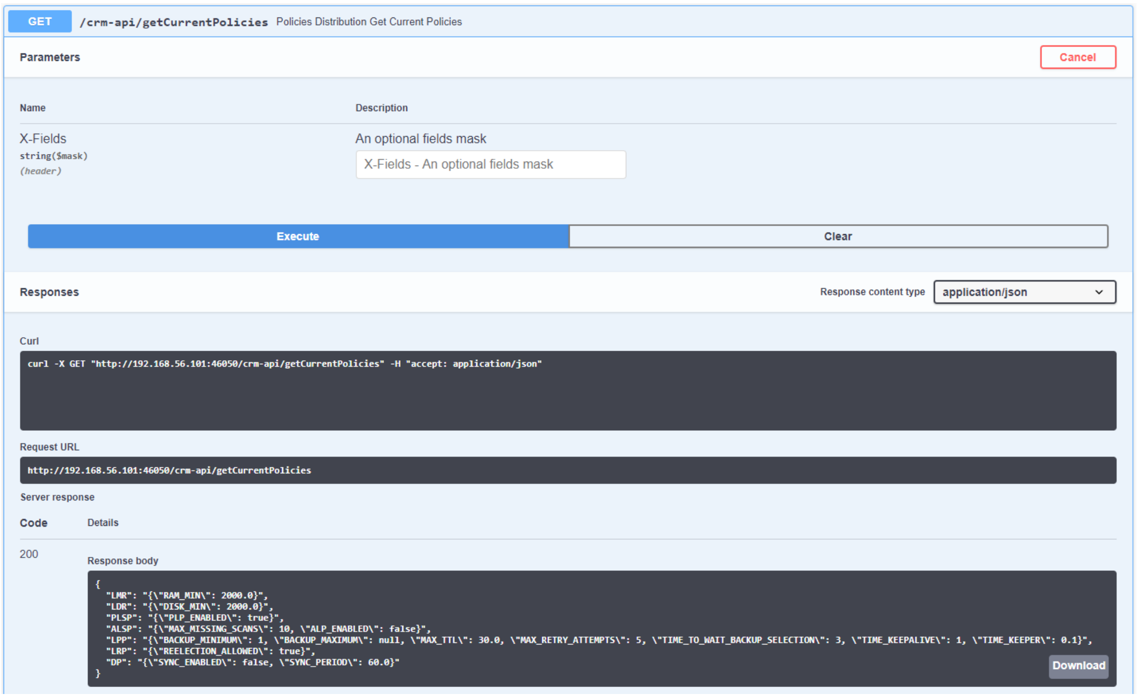
Parameters

Name	Description
payload * required (body)	Example Value | Model `{ "deviceID": "string" }` Parameter content type: application/json
X-Fields string($mask) (header)	An optional fields mask

Responses — Response content type: application/json

Code	Description
200	*Success* Example Value | Model `{ "deviceID": "string", "backupPriority": 0, "controlInformation": "string" }`
403	*Agent not authorized (Not recognized as backup)*
405	*Device is not a Leader*

Fig. 27 Keepalive API documentation



GET `/crm-api/getCurrentPolicies` Policies Distribution Get Current Policies

Parameters

Name	Description
X-Fields string($mask) (header)	An optional fields mask X-Fields - An optional fields mask

Responses — Response content type: application/json

Curl

```
curl -X GET "http://192.168.56.101:46050/crm-api/getCurrentPolicies" -H "accept: application/json"
```

Request URL

```
http://192.168.56.101:46050/crm-api/getCurrentPolicies
```

Server response

Code: 200 — Details: Response body

```
{
  "LMR": "{\"RAM_MIN\": 2000.0}",
  "LDR": "{\"DISK_MIN\": 2000.0}",
  "PLSP": "{\"PLP_ENABLED\": true}",
  "ALSP": "{\"MAX_MISSING_SCANS\": 10, \"ALP_ENABLED\": false}",
  "LPP": "{\"BACKUP_MINIMUM\": 1, \"BACKUP_MAXIMUM\": null, \"MAX_TTL\": 30.0, \"MAX_RETRY_ATTEMPTS\": 5, \"TIME_TO_WAIT_BACKUP_SELECTION\": 3, \"TIME_KEEPALIVE\": 1, \"TIME_KEEPER\": 0.1}",
  "LRP": "{\"REELECTION_ALLOWED\": true}",
  "DP": "{\"SYNC_ENABLED\": false, \"SYNC_PERIOD\": 60.0}"
}
```

Fig. 28 Get current Policies execution example

With this graphical and easy-to-use interface, the whole CRM can be controlled and modified, at manager level without modifying the code.

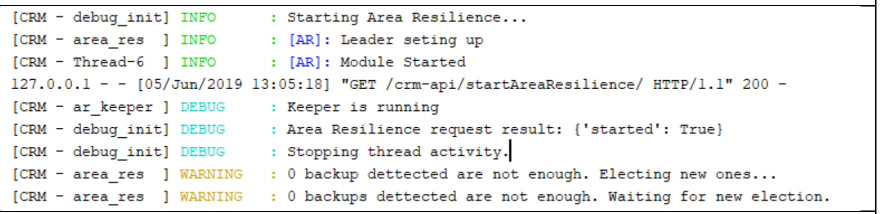
8.2. Resilience

To verify that the CRM provides resilience in a F2C scenario, there are a set of tests to evaluate if in the proposed situations, the CRM works as intended and provides a solution to the problem.

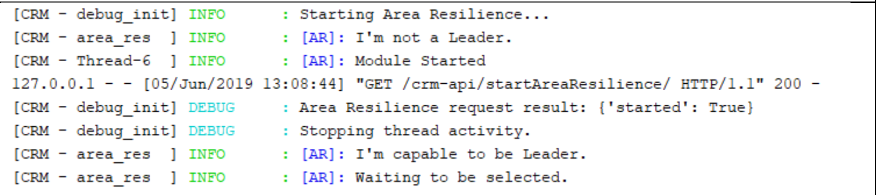
There are three groups of tests: Firsts, some basic functionality tests about the functionality of the modules itself; Second, the tests regarding the Area Resilience functionality and some corner cases of the scenario; Third, the tests for the Leader Reelection.

All the tests are provided with a screenshot of the CRM output, and in some of them with the API website as well.

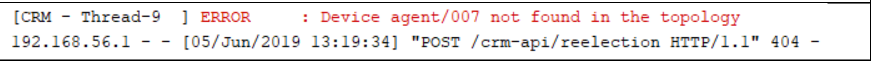
Basic Functionality Tests of module startup:

Area Resilience on Leader alone	
Test Description	Area Resilience starts as Leader. The Leader tries to select a Backup in an empty topology.
Expected Result	Area Resilience successfully started acting as Leader. Backup promotion retry until a backup is selected or stopped.
Test Result	**Success**
Screenshot	
Observations	

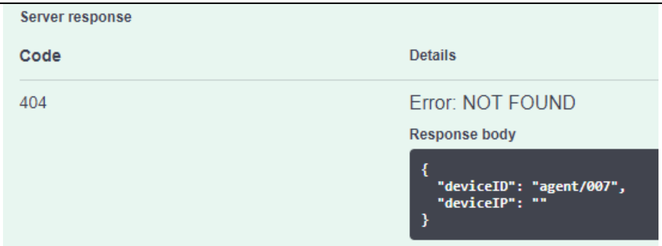
```
[CRM - debug_init] INFO      : Starting Area Resilience...
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-6  ] INFO      : [AR]: Module Started
127.0.0.1 - - [05/Jun/2019 13:05:18] "GET /crm-api/startAreaResilience/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG     : Keeper is running
[CRM - debug_init] DEBUG     : Area Resilience request result: {'started': True}
[CRM - debug_init] DEBUG     : Stopping thread activity.
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] WARNING   : 0 backups dettected are not enough. Waiting for new election.
```

Area Resilience on Agent alone	
Test Description	Area Resilience starts as Agent. The agent, set as a capable device, should wait for a Leader election.
Expected Result	Area Resilience successfully started acting as Agent. Wait to be selected or stopped.
Test Result	**Success**
Screenshot	
Observations	

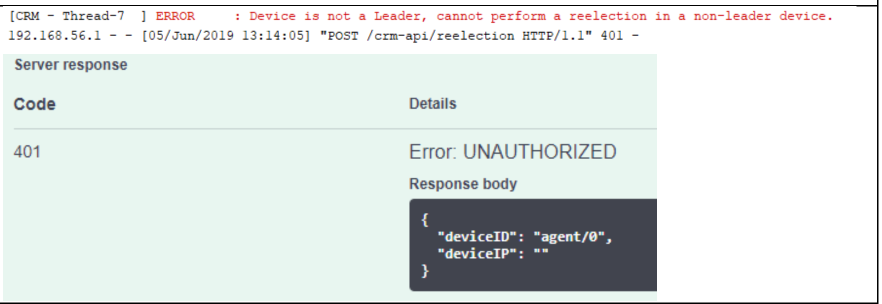
```
[CRM - debug_init] INFO      : Starting Area Resilience...
[CRM - area_res  ] INFO      : [AR]: I'm not a Leader.
[CRM - Thread-6  ] INFO      : [AR]: Module Started
127.0.0.1 - - [05/Jun/2019 13:08:44] "GET /crm-api/startAreaResilience/ HTTP/1.1" 200 -
[CRM - debug_init] DEBUG     : Area Resilience request result: {'started': True}
[CRM - debug_init] DEBUG     : Stopping thread activity.
[CRM - area_res  ] INFO      : [AR]: I'm capable to be Leader.
[CRM - area_res  ] INFO      : [AR]: Waiting to be selected.
```

Leader Reelection of a non-existent Agent	
Test Description	Trigger the reelection with a non-valid or non-existent agent ID.
Expected Result	Error code on request, as specified on the API.
Test Result	**Success**
Screenshots	

```
[CRM - Thread-9  ] ERROR     : Device agent/007 not found in the topology
192.168.56.1 - - [05/Jun/2019 13:19:34] "POST /crm-api/reelection HTTP/1.1" 404 -
```

	 Server response Code | Details 404 | Error: NOT FOUND Response body `{ "deviceID": "agent/007", "deviceIP": "" }`
Observations	Screenshot at top is from log output of the agent, the one at the bottom is from the API dashboard at the agent.

Leader Reelection on a non-Leader device	
Test Description	Trigger the reelection in an agent that is not a Leader.
Expected Result	Error code on request, as specified on the API.
Test Result	**Success**

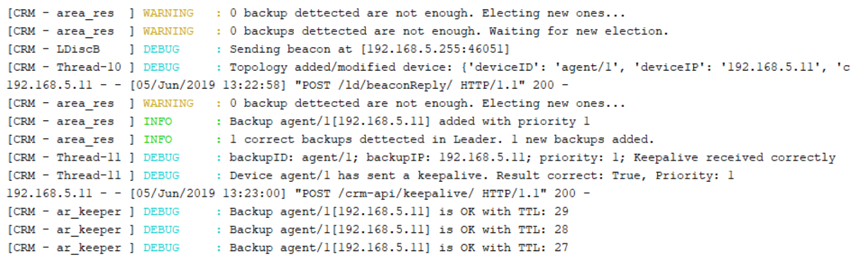
Screenshots	 `[CRM - Thread-7  ] ERROR     : Device is not a Leader, cannot perform a reelection in a non-leader device.`
	`192.168.56.1 - - [05/Jun/2019 13:14:05] "POST /crm-api/reelection HTTP/1.1" 401 -`
	Server response
	Code | Details
	401 | Error: UNAUTHORIZED
	Response body
	`{ "deviceID": "agent/0", "deviceIP": "" }`

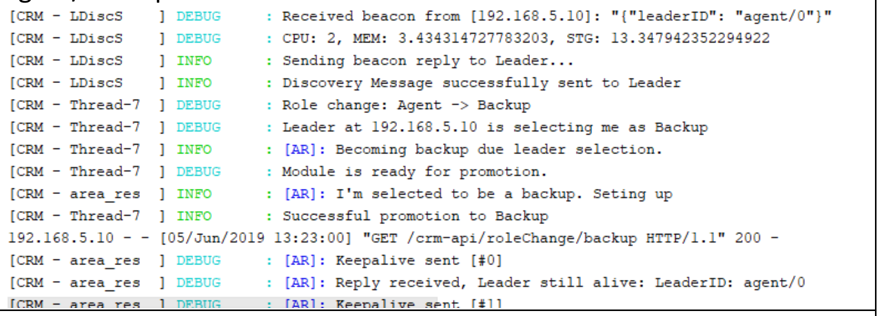
Observations	Screenshot at top is from log output of the agent, the one at the bottom is from the API dashboard at the agent. *Error was detected while the realization of this test that caused a different error code than the one expected, due to topology search before role checking. A patch was created to solve it.*

All tests have successfully passed, meaning that the basic standalone CRM can work and be deployed.

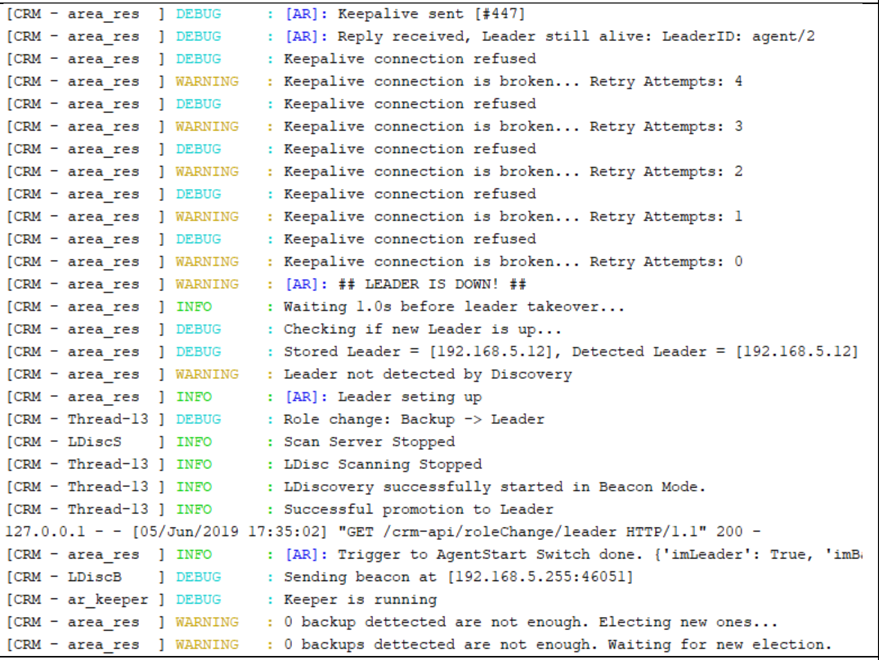
Area Resilience Tests

Leader promote a Backup	
Test Description	Leader select a device from the topology and promotes to Backup.
Expected Result	An agent is promoted to Backup and Keepalive protocol start sending messages while Leader check that Backup is active.
Test Result	**Success**

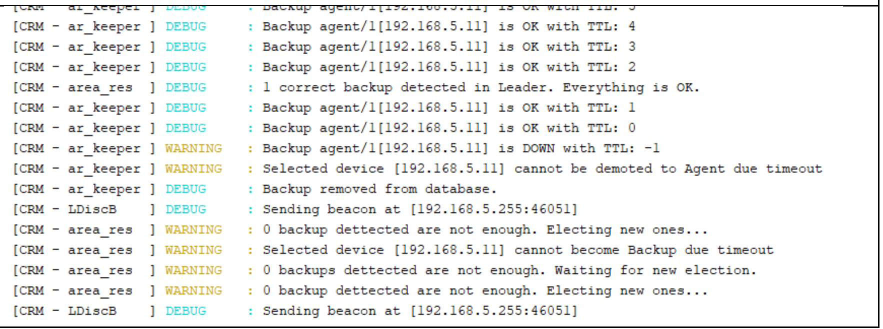
Screenshot	Leader view:
	 `[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...`
	`[CRM - area_res  ] WARNING   : 0 backups dettected are not enough. Waiting for new election.`
	`[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]`
	`[CRM - Thread-10 ] DEBUG     : Topology added/modified device: {'deviceID': 'agent/1', 'deviceIP': '192.168.5.11', 'c`
	`192.168.5.11 - - [05/Jun/2019 13:22:58] "POST /ld/beaconReply/ HTTP/1.1" 200 -`
	`[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...`
	`[CRM - area_res  ] INFO      : Backup agent/1[192.168.5.11] added with priority 1`
	`[CRM - area_res  ] INFO      : 1 correct backups dettected in Leader. 1 new backups added.`
	`[CRM - Thread-11 ] DEBUG     : backupID: agent/1; backupIP: 192.168.5.11; priority: 1; Keepalive received correctly`
	`[CRM - Thread-11 ] DEBUG     : Device agent/1 has sent a keepalive. Result correct: True, Priority: 1`
	`192.168.5.11 - - [05/Jun/2019 13:23:00] "POST /crm-api/keepalive/ HTTP/1.1" 200 -`
	`[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 29`
	`[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 28`
	`[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 27`

	Agent/Backup view: 
Observations	Discovery procedure is also validated with this test.

```
[CRM - LDiscS    ] DEBUG     : Received beacon from [192.168.5.10]: "{"leaderID": "agent/0"}"
[CRM - LDiscS    ] DEBUG     : CPU: 2, MEM: 3.434314727783203, STG: 13.347942352294922
[CRM - LDiscS    ] INFO      : Sending beacon reply to Leader...
[CRM - LDiscS    ] INFO      : Discovery Message successfully sent to Leader
[CRM - Thread-7  ] DEBUG     : Role change: Agent -> Backup
[CRM - Thread-7  ] DEBUG     : Leader at 192.168.5.10 is selecting me as Backup
[CRM - Thread-7  ] INFO      : [AR]: Becoming backup due leader selection.
[CRM - Thread-7  ] DEBUG     : Module is ready for promotion.
[CRM - area_res  ] INFO      : [AR]: I'm selected to be a backup. Seting up
[CRM - Thread-7  ] INFO      : Successful promotion to Backup
192.168.5.10 - - [05/Jun/2019 13:23:00] "GET /crm-api/roleChange/backup HTTP/1.1" 200 -
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#0]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#1]
```

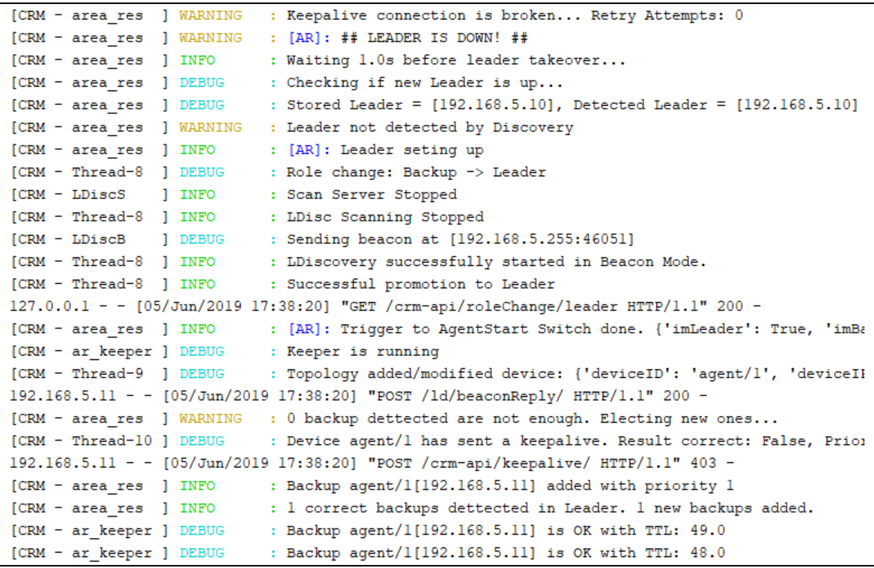
Leader failure	
Test Description	Leader fails and the backup become the new leader.
Expected Result	Backup become the Leader.
Test Result	**Success**
Screenshot	
Observations	No new backup is selected, as in the scenario there are no available devices.

```
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#447]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/2
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 3
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 2
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 1
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : Waiting 1.0s before leader takeover...
[CRM - area_res  ] DEBUG     : Checking if new Leader is up...
[CRM - area_res  ] DEBUG     : Stored Leader = [192.168.5.12], Detected Leader = [192.168.5.12]
[CRM - area_res  ] WARNING   : Leader not detected by Discovery
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-13 ] DEBUG     : Role change: Backup -> Leader
[CRM - LDiscS    ] INFO      : Scan Server Stopped
[CRM - Thread-13 ] INFO      : LDisc Scanning Stopped
[CRM - Thread-13 ] INFO      : LDiscovery successfully started in Beacon Mode.
[CRM - Thread-13 ] INFO      : Successful promotion to Leader
127.0.0.1 - - [05/Jun/2019 17:35:02] "GET /crm-api/roleChange/leader HTTP/1.1" 200 -
[CRM - area_res  ] INFO      : [AR]: Trigger to AgentStart Switch done. {'imLeader': True, 'imB
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
[CRM - ar_keeper ] DEBUG     : Keeper is running
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] WARNING   : 0 backups dettected are not enough. Waiting for new election.
```

Backup Failure	
Test Description	Backup fails and the Leader detect it.
Expected Result	The Leader remove the Backup from the database.
Test Result	**Success**
Screenshot	

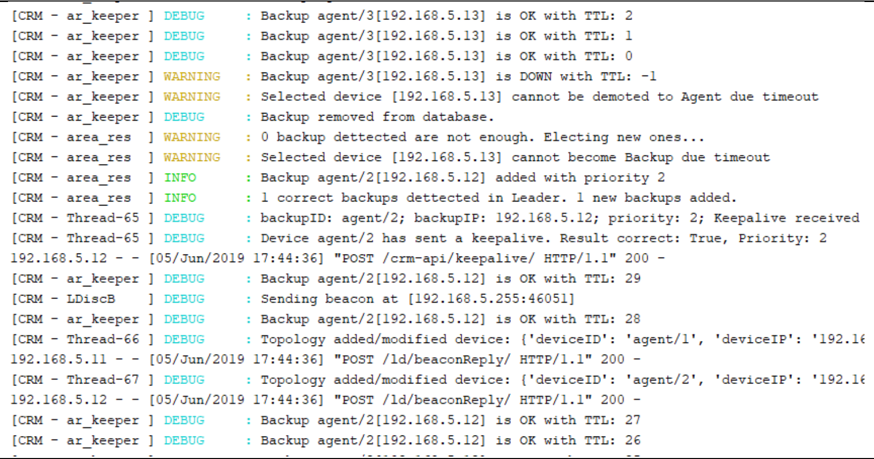
```
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 4
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 3
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 2
[CRM - area_res  ] DEBUG     : 1 correct backup detected in Leader. Everything is OK.
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 1
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 0
[CRM - ar_keeper ] WARNING   : Backup agent/1[192.168.5.11] is DOWN with TTL: -1
[CRM - ar_keeper ] WARNING   : Selected device [192.168.5.11] cannot be demoted to Agent due timeout
[CRM - ar_keeper ] DEBUG     : Backup removed from database.
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] WARNING   : Selected device [192.168.5.11] cannot become Backup due timeout
[CRM - area_res  ] WARNING   : 0 backups dettected are not enough. Waiting for new election.
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
```

Observations	Device demotion cannot be done due to the disconnection of the device. Device is not removed from the topology, but Leader is not affected, as selection cannot be done/not considered if a positive code is not given back from Backup.

Leader failure and new backup election	
Test Description	Leader fails and the Backup must detect it and take action.
Expected Result	Backup becomes the new Leader and selects an available device as new Backup.
Test Result	**Success**
Screenshot	

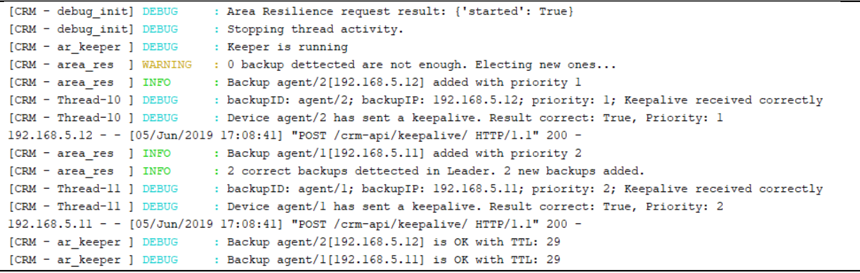
```
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : Waiting 1.0s before leader takeover...
[CRM - area_res  ] DEBUG     : Checking if new Leader is up...
[CRM - area_res  ] DEBUG     : Stored Leader = [192.168.5.10], Detected Leader = [192.168.5.10]
[CRM - area_res  ] WARNING   : Leader not detected by Discovery
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-8  ] DEBUG     : Role change: Backup -> Leader
[CRM - LDiscS    ] INFO      : Scan Server Stopped
[CRM - Thread-8  ] INFO      : LDisc Scanning Stopped
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
[CRM - Thread-8  ] INFO      : LDiscovery successfully started in Beacon Mode.
[CRM - Thread-8  ] INFO      : Successful promotion to Leader
127.0.0.1 - - [05/Jun/2019 17:38:20] "GET /crm-api/roleChange/leader HTTP/1.1" 200 -
[CRM - area_res  ] INFO      : [AR]: Trigger to AgentStart Switch done. {'imLeader': True, 'imB
[CRM - ar_keeper ] DEBUG     : Keeper is running
[CRM - Thread-9  ] DEBUG     : Topology added/modified device: {'deviceID': 'agent/1', 'deviceI
192.168.5.11 - - [05/Jun/2019 17:38:20] "POST /ld/beaconReply/ HTTP/1.1" 200 -
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - Thread-10 ] DEBUG     : Device agent/1 has sent a keepalive. Result correct: False, Prio
192.168.5.11 - - [05/Jun/2019 17:38:20] "POST /crm-api/keepalive/ HTTP/1.1" 403 -
[CRM - area_res  ] INFO      : Backup agent/1[192.168.5.11] added with priority 1
[CRM - area_res  ] INFO      : 1 correct backups dettected in Leader. 1 new backups added.
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 49.0
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 48.0
```

Observations	New active Backup send the Keepalive challenge before the Leader add it to the accepted database. At the next Keepalive it gets correctly the new value. Does not affect the Backup nor the Leader.

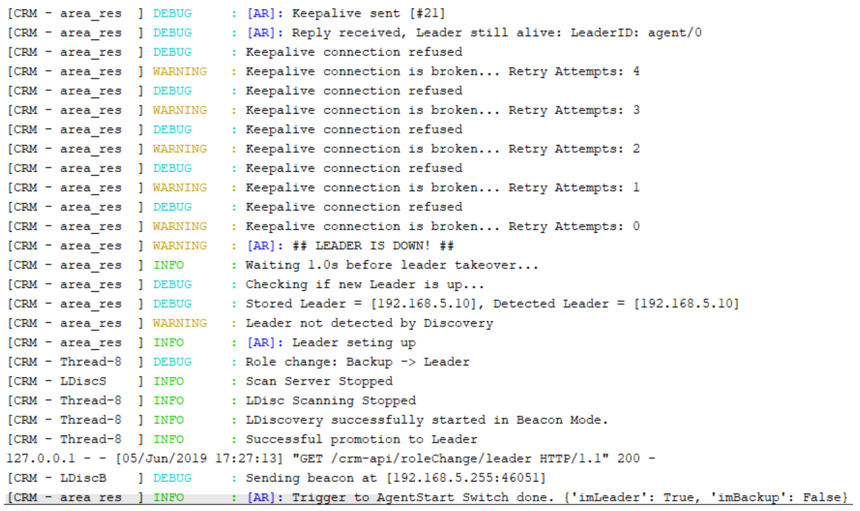
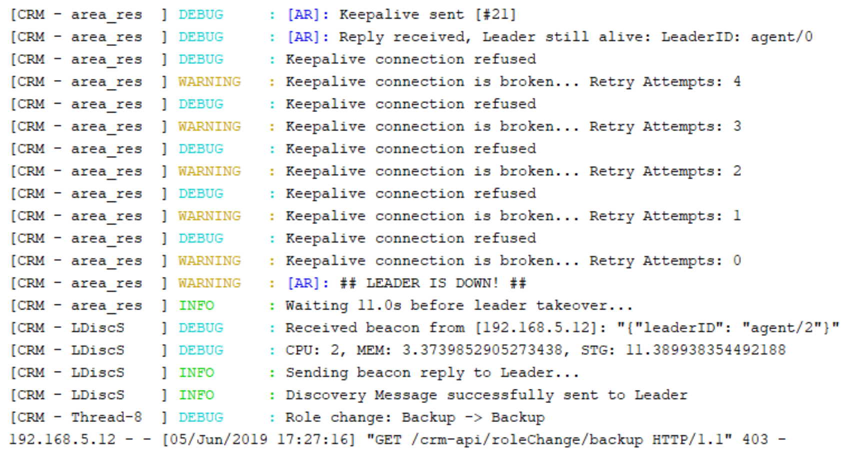
Backup failure (with new Backup)	
Test Description	Backup failure is detected by the Leader.
Expected Result	A new available device is promoted as Backup of the Leader.
Test Result	**Success**
Screenshot	

```
[CRM - ar_keeper ] DEBUG     : Backup agent/3[192.168.5.13] is OK with TTL: 2
[CRM - ar_keeper ] DEBUG     : Backup agent/3[192.168.5.13] is OK with TTL: 1
[CRM - ar_keeper ] DEBUG     : Backup agent/3[192.168.5.13] is OK with TTL: 0
[CRM - ar_keeper ] WARNING   : Backup agent/3[192.168.5.13] is DOWN with TTL: -1
[CRM - ar_keeper ] WARNING   : Selected device [192.168.5.13] cannot be demoted to Agent due timeout
[CRM - ar_keeper ] DEBUG     : Backup removed from database.
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] WARNING   : Selected device [192.168.5.13] cannot become Backup due timeout
[CRM - area_res  ] INFO      : Backup agent/2[192.168.5.12] added with priority 2
[CRM - area_res  ] INFO      : 1 correct backups dettected in Leader. 1 new backups added.
[CRM - Thread-65 ] DEBUG     : backupID: agent/2; backupIP: 192.168.5.12; priority: 2; Keepalive received
[CRM - Thread-65 ] DEBUG     : Device agent/2 has sent a keepalive. Result correct: True, Priority: 2
192.168.5.12 - - [05/Jun/2019 17:44:36] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG     : Backup agent/2[192.168.5.12] is OK with TTL: 29
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
[CRM - ar_keeper ] DEBUG     : Backup agent/2[192.168.5.12] is OK with TTL: 28
[CRM - Thread-66 ] DEBUG     : Topology added/modified device: {'deviceID': 'agent/1', 'deviceIP': '192.1
192.168.5.11 - - [05/Jun/2019 17:44:36] "POST /ld/beaconReply/ HTTP/1.1" 200 -
[CRM - Thread-67 ] DEBUG     : Topology added/modified device: {'deviceID': 'agent/2', 'deviceIP': '192.1
192.168.5.12 - - [05/Jun/2019 17:44:36] "POST /ld/beaconReply/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG     : Backup agent/2[192.168.5.12] is OK with TTL: 27
[CRM - ar_keeper ] DEBUG     : Backup agent/2[192.168.5.12] is OK with TTL: 26
```

Observations	Old device still not removed from topology. Demotion of the failed backup failed, but expected as the device is no more present on the network.

Multiple backups	
Test Description	Multiple backups can coexist in the same area with the same Leader.
Expected Result	Normal election and keepalive, with correct priority assignment.
Test Result	**Success**
Screenshot	
Observations	

```
[CRM - debug_init] DEBUG     : Area Resilience request result: {'started': True}
[CRM - debug_init] DEBUG     : Stopping thread activity.
[CRM - ar_keeper ] DEBUG     : Keeper is running
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] INFO      : Backup agent/2[192.168.5.12] added with priority 1
[CRM - Thread-10 ] DEBUG     : backupID: agent/2; backupIP: 192.168.5.12; priority: 1; Keepalive received correctly
[CRM - Thread-10 ] DEBUG     : Device agent/2 has sent a keepalive. Result correct: True, Priority: 1
192.168.5.12 - - [05/Jun/2019 17:08:41] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - area_res  ] INFO      : Backup agent/1[192.168.5.11] added with priority 2
[CRM - area_res  ] INFO      : 2 correct backups dettected in Leader. 2 new backups added.
[CRM - Thread-11 ] DEBUG     : backupID: agent/1; backupIP: 192.168.5.11; priority: 2; Keepalive received correctly
[CRM - Thread-11 ] DEBUG     : Device agent/1 has sent a keepalive. Result correct: True, Priority: 2
192.168.5.11 - - [05/Jun/2019 17:08:41] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG     : Backup agent/2[192.168.5.12] is OK with TTL: 29
[CRM - ar_keeper ] DEBUG     : Backup agent/1[192.168.5.11] is OK with TTL: 29
```

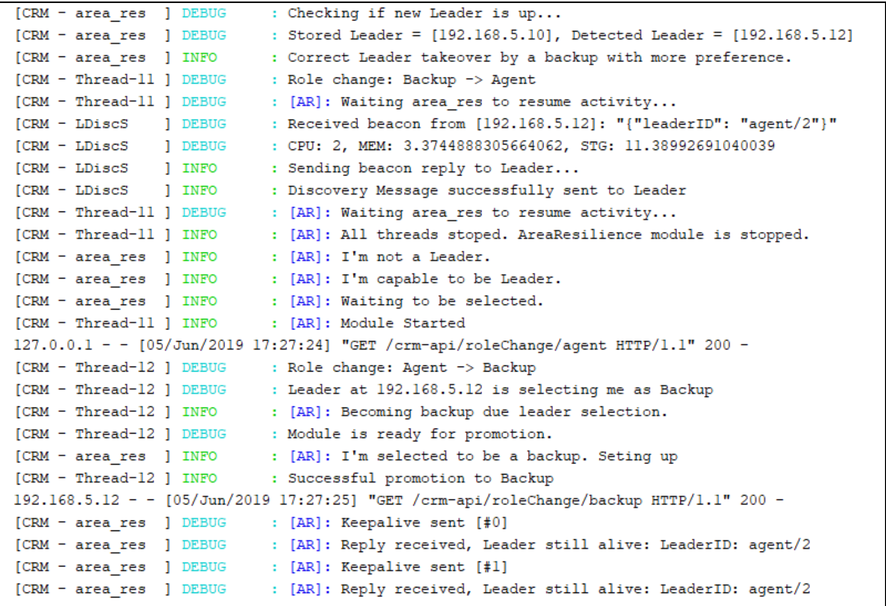
Leader failure and new backup election (with multiple active Backups)	
Test Description	Leader fails with multiple active backups.
Expected Result	The backup with greater preference becomes the new Leader while the other backups become normal agents (or Backups of the new Leader).
Test Result	**Success**
Screenshot	Backup with Priority = 1  Backup with Priority = 2 

Backup with Priority = 1

```
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#21]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 3
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 2
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 1
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : Waiting 1.0s before leader takeover...
[CRM - area_res  ] DEBUG     : Checking if new Leader is up...
[CRM - area_res  ] DEBUG     : Stored Leader = [192.168.5.10], Detected Leader = [192.168.5.10]
[CRM - area_res  ] WARNING   : Leader not detected by Discovery
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-8  ] DEBUG     : Role change: Backup -> Leader
[CRM - LDiscS    ] INFO      : Scan Server Stopped
[CRM - Thread-8  ] INFO      : LDisc Scanning Stopped
[CRM - Thread-8  ] INFO      : LDiscovery successfully started in Beacon Mode.
[CRM - Thread-8  ] INFO      : Successful promotion to Leader
127.0.0.1 - - [05/Jun/2019 17:27:13] "GET /crm-api/roleChange/leader HTTP/1.1" 200 -
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
[CRM - area res  ] INFO      : [AR]: Trigger to AgentStart Switch done. {'imLeader': True, 'imBackup': False}
```

Backup with Priority = 2

```
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#21]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 3
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 2
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 1
[CRM - area_res  ] DEBUG     : Keepalive connection refused
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : Waiting 11.0s before leader takeover...
[CRM - LDiscS    ] DEBUG     : Received beacon from [192.168.5.12]: "{"leaderID": "agent/2"}"
[CRM - LDiscS    ] DEBUG     : CPU: 2, MEM: 3.3739852905273438, STG: 11.389938354492188
[CRM - LDiscS    ] INFO      : Sending beacon reply to Leader...
[CRM - LDiscS    ] INFO      : Discovery Message successfully sent to Leader
[CRM - Thread-8  ] DEBUG     : Role change: Backup -> Backup
192.168.5.12 - - [05/Jun/2019 17:27:16] "GET /crm-api/roleChange/backup HTTP/1.1" 403 -
```



```
[CRM - area_res  ] DEBUG     : Checking if new Leader is up...
[CRM - area_res  ] DEBUG     : Stored Leader = [192.168.5.10], Detected Leader = [192.168.5.12]
[CRM - area_res  ] INFO      : Correct Leader takeover by a backup with more preference.
[CRM - Thread-11 ] DEBUG     : Role change: Backup -> Agent
[CRM - Thread-11 ] DEBUG     : [AR]: Waiting area_res to resume activity...
[CRM - LDiscS    ] DEBUG     : Received beacon from [192.168.5.12]: "{"leaderID": "agent/2"}"
[CRM - LDiscS    ] DEBUG     : CPU: 2, MEM: 3.3744888305664062, STG: 11.38992691040039
[CRM - LDiscS    ] INFO      : Sending beacon reply to Leader...
[CRM - LDiscS    ] INFO      : Discovery Message successfully sent to Leader
[CRM - Thread-11 ] DEBUG     : [AR]: Waiting area_res to resume activity...
[CRM - Thread-11 ] INFO      : [AR]: All threads stoped. AreaResilience module is stopped.
[CRM - area_res  ] INFO      : [AR]: I'm not a Leader.
[CRM - area_res  ] INFO      : [AR]: I'm capable to be Leader.
[CRM - area_res  ] INFO      : [AR]: Waiting to be selected.
[CRM - Thread-11 ] INFO      : [AR]: Module Started
127.0.0.1 - - [05/Jun/2019 17:27:24] "GET /crm-api/roleChange/agent HTTP/1.1" 200 -
[CRM - Thread-12 ] DEBUG     : Role change: Agent -> Backup
[CRM - Thread-12 ] DEBUG     : Leader at 192.168.5.12 is selecting me as Backup
[CRM - Thread-12 ] INFO      : [AR]: Becoming backup due leader selection.
[CRM - Thread-12 ] DEBUG     : Module is ready for promotion.
[CRM - area_res  ] INFO      : [AR]: I'm selected to be a backup. Seting up
[CRM - Thread-12 ] INFO      : Successful promotion to Backup
192.168.5.12 - - [05/Jun/2019 17:27:25] "GET /crm-api/roleChange/backup HTTP/1.1" 200 -
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#0]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/2
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#1]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/2
```

Observations	Active backups take more time to recover rather than non-active ones. Consider some wait time reduction. Higher priority number implies a lower preference.

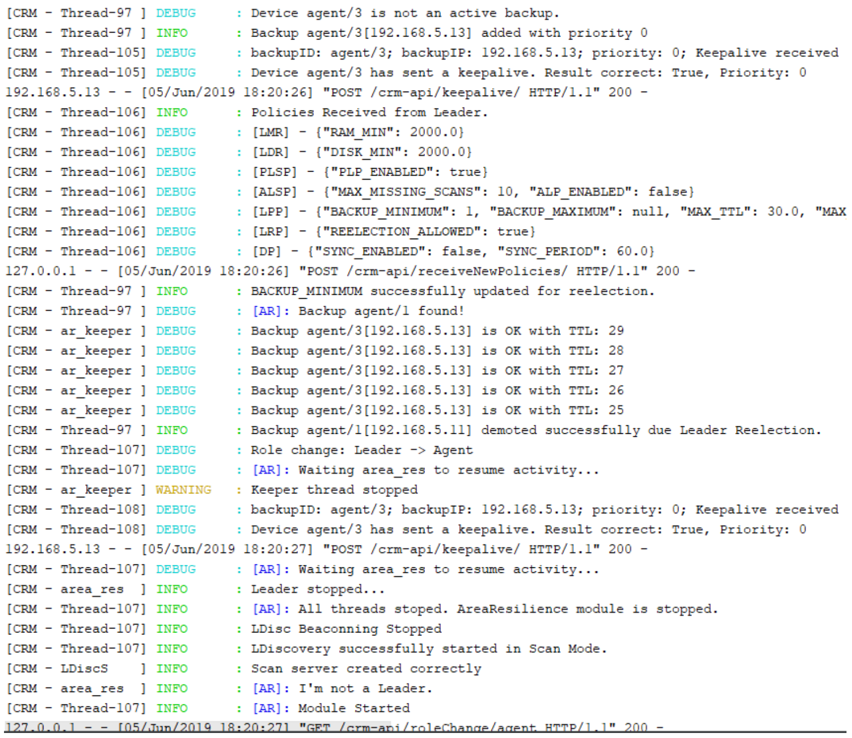
All the situations tested for the Area Resilience have successfully passed, some of them with known caveats that does not influence the expected result and the overall procedure is satisfactory.

Leader Reelection Tests

Leader reelection (new Leader not backup)	
Test Description	Agent selected to be the new leader is not an active Backup of the area.
Expected Result	The Agent becomes the new Leader and select a new Backup. Old leader and backup are demoted before being electable again.
Test Result	**Success**

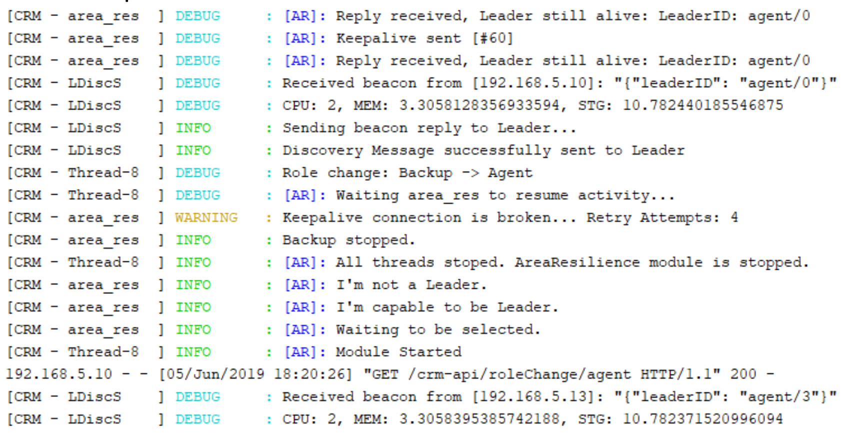
Screenshot

Leader view:



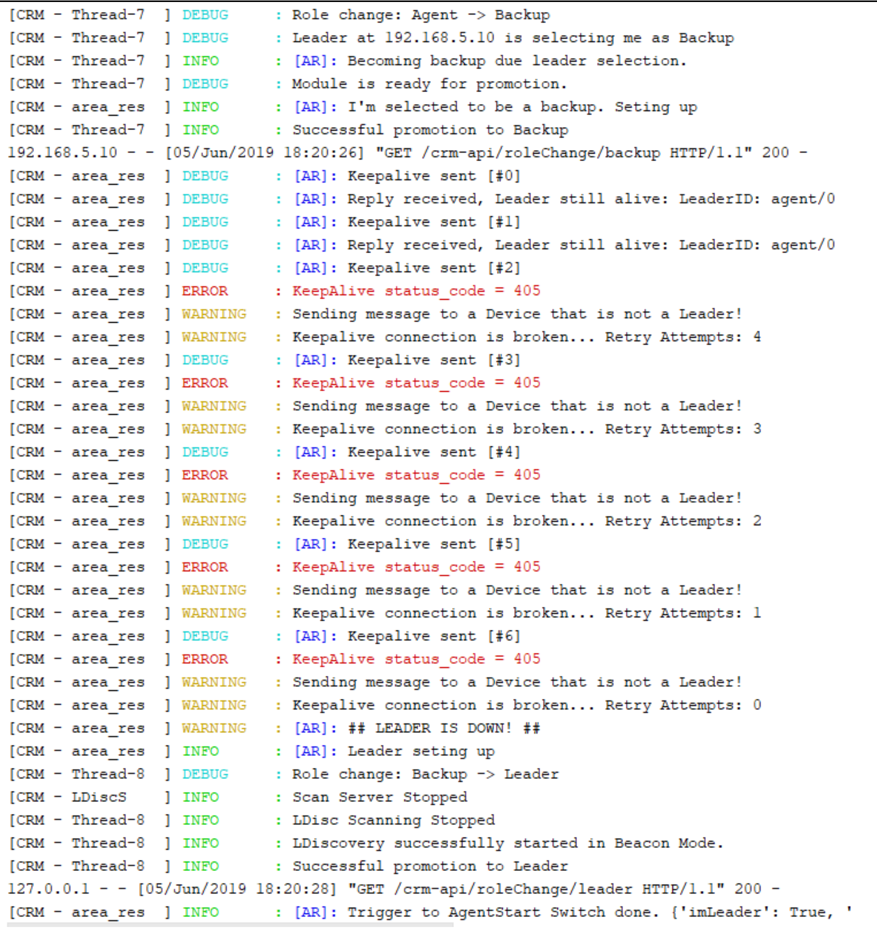
```
[CRM - Thread-97 ] DEBUG      : Device agent/3 is not an active backup.
[CRM - Thread-97 ] INFO       : Backup agent/3[192.168.5.13] added with priority 0
[CRM - Thread-105] DEBUG      : backupID: agent/3; backupIP: 192.168.5.13; priority: 0; Keepalive received
[CRM - Thread-105] DEBUG      : Device agent/3 has sent a keepalive. Result correct: True, Priority: 0
192.168.5.13 - - [05/Jun/2019 18:20:26] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - Thread-106] INFO       : Policies Received from Leader.
[CRM - Thread-106] DEBUG      : [LMR] - {"RAM_MIN": 2000.0}
[CRM - Thread-106] DEBUG      : [LDR] - {"DISK_MIN": 2000.0}
[CRM - Thread-106] DEBUG      : [PLSP] - {"PLP_ENABLED": true}
[CRM - Thread-106] DEBUG      : [ALSP] - {"MAX_MISSING_SCANS": 10, "ALP_ENABLED": false}
[CRM - Thread-106] DEBUG      : [LPP] - {"BACKUP_MINIMUM": 1, "BACKUP_MAXIMUM": null, "MAX_TTL": 30.0, "MAX
[CRM - Thread-106] DEBUG      : [LRP] - {"REELECTION_ALLOWED": true}
[CRM - Thread-106] DEBUG      : [DP] - {"SYNC_ENABLED": false, "SYNC_PERIOD": 60.0}
127.0.0.1 - - [05/Jun/2019 18:20:26] "POST /crm-api/receiveNewPolicies/ HTTP/1.1" 200 -
[CRM - Thread-97 ] INFO       : BACKUP_MINIMUM successfully updated for reelection.
[CRM - Thread-97 ] DEBUG      : [AR]: Backup agent/1 found!
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 29
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 28
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 27
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 26
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 25
[CRM - Thread-97 ] INFO       : Backup agent/1[192.168.5.11] demoted successfully due Leader Reelection.
[CRM - Thread-107] DEBUG      : Role change: Leader -> Agent
[CRM - Thread-107] DEBUG      : [AR]: Waiting area_res to resume activity...
[CRM - ar_keeper ] WARNING    : Keeper thread stopped
[CRM - Thread-108] DEBUG      : backupID: agent/3; backupIP: 192.168.5.13; priority: 0; Keepalive received
[CRM - Thread-108] DEBUG      : Device agent/3 has sent a keepalive. Result correct: True, Priority: 0
192.168.5.13 - - [05/Jun/2019 18:20:27] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - Thread-107] DEBUG      : [AR]: Waiting area_res to resume activity...
[CRM - area_res  ] INFO       : Leader stopped...
[CRM - Thread-107] INFO       : [AR]: All threads stoped. AreaResilience module is stopped.
[CRM - Thread-107] INFO       : LDisc Beaconning Stopped
[CRM - Thread-107] INFO       : LDiscovery successfully started in Scan Mode.
[CRM - LDiscS    ] INFO       : Scan server created correctly
[CRM - area_res  ] INFO       : [AR]: I'm not a Leader.
[CRM - Thread-107] INFO       : [AR]: Module Started
127.0.0.1 - - [05/Jun/2019 18:20:27] "GET /crm-api/roleChange/agent HTTP/1.1" 200 -
```

Old backup view:



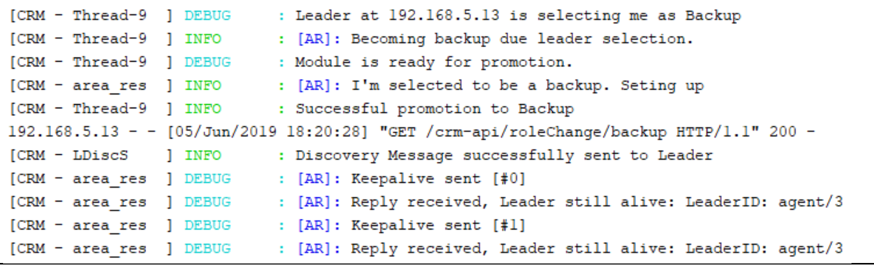
```
[CRM - area_res   ] DEBUG      : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res   ] DEBUG      : [AR]: Keepalive sent [#60]
[CRM - area_res   ] DEBUG      : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - LDiscS     ] DEBUG      : Received beacon from [192.168.5.10]: "{"leaderID": "agent/0"}"
[CRM - LDiscS     ] DEBUG      : CPU: 2, MEM: 3.3058128356933594, STG: 10.782440185546875
[CRM - LDiscS     ] INFO       : Sending beacon reply to Leader...
[CRM - LDiscS     ] INFO       : Discovery Message successfully sent to Leader
[CRM - Thread-8   ] DEBUG      : Role change: Backup -> Agent
[CRM - Thread-8   ] DEBUG      : [AR]: Waiting area_res to resume activity...
[CRM - area_res   ] WARNING    : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res   ] INFO       : Backup stopped.
[CRM - Thread-8   ] INFO       : [AR]: All threads stoped. AreaResilience module is stopped.
[CRM - area_res   ] INFO       : [AR]: I'm not a Leader.
[CRM - area_res   ] INFO       : [AR]: I'm capable to be Leader.
[CRM - area_res   ] INFO       : [AR]: Waiting to be selected.
[CRM - Thread-8   ] INFO       : [AR]: Module Started
192.168.5.10 - - [05/Jun/2019 18:20:26] "GET /crm-api/roleChange/agent HTTP/1.1" 200 -
[CRM - LDiscS     ] DEBUG      : Received beacon from [192.168.5.13]: "{"leaderID": "agent/3"}"
[CRM - LDiscS     ] DEBUG      : CPU: 2, MEM: 3.3058395385742188, STG: 10.782371520996094
```

New Leader view:



```
[CRM - Thread-7  ] DEBUG     : Role change: Agent -> Backup
[CRM - Thread-7  ] DEBUG     : Leader at 192.168.5.10 is selecting me as Backup
[CRM - Thread-7  ] INFO      : [AR]: Becoming backup due leader selection.
[CRM - Thread-7  ] DEBUG     : Module is ready for promotion.
[CRM - area_res  ] INFO      : [AR]: I'm selected to be a backup. Seting up
[CRM - Thread-7  ] INFO      : Successful promotion to Backup
192.168.5.10 - - [05/Jun/2019 18:20:26] "GET /crm-api/roleChange/backup HTTP/1.1" 200 -
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#0]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#1]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/0
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#2]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#3]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 3
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#4]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 2
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#5]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 1
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#6]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-8  ] DEBUG     : Role change: Backup -> Leader
[CRM - LDiscS    ] INFO      : Scan Server Stopped
[CRM - Thread-8  ] INFO      : LDisc Scanning Stopped
[CRM - Thread-8  ] INFO      : LDiscovery successfully started in Beacon Mode.
[CRM - Thread-8  ] INFO      : Successful promotion to Leader
127.0.0.1 - - [05/Jun/2019 18:20:28] "GET /crm-api/roleChange/leader HTTP/1.1" 200 -
[CRM - area_res  ] INFO      : [AR]: Trigger to AgentStart Switch done. {'imLeader': True, '
```

New backup view:



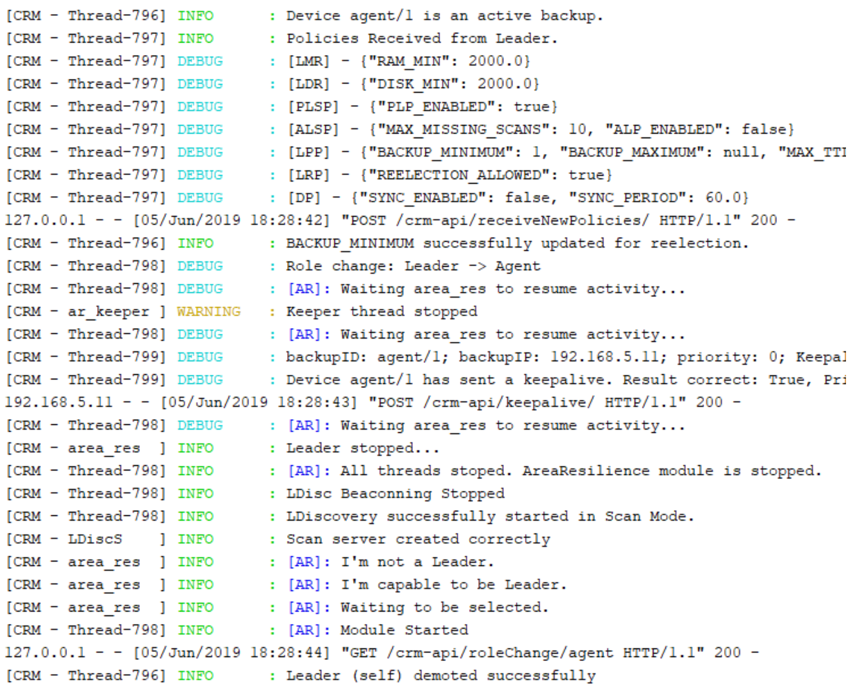
```
[CRM - Thread-9  ] DEBUG     : Leader at 192.168.5.13 is selecting me as Backup
[CRM - Thread-9  ] INFO      : [AR]: Becoming backup due leader selection.
[CRM - Thread-9  ] DEBUG     : Module is ready for promotion.
[CRM - area_res  ] INFO      : [AR]: I'm selected to be a backup. Seting up
[CRM - Thread-9  ] INFO      : Successful promotion to Backup
192.168.5.13 - - [05/Jun/2019 18:20:28] "GET /crm-api/roleChange/backup HTTP/1.1" 200 -
[CRM - LDiscS    ] INFO      : Discovery Message successfully sent to Leader
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#0]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/3
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#1]
[CRM - area_res  ] DEBUG     : [AR]: Reply received, Leader still alive: LeaderID: agent/3
```

Observations	

Leader reelection (new Leader active backup)	
Test Description	Agent selected to be the new leader is an active Backup of the area.
Expected Result	The Agent becomes the new Leader and select a new Backup. Old leader is demoted before being electable again. No active backup demotion is performed.
Test Result	**Success**

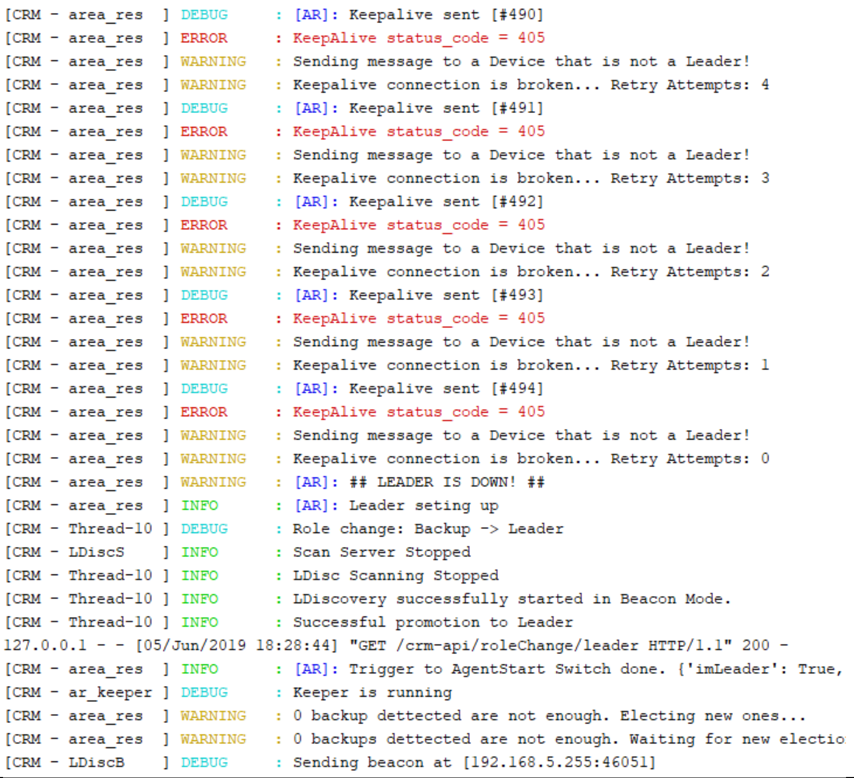
Screenshot

Leader view:



```
[CRM - Thread-796] INFO      : Device agent/1 is an active backup.
[CRM - Thread-797] INFO      : Policies Received from Leader.
[CRM - Thread-797] DEBUG     : [LMR] - {"RAM_MIN": 2000.0}
[CRM - Thread-797] DEBUG     : [LDR] - {"DISK_MIN": 2000.0}
[CRM - Thread-797] DEBUG     : [PLSP] - {"PLP_ENABLED": true}
[CRM - Thread-797] DEBUG     : [ALSP] - {"MAX_MISSING_SCANS": 10, "ALP_ENABLED": false}
[CRM - Thread-797] DEBUG     : [LPP] - {"BACKUP_MINIMUM": 1, "BACKUP_MAXIMUM": null, "MAX_TTI
[CRM - Thread-797] DEBUG     : [LRP] - {"REELECTION_ALLOWED": true}
[CRM - Thread-797] DEBUG     : [DP] - {"SYNC_ENABLED": false, "SYNC_PERIOD": 60.0}
127.0.0.1 - - [05/Jun/2019 18:28:42] "POST /crm-api/receiveNewPolicies/ HTTP/1.1" 200 -
[CRM - Thread-796] INFO      : BACKUP_MINIMUM successfully updated for reelection.
[CRM - Thread-798] DEBUG     : Role change: Leader -> Agent
[CRM - Thread-798] DEBUG     : [AR]: Waiting area_res to resume activity...
[CRM - ar_keeper ] WARNING   : Keeper thread stopped
[CRM - Thread-798] DEBUG     : [AR]: Waiting area_res to resume activity...
[CRM - Thread-799] DEBUG     : backupID: agent/1; backupIP: 192.168.5.11; priority: 0; Keepal
[CRM - Thread-799] DEBUG     : Device agent/1 has sent a keepalive. Result correct: True, Pri
192.168.5.11 - - [05/Jun/2019 18:28:43] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - Thread-798] DEBUG     : [AR]: Waiting area_res to resume activity...
[CRM - area_res  ] INFO      : Leader stopped...
[CRM - Thread-798] INFO      : [AR]: All threads stoped. AreaResilience module is stopped.
[CRM - Thread-798] INFO      : LDisc Beaconning Stopped
[CRM - Thread-798] INFO      : LDiscovery successfully started in Scan Mode.
[CRM - LDiscS    ] INFO      : Scan server created correctly
[CRM - area_res  ] INFO      : [AR]: I'm not a Leader.
[CRM - area_res  ] INFO      : [AR]: I'm capable to be Leader.
[CRM - area_res  ] INFO      : [AR]: Waiting to be selected.
[CRM - Thread-798] INFO      : [AR]: Module Started
127.0.0.1 - - [05/Jun/2019 18:28:44] "GET /crm-api/roleChange/agent HTTP/1.1" 200 -
[CRM - Thread-796] INFO      : Leader (self) demoted successfully
```

New Leader view:



```
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#490]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 4
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#491]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 3
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#492]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 2
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#493]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 1
[CRM - area_res  ] DEBUG     : [AR]: Keepalive sent [#494]
[CRM - area_res  ] ERROR     : KeepAlive status_code = 405
[CRM - area_res  ] WARNING   : Sending message to a Device that is not a Leader!
[CRM - area_res  ] WARNING   : Keepalive connection is broken... Retry Attempts: 0
[CRM - area_res  ] WARNING   : [AR]: ## LEADER IS DOWN! ##
[CRM - area_res  ] INFO      : [AR]: Leader seting up
[CRM - Thread-10 ] DEBUG     : Role change: Backup -> Leader
[CRM - LDiscS    ] INFO      : Scan Server Stopped
[CRM - Thread-10 ] INFO      : LDisc Scanning Stopped
[CRM - Thread-10 ] INFO      : LDiscovery successfully started in Beacon Mode.
[CRM - Thread-10 ] INFO      : Successful promotion to Leader
127.0.0.1 - - [05/Jun/2019 18:28:44] "GET /crm-api/roleChange/leader HTTP/1.1" 200 -
[CRM - area_res  ] INFO      : [AR]: Trigger to AgentStart Switch done. {'imLeader': True,
[CRM - ar_keeper ] DEBUG     : Keeper is running
[CRM - area_res  ] WARNING   : 0 backup dettected are not enough. Electing new ones...
[CRM - area_res  ] WARNING   : 0 backups dettected are not enough. Waiting for new electio
[CRM - LDiscB    ] DEBUG     : Sending beacon at [192.168.5.255:46051]
```

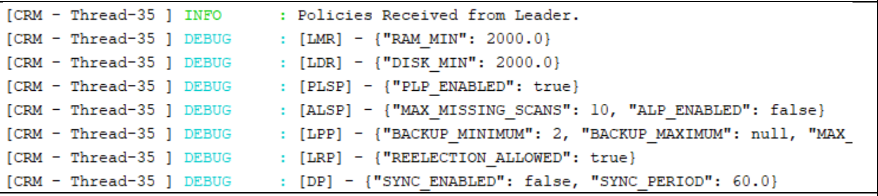
Observations

Keepalive reply gets a 405 code, corresponding to a device that is not an active Leader has been challenged. As the Leader is demoted but is still reachable by network (is not down), keepalive can reach out, but non positive reply is given.

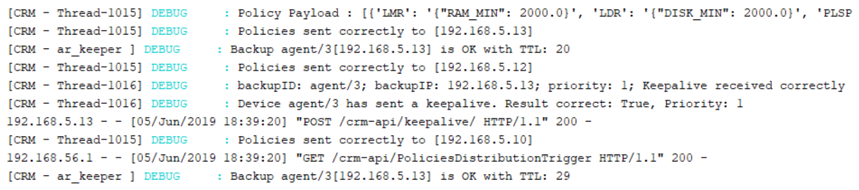
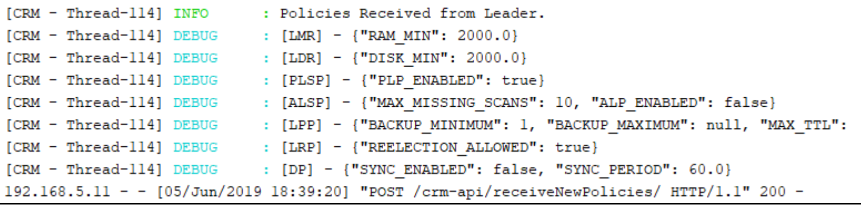
Finally, the Leader Reelection also work as intended on the proposed test scenarios with a satisfactory behavior.

8.3. Policies

Continuing with the verification of the CRM core functionalities, the other pillar is the Policies and its distribution into the system. As this functionality does not imply any situational case or role, the tests are simpler.

Receive new Policies	
Test Description	Agent receive policies from the Leader.
Expected Result	New policies are stored.
Test Result	**Success**
Screenshot	
Observations	

```
[CRM - Thread-35 ] INFO       : Policies Received from Leader.
[CRM - Thread-35 ] DEBUG      : [LMR] - {"RAM_MIN": 2000.0}
[CRM - Thread-35 ] DEBUG      : [LDR] - {"DISK_MIN": 2000.0}
[CRM - Thread-35 ] DEBUG      : [PLSP] - {"PLP_ENABLED": true}
[CRM - Thread-35 ] DEBUG      : [ALSP] - {"MAX_MISSING_SCANS": 10, "ALP_ENABLED": false}
[CRM - Thread-35 ] DEBUG      : [LPP] - {"BACKUP_MINIMUM": 2, "BACKUP_MAXIMUM": null, "MAX_
[CRM - Thread-35 ] DEBUG      : [LRP] - {"REELECTION_ALLOWED": true}
[CRM - Thread-35 ] DEBUG      : [DP] - {"SYNC_ENABLED": false, "SYNC_PERIOD": 60.0}
```

Policies Distribution	
Test Description	Leader spread the policies among all the Agents in the area.
Expected Result	Policies successfully setup.
Test Result	**Success**
Screenshot	Leader view:  Agent view: 
Observations	

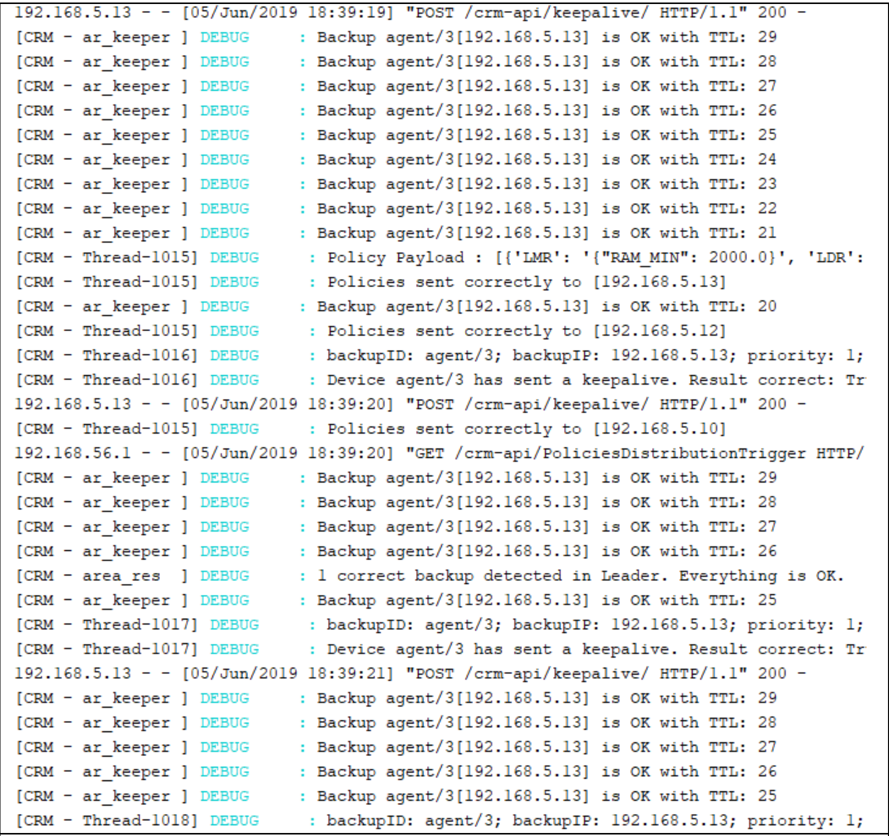
Leader view:

```
[CRM - Thread-1015] DEBUG     : Policy Payload : [{'LMR': '{"RAM_MIN": 2000.0}', 'LDR': '{"DISK_MIN": 2000.0}', 'PLSP
[CRM - Thread-1015] DEBUG     : Policies sent correctly to [192.168.5.13]
[CRM - ar_keeper ] DEBUG     : Backup agent/3[192.168.5.13] is OK with TTL: 20
[CRM - Thread-1015] DEBUG     : Policies sent correctly to [192.168.5.12]
[CRM - Thread-1016] DEBUG     : backupID: agent/3; backupIP: 192.168.5.13; priority: 1; Keepalive received correctly
[CRM - Thread-1016] DEBUG     : Device agent/3 has sent a keepalive. Result correct: True, Priority: 1
192.168.5.13 - - [05/Jun/2019 18:39:20] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - Thread-1015] DEBUG     : Policies sent correctly to [192.168.5.10]
192.168.56.1 - - [05/Jun/2019 18:39:20] "GET /crm-api/PoliciesDistributionTrigger HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG     : Backup agent/3[192.168.5.13] is OK with TTL: 29
```

Agent view:

```
[CRM - Thread-114] INFO       : Policies Received from Leader.
[CRM - Thread-114] DEBUG      : [LMR] - {"RAM_MIN": 2000.0}
[CRM - Thread-114] DEBUG      : [LDR] - {"DISK_MIN": 2000.0}
[CRM - Thread-114] DEBUG      : [PLSP] - {"PLP_ENABLED": true}
[CRM - Thread-114] DEBUG      : [ALSP] - {"MAX_MISSING_SCANS": 10, "ALP_ENABLED": false}
[CRM - Thread-114] DEBUG      : [LPP] - {"BACKUP_MINIMUM": 1, "BACKUP_MAXIMUM": null, "MAX_TTL":
[CRM - Thread-114] DEBUG      : [LRP] - {"REELECTION_ALLOWED": true}
[CRM - Thread-114] DEBUG      : [DP] - {"SYNC_ENABLED": false, "SYNC_PERIOD": 60.0}
192.168.5.11 - - [05/Jun/2019 18:39:20] "POST /crm-api/receiveNewPolicies/ HTTP/1.1" 200 -
```

Dynamic Policies on action (Area Resilience)	
Test Description	The Keepalive interval policy is modified in the area
Expected Result	The backup send keepalive challenge messages faster than before.
Test Result	**Success**

Screenshot	
Observations	

```
192.168.5.13 - - [05/Jun/2019 18:39:19] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 29
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 28
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 27
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 26
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 25
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 24
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 23
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 22
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 21
[CRM - Thread-1015] DEBUG      : Policy Payload : [{'LMR': '{"RAM_MIN": 2000.0}', 'LDR':
[CRM - Thread-1015] DEBUG      : Policies sent correctly to [192.168.5.13]
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 20
[CRM - Thread-1015] DEBUG      : Policies sent correctly to [192.168.5.12]
[CRM - Thread-1016] DEBUG      : backupID: agent/3; backupIP: 192.168.5.13; priority: 1;
[CRM - Thread-1016] DEBUG      : Device agent/3 has sent a keepalive. Result correct: Tr
192.168.5.13 - - [05/Jun/2019 18:39:20] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - Thread-1015] DEBUG      : Policies sent correctly to [192.168.5.10]
192.168.56.1 - - [05/Jun/2019 18:39:20] "GET /crm-api/PoliciesDistributionTrigger HTTP/
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 29
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 28
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 27
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 26
[CRM - area_res  ] DEBUG      : 1 correct backup detected in Leader. Everything is OK.
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 25
[CRM - Thread-1017] DEBUG      : backupID: agent/3; backupIP: 192.168.5.13; priority: 1;
[CRM - Thread-1017] DEBUG      : Device agent/3 has sent a keepalive. Result correct: Tr
192.168.5.13 - - [05/Jun/2019 18:39:21] "POST /crm-api/keepalive/ HTTP/1.1" 200 -
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 29
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 28
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 27
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 26
[CRM - ar_keeper ] DEBUG      : Backup agent/3[192.168.5.13] is OK with TTL: 25
[CRM - Thread-1018] DEBUG      : backupID: agent/3; backupIP: 192.168.5.13; priority: 1;
```

The result is that the policies design and implementation work as expected, without any other concern.

8.4. Dynamic Policies study

On the early state of the project, an emulation was conducted to study if applying dynamic rules into a F2C system was significant enough to create such mechanism or not. The selected mechanism to conduct the experiment was the Control Data Replication inside the Leader Protection. Given the applied policy in a specific scenario, the expected behavior should be different. These different designed strategies are explained with more detail in section 5.1.2. Control Data Replication.

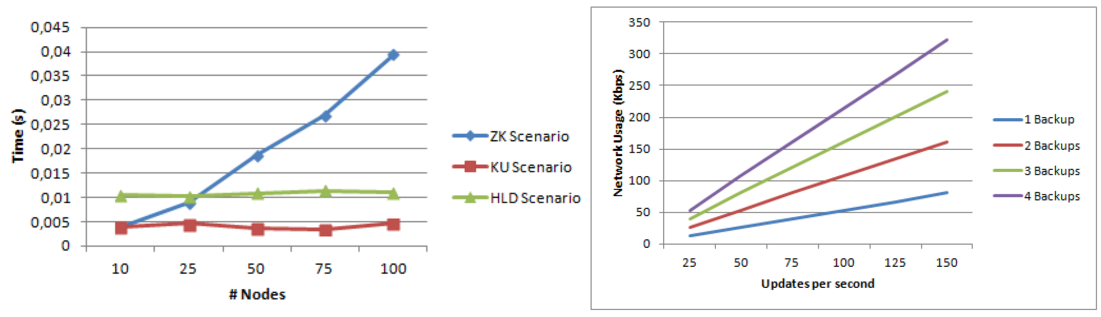


Fig. 29 Experiment Results: Left) Database update over number of node. Right) Network usage overhead

The experiment setup is a scenario with virtual machines as nodes, each with a simple client to reply to the discovery messages. A master node, emulating a Leader, is in charge of sending messages to the devices and update the database. To compare between different strategies, the used parameter is the time taken to fill correctly the database with the state of the area when a failure of the Leader is induced.

Results shown in Fig. 29 verify that the enforcement of different rules have an overall impact on the system and being able to tune the rules is a desirable functionality. On the left side, it can be appreciated that the Zero-Knowledge (ZK) strategy is good at small scenarios, but keeps increasing when new devices are added. Keep-Updating (KU) or High-Layer Download (HLD) are preferable strategies in large scenarios. However, at the right side the network impact is analyzed, showing that in an area with multiple backups, the KU strategy impact the network bandwidth incrementally only for this duty.

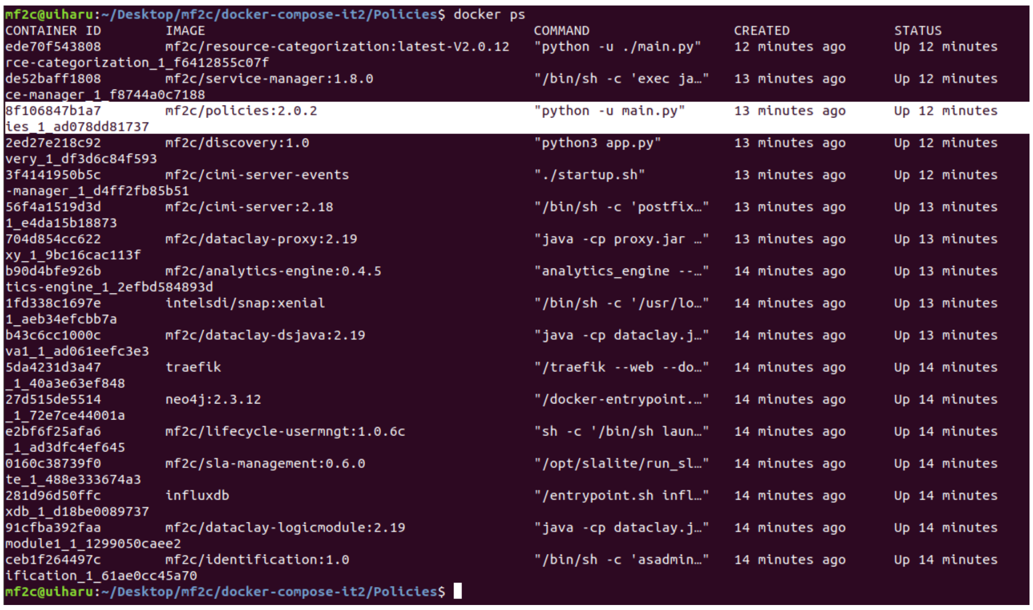
As a conclusion, it has been proven that there is a real necessity to provide dynamic policies to F2C systems. CRM already includes the distribution and the possibility to change policies on real-time and spread them into the area.

8.5. Validation in the mF2C Use Case

mF2C is an ongoing project, currently ending the development stage to enter into the use cases integration. The implications are that some procedures are still changing or still to be implemented.

The project has been already incorporated into the project with successful results, as is actually working without any integration error at this moment. Recently, the project is already testing the formation of the areas and the failure mechanisms, procedures provided by the integrated CRM along other modules.

Besides the individual testing performed by members of the project consortium, the project integration has been also validated in a local instance of the mF2C agent inside a virtual machine. In Fig. 30, the Policies block (name of the CRM module inside the mF2C architecture) is working and the agent starts as expected.



```
mf2c@uiharu:~/Desktop/mf2c/docker-compose-it2/Policies$ docker ps
CONTAINER ID        IMAGE                                         COMMAND                  CREATED             STATUS
ede70f543808        mf2c/resource-categorization:latest-V2.0.12   "python -u ./main.py"    12 minutes ago      Up 12 minutes
rce-categorization_1_f6412855c07f
de52baff1808        mf2c/service-manager:1.8.0                    "/bin/sh -c 'exec ja…"   13 minutes ago      Up 12 minutes
ce-manager_1_f8744a0c7188
8f106847b1a7        mf2c/policies:2.0.2                           "python -u main.py"      13 minutes ago      Up 12 minutes
ies_1_ad078dd81737
2ed27e218c92        mf2c/discovery:1.0                            "python3 app.py"         13 minutes ago      Up 12 minutes
very_1_df3d6c84f593
3f4141950b5c        mf2c/cimi-server-events                       "./startup.sh"           13 minutes ago      Up 12 minutes
-manager_1_d4ff2fb85b51
56f4a1519d3d        mf2c/cimi-server:2.18                         "/bin/sh -c 'postfix…"   13 minutes ago      Up 13 minutes
1_e4da15b18873
704d854cc622        mf2c/dataclay-proxy:2.19                      "java -cp proxy.jar …"   13 minutes ago      Up 13 minutes
xy_1_9bc16cac113f
b90d4bfe926b        mf2c/analytics-engine:0.4.5                   "analytics_engine --…"   14 minutes ago      Up 13 minutes
tics-engine_1_2efbd584893d
1fd338c1697e        intelsdi/snap:xenial                          "/bin/sh -c '/usr/lo…"   14 minutes ago      Up 13 minutes
1_aeb34efcbb7a
b43c6cc1000c        mf2c/dataclay-dsjava:2.19                     "java -cp dataclay.j…"   14 minutes ago      Up 13 minutes
va1_1_ad061eefc3e3
5da4231d3a47        traefik                                       "/traefik --web --do…"   14 minutes ago      Up 14 minutes
_1_40a3e63ef848
27d515de5514        neo4j:2.3.12                                  "/docker-entrypoint.…"   14 minutes ago      Up 14 minutes
_1_72e7ce44001a
e2bf6f25afa6        mf2c/lifecycle-usermngt:1.0.6c                "sh -c '/bin/sh laun…"   14 minutes ago      Up 14 minutes
_1_ad3dfc4ef645
0160c38739f0        mf2c/sla-management:0.6.0                     "/opt/slalite/run_sl…"   14 minutes ago      Up 14 minutes
te_1_488e333674a3
281d96d50ffc        influxdb                                      "/entrypoint.sh infl…"   14 minutes ago      Up 14 minutes
xdb_1_d18be0089737
91cfba392faa        mf2c/dataclay-logicmodule:2.19                "java -cp dataclay.j…"   14 minutes ago      Up 14 minutes
module1_1_1299050caee2
ceb1f264497c        mf2c/identification:1.0                       "/bin/sh -c 'asadmin…"   14 minutes ago      Up 14 minutes
ification_1_61ae0cc45a70
mf2c@uiharu:~/Desktop/mf2c/docker-compose-it2/Policies$
```

Fig. 30 mF2C docker internal deployment of the Agent

09.FUTURE WORK

Even if this project covers all of the proposed objectives, there are more improvements and added functionalities that can be done that emerged during the development of the project. On this section, a list of topics or ideas to be developed in a continuation of the project are presented.

- **Security**:

  All the communications between devices are insecure; no authentication nor encryption is provided. This is a serious thread, specially to the protection of the area and the policies distribution mechanisms, that take control of the behavior of the device.

  To address the encryption of the data transferred device-to-device, a simple TLS connection should be enough. However, authentication is required to make it possible. Devices need to trust each other, specially the Leader that coordinates and sends instructions.

  As a possible solution, the traditional certification process where a Certificate Authority (CA) issues certificates to trusted domains and users can trust them can be implemented, as in the mF2C project is the followed strategy [51]. However, in a mobile and heterogeneous scenario, is not the best way to provide authentication and security to edge devices. Another possible solution is the use of the Blockchain as a tool to distribute along all the devices rules and authentication, as explained in the next point.

- **Blockchain incorporation**:

  Blockchain [52] [53] is commonly defined as a data structure distributed across multiple peers. The data structure is formed by a sequenced list of blocks, with data structured in transactions. Blocks are "chained" with the anterior block using a hash as pointer [54]. Moreover, Blockchain also refers to the networking architecture and consensus algorithm that provides peer-to-peer (P2P) data distribution in a trusted and consensuated way. Its use has been popularized with the appearance of Bitcoin, the anonymous cryptocurrency over blockchain [55]. One drawback of this technology is that it can be only used is some specific scenarios, very well described in [56].

  Using blockchain itself will require a set of modifications on the data format stored, how is managed and the consensus algorithm used. For instance, adding new policies to the system will require to modify how this information is stored and interpreted and most important, how the device can trust that the other device is allowed to change information in the system.

  One possible solution here is using the Secure Edge Network (SEN) proposal [57]. SEN is a framework over blockchain that stablishes a set of rules that specifies the permissions on the system. Moreover, this framework is designed to be an optimal solution on edge deployment as a transversal solution for authentication across all the devices besides their capabilities.

- **Machine Learning on Policies:**

  The dynamic policies proposed are very static in the sense of even if are automatic, still there is not any learning on which policies are the best in different instants of time or status of the system.

  Applying machine learning to the policies will suppose a next step on the system performance and better use of its resources.

- **Module Integration:**

  The integration with other applications require that an entire sub-module is substituted with a specific code, which is not bad at all. An extensive implementation of the API covering most of the CRM functions and allowing some callbacks to be setup will make even more independent and closed the solution.

- **Code Re-factorization**:

  Remove some of the unused code, optimization tasks and other minor actions in the overall structure are required, caused in part by the incremental development and proof-of-concept product philosophy.

10.CONCLUSIONS

To conclude the project, the proposed Control Resilience Management (CRM) design and proof-of-concept implementation has successfully accomplished all the proposed objectives, and has been validated into a real F2C application.

Providing resilience in such described scenario is a difficult task and requires a lot of different procedures. CRM can solve some of the current challenges regarding the resilience of the system, allowing mobility of the devices without the degradation of system, and benefit F2C applications, just integrating the solution into their architectures. The dynamic policies designed can also bring more control over the system, without relaunching applications or stopping whole clusters. Together, they suppose a powerful tool.

In specific, all the objectives that have been proposed in the project are accomplished:

- The project has provided the design and implementation of a closed solution that provides resilience over a F2C architecture, with a main focus on the edge. This solution works as a standalone module that can be used without other applications and can be integrated.

- The design and proof-of-concept implementation of protection mechanisms for the Leader.

- The design and proof-of-concept implementation of the different policies, related with the resilience of the F2C scenario.

- Validated into a real F2C application, integrating the solution and fulfilling the expected functionalities, with the required changes.

The proposed architecture, compliant with the F2C definitions, is completely modular thanks to the interface specification. This brings the added value of changing any of the internal components with a different one without an extensive integration task. A new CRM implementation can be done with other procedures or added functionalities to adapt the solution to the required necessities.

As F2C is an evolving concept, new paradigms will arrive. The CRM design can be used as the structural base for newer functionalities. In the standalone mode or the integrated module, the CRM can be adapted or increased from the open source code provided, or deployed with the standard version.

We know for sure that scenarios where the edge has more relevance are the future. Contributing on the creation of new blocks into the overall wall is a fulfilling task and a great responsibility, that this project has the privilege to participate in.

11.ACKNOWLEDGEMENTS

This project has been possible in part thanks to the guidance and comments from Xavi Masip-Bruin, director of “Centre de Recerca Avançada en Arquitectures de Xarxa (CRAAX)”[9] research group, professor at Department of Computer Architecture in Universitat Politèncica de Catalunya (UPC) and thesis supervisor; and Beatriz Otero Calviño, director of the Department of Computer Architecture in Universitat Politèncica de Catalunya (UPC) and thesis tutor.

Special thanks to Sergi Sanchez López, Eva Marín Tordera, and Jordi Garcia Almiñana, professors at Department of Computer Architecture in Universitat Politèncica de Catalunya (UPC) and researchers at CRAAX, for the comments on the project.

Thanks to Pau Marcer Albareda, master student and cybersecurity researcher at CRAAX, for the comments on the project and the security counsels, especially on the Blockchain opportunities of the project.

Thanks to the members of the mF2C European Project consortium for the help and implication on the integration of the project into the mF2C Agent.

Finally, thanks to the GUAU[10] National Project and the CRAAX research group for the provided infrastructure for the development and testing of the project.

[9] CRAAX website: https://craax.upc.edu
[10] Project code reference: TEC2015-66220-R

12.REFERENCES

[1] D. Evans, "The Internet of Everything; How More Relevant and Valuable Connections Will Change the World," 2012. [Online]. Available: https://www.cisco.com/c/dam/global/en_my/assets/ciscoinnovate/pdfs/IoE.pdf. [Accessed 23 May 2019].

[2] K. Ashton, "That 'Internet of Things' Thing," *RFID Journal,* p. 1, 2009.

[3] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017-2022," 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

[4] F. Bonomi, R. Milito, J. Zhu and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," *Proceedings of the first edition of the MCC workshop on Mobile cloud computing,* pp. 13-16, 2012.

[5] F. Bonomi, R. Milito, P. Natarajan and J. Zhu, "Fog Computing: A Platform for Internet," in *Big data and internet of things: A roadmap for smart environments*, Springer, 2014, pp. 169-186.

[6] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A survey on sensor networks," in *IEEE Communications magazine*, vol. 40, IEEE, 2002, pp. 102-114.

[7] B. Parno and A. Perrig, "Challenges in securing vehicular networks," in *Workshop on hot topics in networks (HotNets-IV)*, Maryland, USA, 2005, pp. 1-6.

[8] X. Yang, L. Liu, N. H. Vaidya and F. Zhao, "A vehicle-to-vehicle communication protocol for cooperative collision warning," *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.,* pp. 114-123, 2004.

[9] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao and L. Zhao, "Vehicle-to-Everything (v2x) Services Supported by LTE-Based Systems and 5G," *IEEE Communications Standards Magazine,* vol. 1, no. 2, pp. 70-76, 2017.

[10] D. Linthicum, "Edge computing vs. fog computing: Definitions and enterprise uses," [Online]. Available: https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html. [Accessed 4 May 2019].

[11] OpenFog Consortium, "OpenFog Architecture Overview," 2016.

[12] E. Marín-Tordera, X. Masip-Bruin, J. García-Almiñana, A. Jukan, G.-J. Ren and J. Zhu, "Do we all really know what a Fog Node is? Current trends towards an open definition," *Computer Communications,* vol. 109, pp. 117-130, 2017.

[13] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan and G.-J. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications,* vol. 23, no. 5, pp. 120-128, 2016.

[14] W. Ramírez, X. Masip-Bruin, E. Marin-Tordera, V. B. C. Souza, A. Jukan, G.-J. Ren and O. G. de Dios, "Evaluating the benefits of combined and continuous Fog-to-Cloud architectures," *Computer Communications,* vol. 113, pp. 43-52, 2017.

[15] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong and J. C. Zhang, "What Will 5G Be?," *IEEE Journal on Selected Areas in Communications,* vol. 32, no. 6, pp. 1065 - 1082, 2014.

[16] European Commission, "5G for Europe Action Plan," [Online]. Available: https://ec.europa.eu/digital-single-market/en/5g-europe-action-plan. [Accessed 1 March 2019].

[17] Generalitat de Catalunya, "The ambulance of the future," *.catalonia Land of digital revolutionaries,* pp. 64 - 69, 2019.

[18] V. B. Souza, X. Masip-Bruin, E. Marín-Tordera, W. Ramírez and S. Sanchez, "Towards distributed service allocation in fog-to-cloud (F2C) scenarios," *2016 IEEE global communications conference (GLOBECOM),* pp. 1-6, 2016.

[19] "Horizon 2020 - Work Programme 2018-2020: 5.i. Information and Communication Technologies," 13 November 2018. [Online]. Available: http://ec.europa.eu/research/participants/data/ref/h2020/wp/2018-2020/main/h2020-wp1820-leit-ict_en.pdf.

[20] European Commission, "CORDIS | H2020: Towards an Open, Secure, Decentralized and Coordinated Fog-to-Cloud Management Ecosystem," [Online]. Available: https://cordis.europa.eu/project/rcn/206164/factsheet/en. [Accessed 14 Feb 2019].

[21] mF2C Consortium, "Technical Whitepaper mF2C Project," 2018.

[22] mF2C Consortium, "mF2C D2.7 mF2C Architecture (IT-2)," 2019.

[23] OpenFog Consortium, "OpenFog Reference Architecture for Fog Computing," 2017.

[24] M. Shirali, A. H. Toroghi and M. Vojdani, "Leader election algorithms: History and novel schemes," *2008 Third International Conference on Convergence and Hybrid Information Technology,* vol. 1, pp. 1001 - 1006, 2008.

[25] Atlassian, "ZooKeeper Wiki," [Online]. Available: https://cwiki.apache.org/confluence/display/ZOOKEEPER/Index.

[26] Apache Foundation, "ZooKeeper: Overview," [Online]. Available: https://zookeeper.apache.org/doc/r3.5.5/zookeeperOver.html. [Accessed 17 March 2019].

[27] Z. Rejiba, X. Masip-Bruin, A. Jurnet, E. Marin-Tordera and G.-J. Ren, "F2C-aware: Enabling discovery in Wi-Fi-powered fog-to-cloud (F2C) systems," *2018 6th IEEE International*

Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 113 - 116, 2018.

[28] OpenFog Consortium, "OpenFog Consortium Use Cases," [Online]. Available: https://www.openfogconsortium.org/resources/#use-cases. [Accessed 1 March 2019].

[29] Apache Software Foundation, "Apache Hadoop Wiki," [Online]. Available: https://wiki.apache.org/hadoop. [Accessed 5 March 2019].

[30] A. Lakshman and P. Malik, "Cassandra - A Decentralized Structured Storage System," *ACM SIGOPS Operating Systems Review,* vol. 44, no. 2, pp. 35-40, 2010.

[31] J. Martí, A. Queralt, D. Gasull, A. Barceló, J. J. Costa and T. Cortes, "Dataclay: A distributed data store for effective inter-player data," in *Journal of Systems and Software*, vol. 131, Elsevier, 2017, pp. 129-145.

[32] "Python.org," [Online]. Available: https://www.python.org/.

[33] "docker," [Online]. Available: https://www.docker.com/.

[34] "Docker Overview," [Online]. Available: https://docs.docker.com/engine/docker-overview/. [Accessed 5 March 2019].

[35] "Swagger UI," [Online]. Available: https://swagger.io/tools/swagger-ui/.

[36] "Flask-RESTPlus," [Online]. Available: https://flask-restplus.readthedocs.io/en/stable/. [Accessed 14 March 2019].

[37] "logging documentation," [Online]. Available: https://docs.python.org/3/library/logging.html. [Accessed 5 March 2019].

[38] "colorlog documentation," [Online]. Available: https://pypi.org/project/colorlog/. [Accessed 15 March 2019].

[39] "psutil documentation," [Online]. Available: https://pypi.org/project/psutil/. [Accessed 5 March 2019].

[40] M. Rehkopf @Maxrehallday, "What is Kanban? - Agile Coach Ep. 11 (2019)," Atlassian Agile Coach, 2019.

[41] "Trello," [Online]. Available: https://trello.com.

[42] "Git," [Online]. Available: https://git-scm.com/.

[43] "GitHub," [Online]. Available: https://github.com.

[44] "GitLab," [Online]. Available: https://gitlab.com/.

[45] "Slack," [Online]. Available: https://slack.com.

[46] "Waffle," [Online]. Available: http://waffle.io/.

[47] "About GitHub Project Boards," [Online]. Available: https://help.github.com/en/articles/about-project-boards. [Accessed 20 March 2019].

[48] "Travis CI," [Online]. Available: https://travis-ci.org/.

[49] Scrum.org, "What is Scrum?," [Online]. Available: https://www.scrum.org/resources/what-is-scrum.

[50] Internet Engineering Task Force (IETF), "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content," June 2014. [Online]. Available: https://tools.ietf.org/html/rfc7231#section-6.

[51] mF2C Consortium, "D2.5 mF2C Security/Privacy Requirements and Features (IT-2)," 2017.

[52] M. E. Peck, "Blockchains: How They Work and Why They'll Change the World," 28 September 2017. [Online]. Available: https://spectrum.ieee.org/computing/networks/blockchains-how-they-work-and-why-theyll-change-the-world. [Accessed 27 May 2019].

[53] Z. Zheng, S. Xie, H. Dai, X. Chen and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," *2017 IEEE International Congress on Big Data (BigData Congress),* pp. 557-564, 2017.

[54] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso and P. Rimba, "A Taxanomy of Blockchain-Based Systems for Architecture Design," *2017 IEEE International Conference on Software Architecture (ICSA),* pp. 243-252, 2017.

[55] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[56] K. Wüst and A. Gervais, "Do you need a Blockchain?," *2018 Crypto Valley Conference on Blockchain Technology (CVCBT),* pp. 45-54, 2018.

[57] P. Marcer Albareda, "Fog - Applying blockchain to secure a distributed set of clusters," Universitat Politècnica de Catalunya, To be published on July 2019.