

A CLOUD-BASED EMERGENCY SYSTEM FOR SMART ENHANCED LIVING
ENVIRONMENTS

by

YANNIS NIKOLOUDAKIS

Technological Educational Institute of Crete, 2017

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING

SCHOOL OF ENGINEERING

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF CRETE

2017

Approved by:

Professor

Evangelos Pallis



Abstract

The combination of Assisted Living (AL) and Ambient Intelligence (AmI) paradigms, has given birth to a multidisciplinary scientific/technological domain, known as the “Ambient Assisted Living” (AAL). Its aim is to improve the quality of citizen’s life through context-aware environments that can be adapted to each individual’s requirements for improving his/her daily life, especially those with physical/mental disabilities. Enhanced with state-of-the art Information & Communication Technologies (ICTs), AAL provides the means for better, safer and more secure living environments by monitoring citizen’s activities, gathering/processing and analyzing real-time data related to his health/behavior, and proactively alarming (in case of emergency) those in charge (e.g. doctors, physicians, first responders, etc.) and/or his family members. For example, people who suffer from cognitive impairment diseases, such as Alzheimer’s, dementia, cerebral palsy etc., need to be constantly monitored about their location in space, since it is very common for this kind of people to leave their premises and be unable to return. In such cases, AAL may assist persons-in-charge (e.g. house-nurses, family members, neighbors, etc.) by informing them about the patient’s exact position, or by proactively alarming first-responders (e.g. caretakers, doctors, hospital, police, etc.) in case of a potential threats/emergency.

Since the majority of AAL services and applications deal with large volume of data that require fast processing and resource-demanding infrastructure for rapid analysis and response/results, typical/common home-based computing platforms become obsolete, while sophisticated installations prove to be too costly. Thus, the Cloud Computing paradigm for remote resources provision along with the Services’ Virtualization concept for assets’ elasticity and scalability arise as the driving forces in AAL deployment, due to its omnipresence and resource-rich characteristics.

In this thesis, we propose such a virtualized cloud-based AAL system that enables health-cares and first-responders to constantly monitor a patient’s indoor/outdoor position and be alarmed in case of emergency. Positioning is made available through an embedded device (carried by the patient) that performs the proximity and cell identification techniques (Wi-Fi and GSM), which in turn sends the geolocation data to a Cloud-based application for further processing. In case that the patient drifts away from his/her premises, the application retrieves the closest-to-the-patient third party agents (hospital, police, volunteers, first-responders, etc.)

through the Location-to-Service-Translation (LoST) protocol, and alerts them about the emergency situation via a SIP-based VoIP communication channel. Finally, a performance evaluation was performed, under controlled-conditions environment and results concerning the system's response time were obtained.

Table of Contents

List of Figures	6
List of Tables	8
Declaration	9
1 Introduction.....	11
2 State of the Art and Contribution.....	13
2.1 Ambient Intelligence.....	13
2.2 Ambient Assisted Living	13
2.3 Behavioral Modeling Applications in Ambient Assisted Living.....	13
2.4 Location Identification in Ambient Assisted Living	14
2.5 Issues in Ambient Assisted Living	15
2.6 Contribution	16
3 Architecture	17
3.1 The Cloud.....	18
3.1.1 Location-to-Service-Translation Service	18
3.1.2 MongoDB Service	18
3.1.3 Profiling Service	18
3.1.4 Positioning Service	19
3.1.5 Service Logic	19
3.2 The Edge	19
3.2.1 Small-Cells - Access Points	20
3.2.2 GSM – GPRS/2G/3G/4G.....	20
3.3 The Extreme Edge.....	20
3.3.1 Embedded Devices.....	20
3.4 Architecture Overview	21
4 Implementation	23
4.1 Overview.....	23
4.2 The Database.....	23



4.3	LoST Server	24
4.4	System Services	25
4.5	User Application	27
4.6	Use Case Scenario.....	28
4.7	Interaction Sequence	35
5	Evaluation	37
5.1	Assumptions.....	37
5.2	Variables	37
5.3	Experiment.....	38
5.3.1	Ten (10) User Requests.....	39
5.3.2	One Hundred (100) User Requests	40
5.3.3	One Thousand (1000) user Requests	41
5.4	Discussion	43
6	Conclusion – Future Work.....	45
	Bibliography	46

List of Figures

Figure 1 High-level overview of the AAL emergency positioning system	17
Figure 2 Embedded device internal architecture	21
Figure 3 Architecture overview	21
Figure 4 Cellular JSON data created by the embedded device.....	22
Figure 5 mLab configuration dashboard.....	23
Figure 6 LoST Server Javascript interface.....	24
Figure 7 Heraklion PSAP SQL contents.....	25
Figure 8 Profiling service web interface – User profile of user classified as unsafe.....	26
Figure 9 Positioning service web interface – Basic settings and logs	26
Figure 10 Service logic web interface – Basic settings and logs.....	27
Figure 11 Android application.....	28
Figure 12 Phase one, the user is bound within house premises surrounded by	29
Figure 13 Phase two, user has wandered-off – Classified as unsafe	30
Figure 14 Positioning service during phase 2 – Service located user and informed the service logic.....	31
Figure 15 Service logic during phase 2 – Received position and acquired nearest PSAP	32
Figure 16 Full profile banner for PSAP	33
Figure 17 Profiling service response – List of volunteers	34
Figure 18 Profiling service during phase 2 – Received user’s location and acquired nearby volunteers	34
Figure 19 Limited profile banner for volunteers.....	35
Figure 20 Sequence diagram.....	36
Figure 21 Sequence Diagram of the task execution sequence.....	38
Figure 22 10 user requests - Fog Implementation	39
Figure 23 10 requests – Cloud Implementation.....	39
Figure 24 100 user requests - Fog Implementation	40
Figure 25 100 user requests - Cloud Implementation.....	41

Figure 26 1000 user requests - Fog Implementation 42

Figure 27 1000 user requests - Cloud Implementation..... 43



List of Tables

Table 1 10 user requests – Fog Implementation	39
Table 2 10 user requests – Cloud Implementation.....	40
Table 3 100 user requests - Fog Implementation.....	40
Table 4 100 user requests - Cloud Implementation	41
Table 5 1000 user requests - Fog Implementation.....	42
Table 6 1000 user requests - Cloud Implementation	43

Declaration

Part of the contribution of this thesis has been submitted and accepted for publication:

Peer Review on Scientific Journals:

Y. Nikoloudakis, S. Panagiotakis, and E. Markakis, "A Fog-based Emergency System for Smart Enhanced Living Environments." *IEEE Cloud Computing*

Peer Review on Scientific Books

Y. Nikoloudakis, S. Panagiotakis, E. Markakis, G. Mastorakis, C.X. Mavromoustakis, "Towards a FOG - enabled navigation system with advanced cross - layer management features and IoT equipment," *Cloud Fog Comput. 5G Mob. Networks Emerg. Adv. Appl.*, 2017

Peer Review on Scientific Conferences

Y. Nikoloudakis, S. Panagiotakis, E. Markakis, G. Atsali, and T. Manios, "Cloud composting: A centralized approach," in *2016 International Conference on Telecommunications and Multimedia (TEMU)*, 2016, pp. 1–6.

In memory of my mother.



1 Introduction

Over the last few years, a major concern of our society has been the growth of elder population with physical/mental disabilities and are in the need of constant care and supervision. Specialized health-care institutions are one of the available solutions, but in most cases, prove to be quite expensive and require the persons of interest to move from their familiar environment to secluded facilities. Both these issues have negative impact on the targeted individuals and their family members from a psychological and financial perspective.

Towards alleviating those issues, various approaches have been proposed. Assisted Living (AL)[1], an endeavor for helping people age well at home empowered by state-of-the-art Information and Communication Technologies (ICT), along with the Ambient Intelligence (AmI) paradigm [2], a composition of pervasive technologies –sensors, robotics, algorithms etc., that provide contextual information –healthcare data, behavioral data, location etc., are the most promising approaches so far.

However, the combination of those two approaches, gave birth to a hybrid multidisciplinary scientific/technological domain, known as the Ambient Assisted Living (AAL) [3], [4]. Its aim is to improve citizen's quality of life by creating intuitive context-aware environments, adaptable to each individual's requirements for improving their everyday life and create safer and more secure living environments, especially for individuals with mental or physical disabilities.

AAL systems, gather and process/analyze large amount of data and perform trivial analytics and algorithms, to deduct certain conclusions and perform possibly life-depending decisions/actions. Therefore, such systems require dependable and resource-rich infrastructures for rapid analysis and response. Typical home-based machines, have static/limited resources and sophisticated installations prove to prohibitively expensive.

Thus, Cloud Computing for its abundance of resources and dynamic provisioning, along with the services' Virtualization concept for assets' elasticity and scalability, prove to be an empowering force for the AAL initiative.

The aging-at-home process, poses various trivial issues for the AAL to tackle. One of them, is the constant monitoring and assistance of people suffering from Alzheimer's or dementia [5]. It is a common characteristic of those individuals to wander-off supervision, flee their premises and being unable to return.

In this thesis, we present a Cloud-based emergency positioning system which constantly monitors users' indoor and outdoor position utilizing various positioning techniques such as proximity and cell-identification (Wi-Fi, GSM), dynamically switching between techniques, depending on the user's localization –indoor/outdoor. Positioning, is made possible through an embedded device, carried by the persons of interest, which performs the localization techniques (proximity and cell-identification) and sends collected data to a cloud-based application for further processing. If a user drifts-away from his/her premises, the system retrieves the closest-to-the-user third-party agents, hospital, police and volunteers, through the Location to Service Translation protocol (LoST) [6], and prompts them with alert messages, containing crucial information such as personal and medical information, geographical location and first response instructions, through a SIP-based VoIP channel. All system components were developed as Docker containers –an OS agnostic virtualization method, to enhance the systems scalability, elasticity and adaptability to existing AAL systems.

This thesis is structured as follows. In *Section 2*, we confer on previous related work and present the current State-of-the-Art upon which this project was based. In *Section 3*, we present the systems architecture and analyze each layer separately. In *Section 4*, we analyze the project implementation and present the basic use case scenario which demonstrates the overall functionality of our system. In *Section 5*, we demonstrate the experimental results and elaborate on the systems efficacy. Finally, in *Section 6* we conclude this thesis by presenting our conclusions and confer on possible future work and improvements.

2 State of the Art and Contribution

The caretaking of the elderly population and the promotion of their life quality, has been in the focus of academic and industrial communities[7], and thus numerous applications, frameworks and paradigms have emerged.

2.1 Ambient Intelligence

Ambient Intelligence (AmI) is a novel interdisciplinary paradigm stemming from the ideas of ubiquitous computing and Norman's[8][9] vision on invisible, unobtrusive computing. It fosters the development of constellations of sensors and actuators to acquire behavioral, health and environmental context and promote the deployment of human-centric, omnipresent, personalized, adaptive anticipatory services[10].

2.2 Ambient Assisted Living

The combination of AL and AmI concepts, is called Ambient Assisted Living (AAL)[11][12][13]. Several tools and technologies have helped the AAL vision become a reality. Such technologies are smart homes, assistive robotics, mobile or wearable sensors, and immersive algorithms such as activity recognition, context modeling, anomaly detection, planning, location and identity identification. In the subsections below, we confer on the current state of the art concerning AAL.

2.3 Behavioral Modeling Applications in Ambient Assisted Living

Academic and commercial communities take a significant interest in creating platforms delivering AAL services. The research mainly focuses on indoor positioning, activity and behavior observation, biometric monitoring, danger detection, and alerting.

Gilles Virone and Andrew Sixsmith presented a platform (SOPRANO) that extracts behavioral patterns from users' daily activities[14]. After processing and evaluating the patterns, an intelligent algorithm provides activity prediction, thus proactively alerting authorities of possible danger or health decline of the target user[15]. The Soprano project also employs pervasive technologies (sensors, actuators, smart interfaces, and artificial intelligence) to create a

supportive environment for older people living at home[16]. This user-driven platform tackles issues such as social isolation, security and safety, forgetfulness, mobility, and quality of care, related to socially and activity-challenged individuals confined to their homes.

Diego Lopez and his colleagues presented a holistic, affordable AAL platform that includes an AAL kit and a centralized management service-provisioning system[17]. It employs an AAL store, where users can dynamically install or uninstall AAL services—such as smart TV interfaces, smart home applications, alerts, and vitals monitoring—to their systems.

The Saapho project presents an infrastructure in which target users interact with the system, configuring settings or initiating services using an Android tablet[18]. Environment (gas and temperature) and health (glucose and heartrate) sensors provide context about the user. Cloud middleware gathers the data, detects abnormal behaviors, and predicts possibly dangerous activities. Unfortunately, a human actor must oversee the data and predictions.

2.4 Location Identification in Ambient Assisted Living

In addition to context modelling, the AAL initiative has been enriched with Location aware applications that enable the constant monitoring of the users' location and avoiding wandering situations. In this subsection, we present a number of projects tackling that issue.

Orange Alerts, presents an infrastructure that addresses individuals suffering from dementia or Alzheimer's disease[19]. The system facilitates a set of services that monitor patients, build patient profiles according to their behavioral statistics, track patients' geographical locations using a GPS interface, and stores the data in a server, where relatives, caregivers, and doctors can access.

Jie Wan et al., introduced an outdoor assistance system, employing memory triggers and behavior deviation mechanisms. Users carry a GPS-enabled mobile device that alerts a centralized server of the user's whereabouts. Users are periodically prompted with reminders of planned time-based activities. Caregivers, are alerted with SMS messages or voice calls to be informed of possible behavioral deviation of the user, or involvement in dangerous situations. The server maintains, users' profile, position and planned activities.

Sebastian Fudickar and Bettina Schnor, developed a platform to assist users with cognitive impairment, perform simple everyday tasks and stay within their familiar environment, as well as alerting their appointed caretakers of possible critical situations. The system, prompts users with alerts concerning scheduled tasks, and alerts caretakers in case of a wandering-off situation. The system utilizes a distributed in-house network (access points) to acquire coarse location information and VOIP to establish voice interaction between users and caretakers[20].

2.5 Issues in Ambient Assisted Living

The implementation of such pervasive technologies, raise several ethical, social and technical issues that need to be tackled, to preserve users' privacy, dignity, and maintain system efficacy.

Technology acceptance is one of the many issues, older people face when introduced to assistive technology. The elderly are fairly reluctant to adopt to new technology, forcing them to deviate from their familiar non-technology daily routine, a situation better described by the ancient proverb: *Better a known devil than an unknown god* [21], treating technology as an unnecessary complicated addition[11] to their lives. In that notion, AAL technologies should be seamless unobtrusive, yet maintain a ubiquitous and pervasive manner. Additionally, user participation should be limited to minimum, avoiding the need for training of users that are originally unwilling to adapt to any kind of technology.

AAL applications, are generating and traversing personal, health and social information through wired and wireless networks, exposing this information to possible malicious interceptions. This kind of information is highly sensitive, and should not be exchanged between entities without a certain level of security or trust. Systems should conform to certain security guidelines to make sure that such delicate information, should only be accessed by authorized recipients[22].

Most AAL systems, collect and process large volumes of real-time data, to analyze and rapidly deduct crucial conclusions or perform vital actions. Those requirements, dictate that AAL systems must be implemented in robust infrastructures, incorporate adequate computing resources and perform in a fast and responsive manner. Thus, the Cloud Computing paradigm introduces a suitable environment for AAL applications. Its dynamic resource provisioning

characteristics, enable the deployment of sophisticated algorithms and in-depth analysis of large amounts of data in real-time. Additionally, services' Virtualization concept (Docker Containers) complements AAL systems with its adaptability, elasticity and scalability characteristics.

2.6 Contribution

Taking into consideration all the issues mentioned in subsection 2.5, the aim of this thesis is to design, develop and implement an unobtrusive, adaptive, Cloud-based positioning AAL module, to monitor users' position and alert all suitable entities in case of emergency. In overall, the main contributions of this thesis are as follows:

- Create a Cloud-Based, AAL positioning module for the constant monitoring of patients' position in space.
- Develop an intuitive indoor/outdoor positioning mechanism that utilizes alternative localizations methods (proximity-Wi-Fi and cell-identification-GSM).
- Create an emergency mechanism to retrieve and alert multiple groups of third-party agents, such as caretakers, hospitals, police and volunteers, who are geographically closer to the patient, over the LoST protocol, through a SIP VoIP channel.

3 Architecture

Our system operates in three basic aggregation planes. The cloud[23][24], wherein all modules and services are deployed, the edge[25][26], where in-house Wi-Fi access-points or small-cells and GSM base-stations operate and the extreme edge[27][28] where the embedded devices carried by the target users are found. In this chapter, we will separately present each plane, and briefly elaborate on the partaking actors, and the interactions between them. A basic overview of the system's architecture is presented in Figure 1.

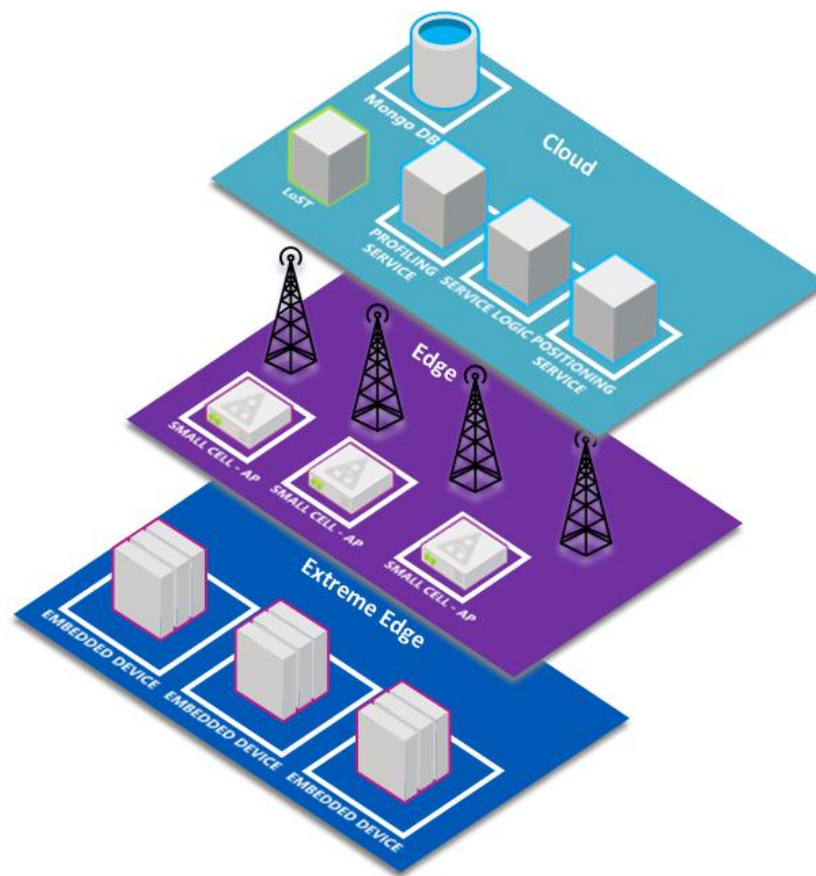


Figure 1 High-level overview of the AAL emergency positioning system

3.1 The Cloud

To ensure our system's robustness, elasticity and ubiquitousness, all system's services were deployed in the cloud. In the sections below, we present each service and elaborate on its functionality.

3.1.1 Location-to-Service-Translation Service

The LoST protocol is an emergency mechanism which locates the geographically nearest Public Safety Answering Point URL, depending on the target user's location. For experimental and demonstration purposes, the LoST service, conforming to the LoST RFC[6], was deployed in the Pasiphae-Lab cloud server to accommodate the LoST protocol functionality. The service operates by receiving a XML request, containing the user's coordinates (latitude – Longitude), and returns the URL of the nearest PSAP. In our use-case, the Service-Logic service receives the user's position from the Positioning service and probes the LoST service, to acquire the URL of the nearest PSAP.

3.1.2 MongoDB Service

It is mandatory that the system maintains a persistent database containing information of users' personal and medical data. In that notion, MongoDB, a scalable, non-SQL, object oriented database was used. To achieve maximum data redundancy, accessibility and ubiquitousness, the database was deployed in Amazon's AWS Database-as-a-Service cloud server.

3.1.3 Profiling Service

The Profiling service, accommodates some of the most crucial functionalities of our system. The system implements a user profiling mechanism that distinguishes users in three categories. The system administrators, the target users and the volunteers. The system administrators can insert, edit and delete users' information and system settings. The target users, are the persons of interest in the presented use case. The system maintains target users' detailed personal and medical information which is stored at the cloud database. Volunteers, are trained users, who can respond to a possible distress call. Lastly, the service classifies target users as *safe* or *unsafe* depending on their current safety status.

3.1.4 Positioning Service

The Positioning service, periodically calculates the users' position by receiving certain information from the embedded device carried by the target users. More specifically, the embedded device measures and sends the Received Signal Strength (RSSI) of a preconfigured in-house Wi-Fi access point or small-cell, to inform the positioning service that the user is bounded within the house premises. Consequently, the user is flagged as *IN*.

If the target user wanders-off, the embedded device loses connectivity with the in-house access-point. Therefore, it collects all surrounding serving base-station information (cell-ID, Mobile Network Code, Location Area Code etc.) and sends it back to the positioning service. The service acquires the users' outdoor position (latitude – longitude), leveraging the received base-station information employing a Web Geolocation API. The user is then flagged as *OUT*. The Service-Logic is informed each time the positioning service calculates a user's position.

3.1.5 Service Logic

The Service-Logic service, is the system's main mediator. It receives the user's position from the Positioning service, classifies them as *safe* or *unsafe* depending on their location, and informs the Profiling service with the users' location and present situation, which, by its turn, updates the user profiles in the database. Consequently, if a user is classified as *unsafe*, the Service-Logic service probes the LoST service and alerts the acquired PSAP with the user's location, personal and medical information. Additionally, the service probes the Profiling service and receives a list of the nearby volunteers and alerts them, providing them with the user's position and personal information along with a set of first response instructions.

3.2 The Edge

At the edge, in-house Wi-Fi access points or small-cells, and cellular base-stations provide the system with the appropriate information for it to acquire users' geographical position. The communication between the embedded device and the system is realized by utilizing a mobile data connection (GPRS/2G/3G/4G)

3.2.1 Small-Cells - Access Points

In-house small-cells or access points, are commodity devices with a wireless interface that provide connectivity to the user's embedded device. The embedded device, periodically measures the RSSI and sends it back to the positioning service. Although those devices are not special in any way, nor require any special configuration, help the system decide whether the user is bounded within the house premises or not.

3.2.2 GSM – GPRS/2G/3G/4G

The cellular network, provides the embedded device with the information (cell-ID, Mobile Network Code, Location Area Code etc.) needed by the positioning system. Consequently, the embedded device will use the cellular data connection (GPRS/2G/3G/4G/5G) to upload the collected data back to the cloud, for the positioning system to acquire the user's location (latitude – longitude).

3.3 The Extreme Edge

The extreme edge is the real-world plane where the user embedded devices operate.

3.3.1 Embedded Devices

Each user, carries an embedded device, integrating various interfaces providing the system with a level of context awareness and geographical information. A Wi-Fi interface connects to the in-house Wi-Fi access-point or small-cell. The device periodically collects and sends the measured RSSI to the positioning service that determines whether the user is inside or outside the access-point or small-cell radius surrounding the user's premises. Once the user is found outside that radius, a GSM interface connects to the outdoor cellular network. The device collects information about the serving base-stations and sends it back to the positioning service over a data connection (GPRS/2G/3G/4G), for it to determine the user's outdoor geographical location. The embedded device internal architecture is depicted in Figure 2.

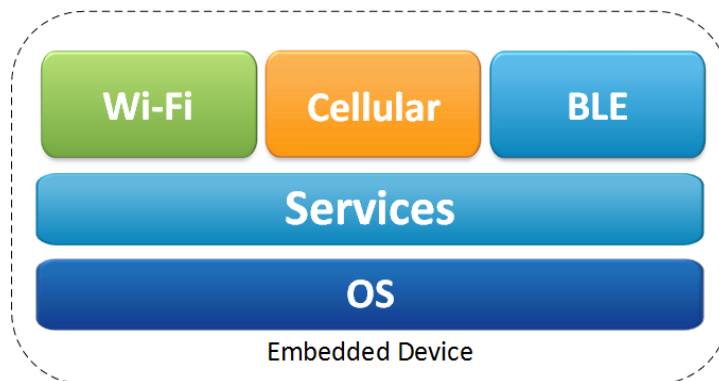


Figure 2 Embedded device internal architecture

3.4 Architecture Overview

As explained above, the system is divided in three basic aggregation layers. The Cloud, the Edge and the Extreme Edge. In Figure 3, one can observe the way all the system’s components interact and communicate.

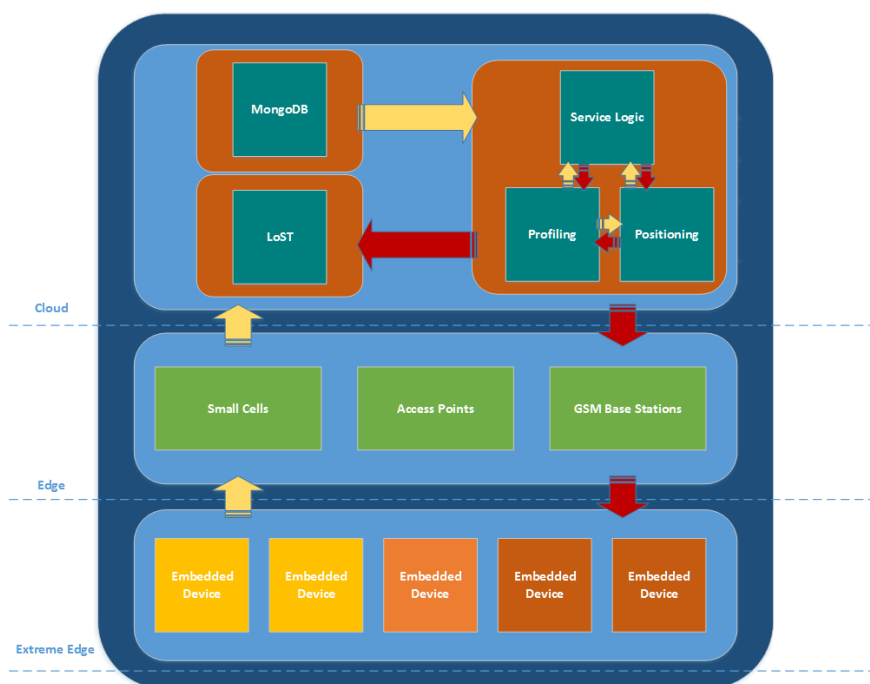
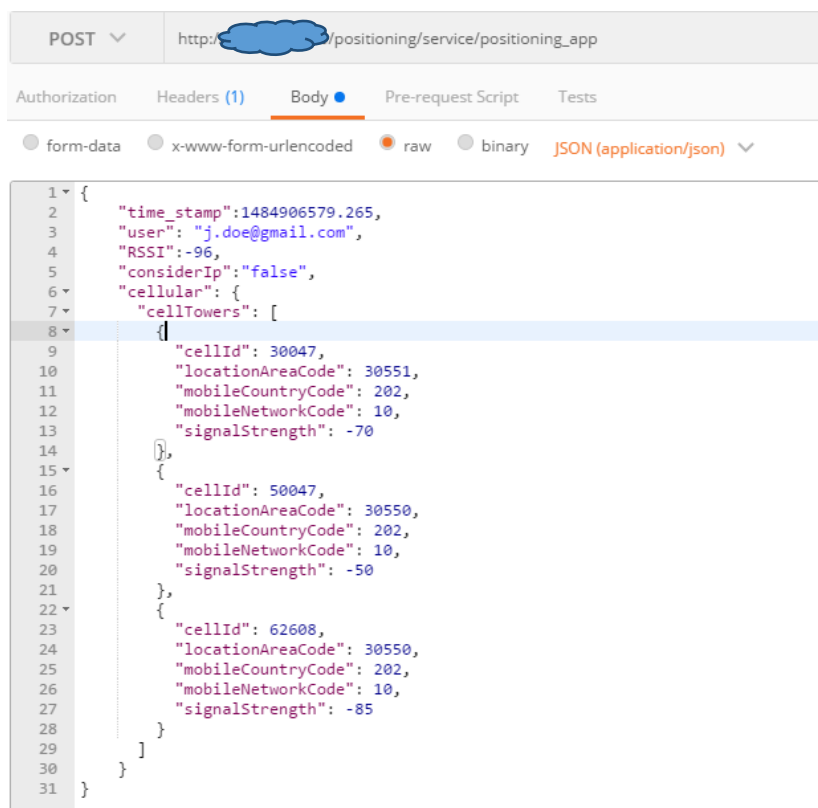


Figure 3 Architecture overview

In the lower layer, the Extreme Edge, we find the embedded devices carried by each user. Each device, collects the appropriate data from its environment (Wi-Fi RSSI, cellular data –CID, MNC, LAC etc.), builds a JSON HTTP POST request (Figure 4) containing the collected information and sends it to the cloud, where the services analyze and utilize that information accordingly.



```

1 {
2   "time_stamp":1484906579.265,
3   "user": "j.doe@gmail.com",
4   "RSSI": -96,
5   "considerIp": "false",
6   "cellular": {
7     "cellTowers": [
8       {
9         "cellId": 30047,
10        "locationAreaCode": 30551,
11        "mobileCountryCode": 202,
12        "mobileNetworkCode": 10,
13        "signalStrength": -70
14      },
15      {
16        "cellId": 50047,
17        "locationAreaCode": 30550,
18        "mobileCountryCode": 202,
19        "mobileNetworkCode": 10,
20        "signalStrength": -50
21      },
22      {
23        "cellId": 62608,
24        "locationAreaCode": 30550,
25        "mobileCountryCode": 202,
26        "mobileNetworkCode": 10,
27        "signalStrength": -85
28      }
29    ]
30  }
31 }

```

Figure 4 Cellular JSON data created by the embedded device

In the Edge, we find the Wi-Fi access points or small cells and the GSM Base Stations. The above act as communication mediums between the embedded devices and the cloud services, but also enable our system calculate the users' coarse location in real time.

Lastly, in the Cloud, the combined interaction of our three services –profiling, positioning, service-logic– along with the LoST service and the system's Database, provide the overall functionality of our system.

4 Implementation

4.1 Overview

For the implementation and evaluation of the presented project, all modules were deployed in separate cloud servers. Our goal was to present our system's interoperability and elasticity, and lastly evaluate the system's overall responsiveness, given the diverse localization of each module. In this subsection, we will present each module separately and elaborate on its configurations.

4.2 The Database

The database module was deployed on the Amazon mLab Database-as-a-Service server (DaaS). The mLab, is a cloud DaaS service providing free MongoDB database deployments. Users can seamlessly create and manage MongoDB databases for their applications. In Figure 5, the database dashboard is depicted, listing all the data tables of our system, along with some basic functions the user can perform on the database.

The screenshot shows the mLab dashboard for a MongoDB database named 'apele_thesis'. The interface includes a navigation bar with 'WELCOME', 'PLANS + PRICING', 'PLAN COMPARISON', 'DOCS + SUPPORT', 'ACCOUNT', and 'LOG OUT'. The user is logged in as 'g.nikoloudakis'. The dashboard displays connection instructions for the mongo shell and a standard MongoDB URI. A warning message states: 'Sandbox databases do not have redundancy and therefore are not suitable for production. Visit our guide to running in production for more info.' Below this, there are tabs for 'Collections', 'Users', 'Stats', 'Backups', and 'Tools'. The 'Collections' tab is active, showing a table of collections with columns for NAME, DOCUMENTS, CAPPED?, and SIZE. The table lists six collections: alert (2 documents, 8.08 KB), positioning_settings (1 document, 8.22 KB), role (2 documents, 16.06 KB), service_logic_settings (1 document, 8.22 KB), settings (1 document, 8.09 KB), and user (15 documents, 40.72 KB). Each row has a delete icon (X) on the right.

NAME	DOCUMENTS	CAPPED?	SIZE
alert	2	false	8.08 KB
positioning_settings	1	false	8.22 KB
role	2	false	16.06 KB
service_logic_settings	1	false	8.22 KB
settings	1	false	8.09 KB
user	15	false	40.72 KB

Figure 5 mLab configuration dashboard

4.3 LoST Server

The LoST server, as described above, is the emergency service which retrieves the geographically nearest PSAP to be alerted of the distress situation. This service is to be accommodated by liable authorities such as the police or the health-emergency dispatching service. For evaluation purposes, the LoST service was deployed as a Linux virtual machine, on a Pasiphae Lab cloud server, wherein a JavaScript-based service accommodates the LoST functionality, employing a RESTful API for user interaction.

Figure 6 depicts the LoST Server Graphical User Interface (GUI) as it appears at <http://lost.owncloud.gr/lost>. The service accepts RESTful API commands at <http://lost.owncloud.gr/lost/lost>. The service uses the PostgreSQL database, and utilizes geodetic SQL queries to retrieve the geographically nearest PSAP, depending on the user's location. In our use-case scenario, five (5) PSAPs have been configured in the database. Each PSAP, contains an emergency number for regular landline calls, a sip URL for SIP calls and a series of coordinates surrounding the specific area of coverage. The Heraklion PSAP contents are presented in Figure 7.

LoST AJAX CLIENT

Instruction: You can either enter a civic address and press the search button, enter geodetic coordinates and press the search button, or just click on the map to generate a LoST query. If the value of radius in the form is 0, the request will be a point. Otherwise, the request is a circle.

Server Address:	<input type="text" value="83.212.113.82"/>	e.g., example.com or 127.0.0.1
Server Port:	<input type="text" value="8080"/>	e.g., 8080
Server Path:	<input type="text" value="/lost/lost"/>	e.g., /lost/lost
Address:	<input type="text"/>	e.g., 1214 Amsterdam Ave New York
or		
Coordinates:	<input type="text"/>	e.g., 40:48:33N 73:57:34W or 40.8091 -73.9593
Radius:	<input type="text"/>	Optional, in meter.
<input type="button" value="search"/> <input type="button" value="reset"/>		
Message:		

Figure 6 LoST Server JavaScript interface


```

1  INSERT INTO geo_us (source, source_id, service, sn, display_name, uri, last_updated, the_geom)
2  VALUES ('Iraklion PSAP',
3  'Nikoloudakis test ID',
4  'urn:service:sos',
5  '112',
6  'Iraklion Testing PSAP',
7  'sip:psap_iraklion@nikoloudakis.iraklion.com',
8  now(),
9  ST_SetSRID(ST_MakePolygon(ST_GeomFromText('LINESTRING(35.45164535490567 24.71240344672975,
10 34.86129370550552 24.65763649442331,
11 34.8485762437594 25.402489033982,
12 35.51462383535765 25.42554216609895,
13 35.45164535490567 24.71240344672975)')), 4326));
14

```

Figure 7 Heraklion PSAP SQL contents

4.4 System Services

The system services were developed in python, using the python-Flask micro-framework. Each service has a separate GUI. The profiling service, hosts the most complex user interface among the services, as this is the interface used by all the users to create and manage their profiles (Figure 8). The positioning service and service-logic, host rather simple interfaces (Figure 9, Figure 10). All services were deployed on the GRNET-Okeanos Cloud server as Docker containers. Because of the critical nature of our system, it is imperative that the robustness of each service is guaranteed. In that notion, all tasks were designed and deployed in a threaded parallel manner, allowing the system to take full advantage of the underlying hardware (multiple CPU cores, memory etc.). Additionally, a dedicated task scheduler was employed in each service (Service Logic, Profiling Service, Positioning Service) to tackle the management of tasks, guaranteeing each task's successful execution.

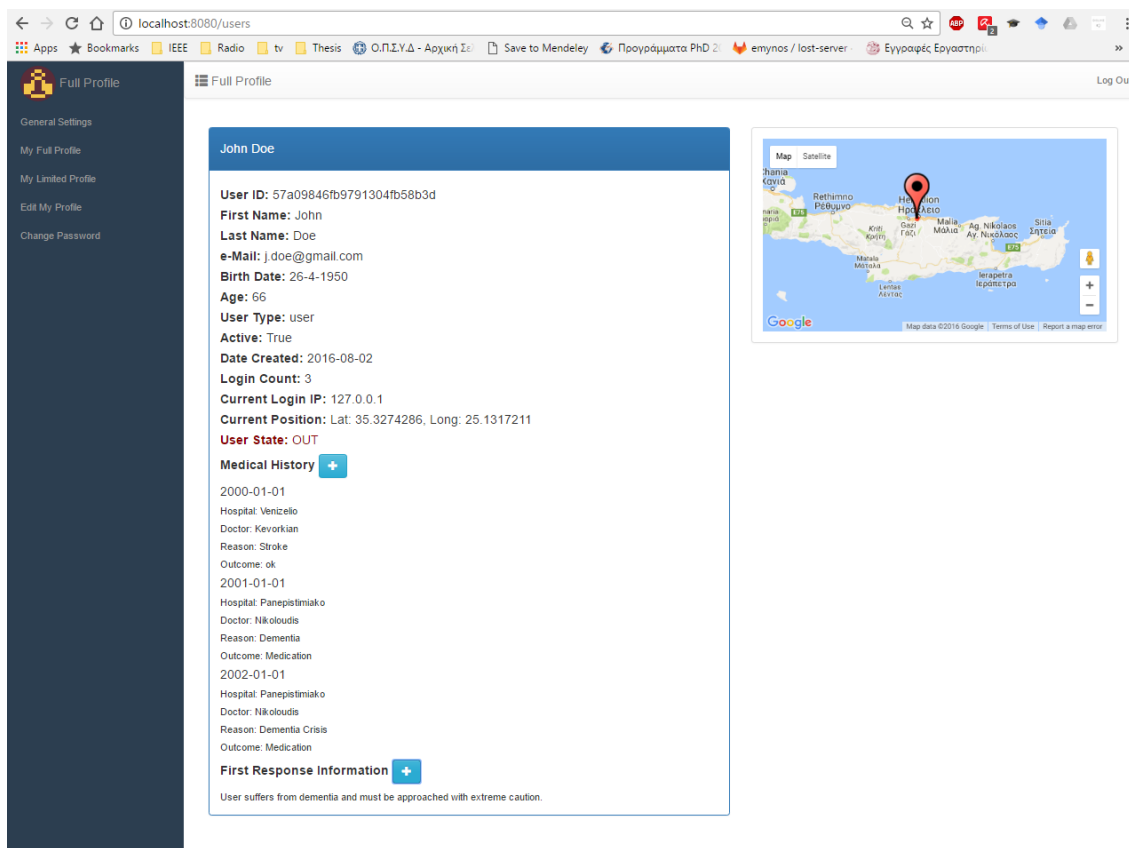


Figure 8 Profiling service web interface – User profile of user classified as unsafe

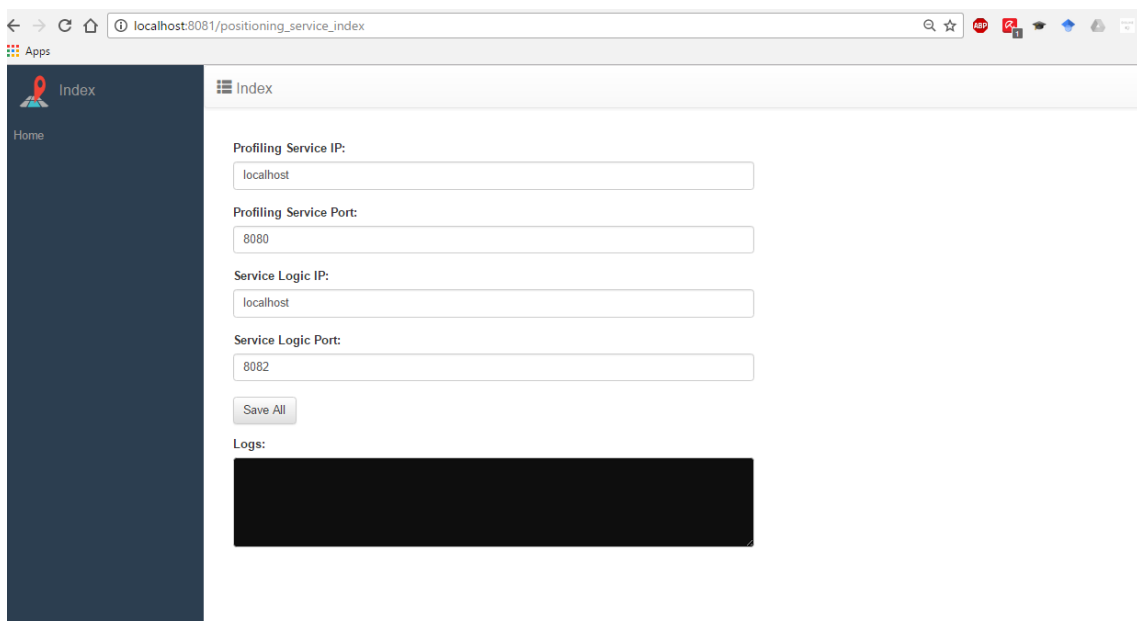


Figure 9 Positioning service web interface – Basic settings and logs

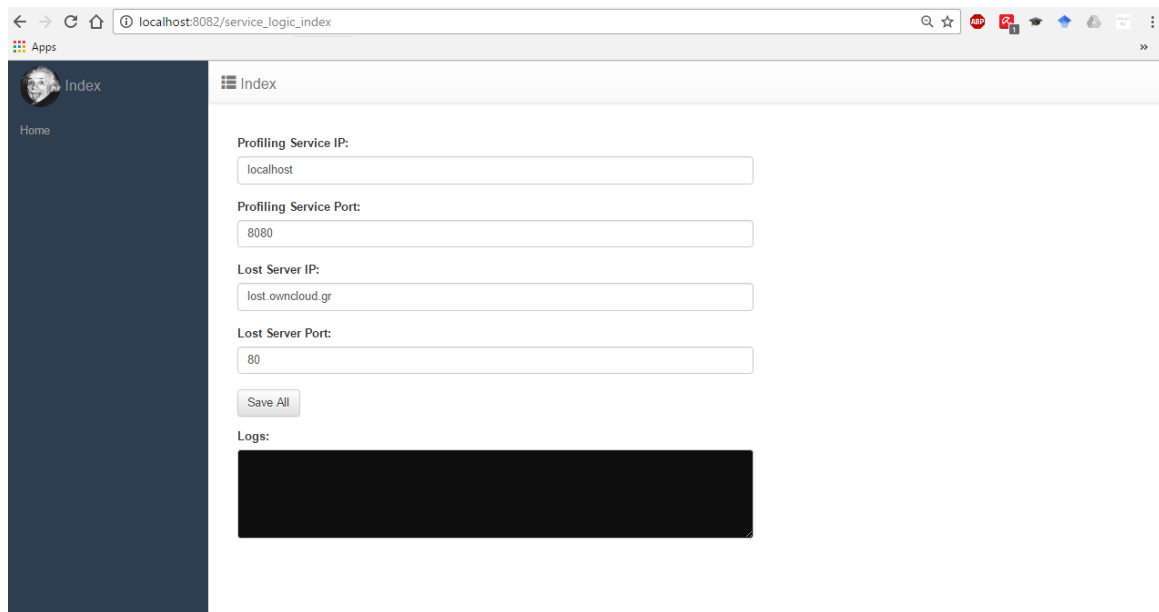


Figure 10 Service logic web interface – Basic settings and logs

4.5 User Application

For the evaluation of our use case scenario, a generic cellular Android phone was used as the embedded device, as it incorporates all the interfaces needed for our project. A basic application was developed using the Android SDK (Figure 11), to accommodate the basic functionalities of the embedded device as described earlier. The application, collects all the available information of the surrounding base-stations and sends it back to the positioning service. For debugging purposes, a logging pane lists all GSM activity.

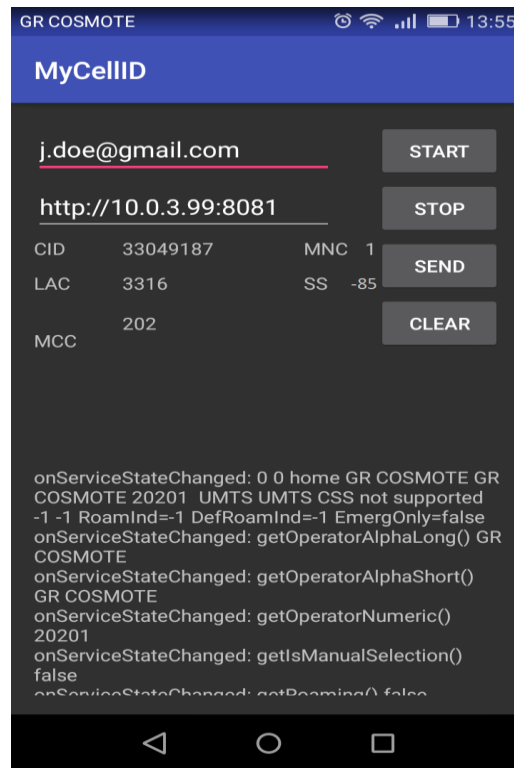


Figure 11 Android application

4.6 Use Case Scenario

The presented use case scenario is divided into two phases. The first phase, deals with the person of interest located within the household boundaries and classified as “safe”. An embedded device, carried by the person of interest, connected to the indoor Wi-Fi access-point or small-cell, continuously measures the RSSI and sends it back to the cloud positioning service. Subsequently, the positioning service receives the RSSI and ensures that the user is bounded within the household boundaries. Figure 12 briefly depicts the aspects of the first phase. Once the user leaves the household premises, thus exiting the Wi-Fi access-point or small-cell radius, the service will stop receiving RSSIs from the predefined device. After a certain period of time, the positioning service will notify the service logic, which will classify the user as “unsafe”.

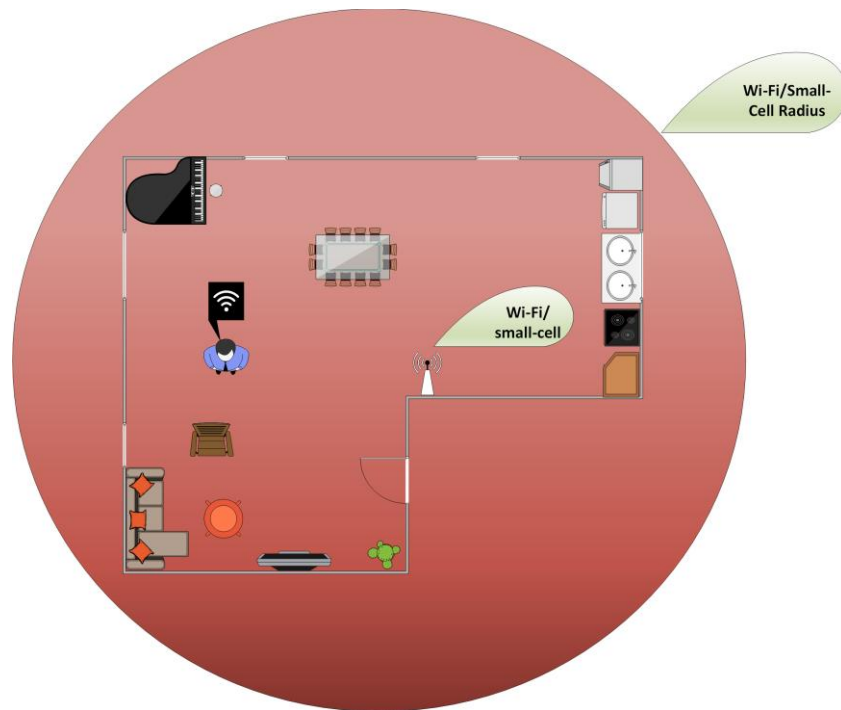


Figure 12 Phase one, the user is bound within house premises surrounded by Wi-Fi radius

The second phase, which is depicted in Figure 13, deals with the user stepping out of the predefined radius, thus being classified as “unsafe”.

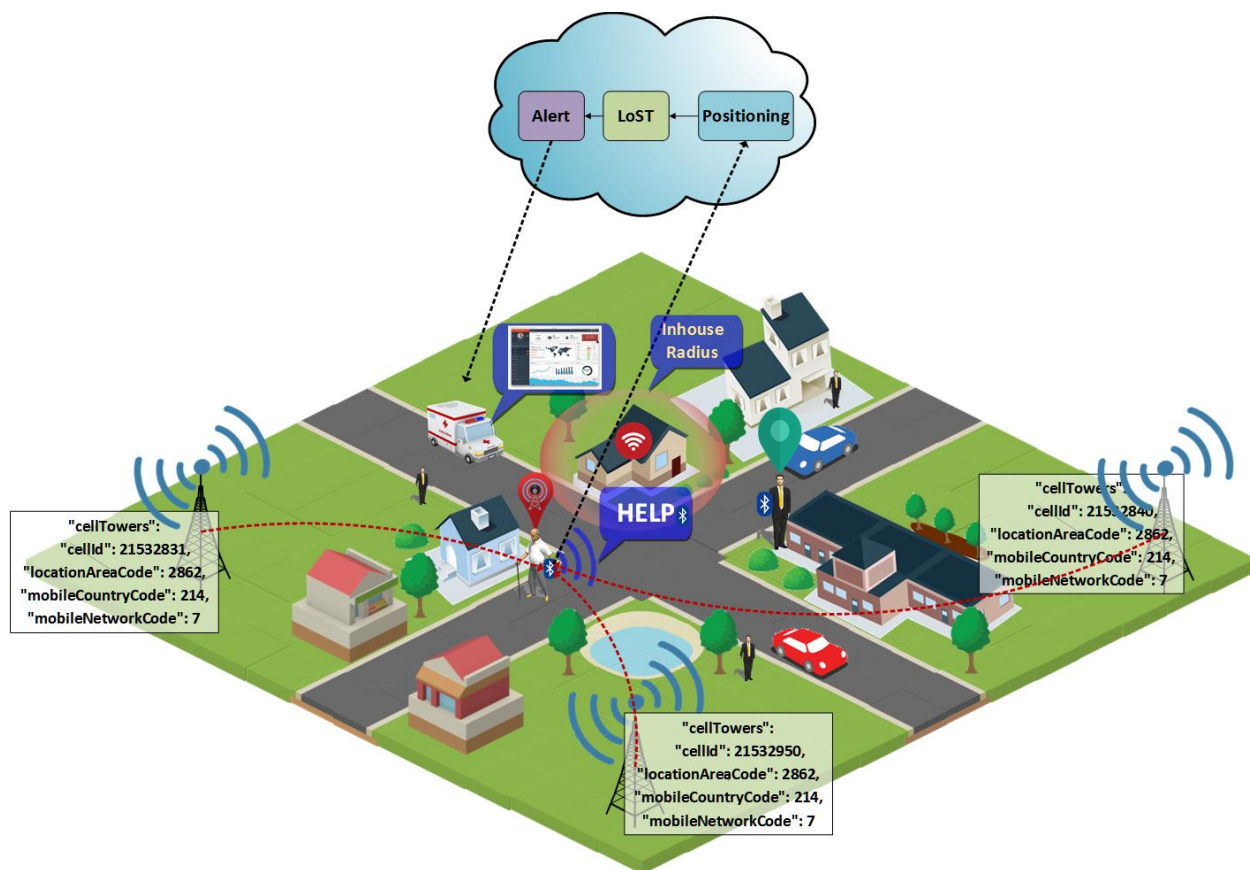


Figure 13 Phase two, user has wandered-off – Classified as *unsafe*

Once found in an outdoor environment, the embedded device will connect to a cellular network and start collecting information concerning the adjacent serving base stations, utilizing a data connection to send the information back to the Positioning service. This task is repeated periodically. The Positioning service, creates a JSON[29] list of the acquired cellular information (CID, MNC, LAC etc.) and probes a Geolocation API (Figure 14) to get the user's position. Consequently, it triggers the Service-Logic module which will locate and inform the authorities responsible for the particular user, by invoking a LoST service(Figure 15), providing them with the user's full profile and geographical location (Figure 16).

The screenshot displays the PyCharm IDE interface. The main editor shows the source code for `positioning_service.py`. The code includes imports for `Flask`, `requests`, and `MongoEngine`, along with configuration for a MongoDB database and a Google Maps API key. The `Run` console at the bottom shows the service starting on `http://0.0.0.0:8081/` and receiving three POST requests. The output of the first request is: `(-96, 'User is outside the House')`. The output of the second request is: `('GOT POSITION FROM GOOGLE', {u'location': {u'lat': 35.3274286, u'lng': 25.1317211}, u'accuracy': 2262.0})`. The output of the third request is: `('INFORMED PROFILING SERVICE', u'localhost', u'8080')`.

```

1  # -*- coding: utf-8 -*-
2
3  from flask.ext.hook import ExtDeprecationWarning
4  import warnings
5  warnings.simplefilter("ignore", category=ExtDeprecationWarning)
6  from flask import Flask, request, json, render_template, redirect
7  import requests
8  from flask_mongoengine import MongoEngine
9  import os
10
11 app = Flask(__name__)
12 app.config['DEBUG'] = True
13 app.config['SECRET_KEY'] = os.urandom(16)
14
15 # MongoDB config
16 app.config['MONGODB_DB'] = 'a'
17 app.config['MONGODB_HOST'] = 'mlab.com'
18 app.config['MONGODB_PORT'] = 11495
19 app.config['MONGODB_USERNAME'] = ''
20 app.config['MONGODB_PASSWORD'] = ''
21
22 # Create database connection object
23 db = MongoEngine(app)
24
25 g_url = 'https://www.googleapis.com/geolocation/v1/geolocate?key=AIzaSyAAyeOvQsvoNanKXf2MS8PmiHKDK-xOdVg'
26 times = 0
27 state = 'IN'
28 LogFile = []
29

```

```

Run positioning_service
C:\Python27\python.exe C:/Users/yannis/Desktop/Pycharm_projects/Thesis/ThesisAPELE/positioning_service/position
* Restarting with stat
* Debugger is active!
* Debugger pin code: 321-722-976
* Running on http://0.0.0.0:8081/ (Press CTRL+C to quit)
10.0.3.95 -- [30/Nov/2016 14:40:11] "POST /service/positioning HTTP/1.1" 200 -
1
10.0.3.95 -- [30/Nov/2016 14:40:12] "POST /service/positioning HTTP/1.1" 200 -
2
10.0.3.95 -- [30/Nov/2016 14:40:14] "POST /service/positioning HTTP/1.1" 200 -
3
(-96, 'User is outside the House')
('GOT POSITION FROM GOOGLE', {u'location': {u'lat': 35.3274286, u'lng': 25.1317211}, u'accuracy': 2262.0})
('INFORMED PROFILING SERVICE', u'localhost', u'8080')
10.0.3.95 -- [30/Nov/2016 14:40:24] "POST /service/positioning HTTP/1.1" 200 -
INFORMED SERVICE LOGIC

```

Figure 14 Positioning service during phase 2 – Service located user and informed the service logic

The screenshot displays the PyCharm IDE interface. The main editor shows the `serviceLogic.py` file with the following code:

```

1  # -*- coding: utf-8 -*-
2
3  import ...
4
5
6
7
8  warnings.simplefilter("ignore", category=ExtDeprecationWarning)
9  from flask import Flask, request, json, render_template, redirect
10 import requests, xml.etree.ElementTree as ET
11 from flask_mongoengine import MongoEngine
12 import os
13
14 app = Flask(__name__)
15 app.config['DEBUG'] = True
16 app.config['SECRET_KEY'] = os.urandom(16)
17
18
19 # MongoDB config
20 app.config['MONGODB_DB'] = '...'
21 app.config['MONGODB_HOST'] = '...mlab.com'
22 app.config['MONGODB_PORT'] = 11495
23 app.config['MONGODB_USERNAME'] = '...'
24 app.config['MONGODB_PASSWORD'] = '...'
25
26 # Create database connection object
27 db = MongoEngine(app)
28
29
30 LogFile = []
31
32
33 class service_logic_settings(db.Document):

```

The Run console shows the following output:

```

Run service_logic
C:\Python27\python.exe C:/Users/yannis/Desktop/Pycharm_projects/Thesis/ThesisAPELE/service_logic/serv...
* Restarting with stat
* Debugger is active!
* Debugger pin code: 321-722-976
* Running on http://0.0.0.0:8082/ (Press CTRL+C to quit)
127.0.0.1 -- [30/Nov/2016 14:40:24] "POST /service/service_logic HTTP/1.1" 200 -
posted to lost server


('PSAP: ', 'sip:psap_iraklion@nikoloudakis.iraklion.com')
('Emergency Number:', '112', '\n')
Volunteers:

('Last Name: ', u'Markakis')
('Last Name: ', u'Sideris')

```

Figure 15 Service logic during phase 2 – Received position and acquired nearest PSAP

Alert Log Out



First Response Information +

User suffers from dementia and must be approached with extreme caution.

Medical History +

2000-01-01
 Hospital: Venizelio
 Doctor: Kevorkian
 Reason: Stroke
 Outcome: ok

2001-01-01
 Hospital: Panepistimiako
 Doctor: Nikoloudis
 Reason: Dementia
 Outcome: Medication

2002-01-01
 Hospital: Panepistimiako
 Doctor: Nikoloudis
 Reason: Dementia Crisis
 Outcome: Medication

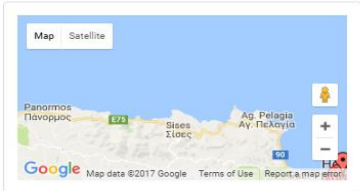
First Name: John
Last Name: Doe
Age: 66
Current Position:
 Lat: 35.3274987,
 Long: 25.1317674
Accuracy: 2266.0m
Volunteer Radius: 5km

Nearby Volunteers +

Esteban Pena 35.348033 24.481443

Anna Hines 35.349562 24.502697

Jorge Sherman 35.338089 24.448



Emergency Number: 112

Figure 16 Full profile banner for PSAP

In addition, the Service-Logic module will acquire a list of the nearest volunteering users from the Profiling service (Figure 17, Figure 18), and inform them of the current situation by providing them with a brief user's profile, a set of first response instructions and the user's geographical location (Figure 19). The user, is finally classified as "safe", either by the authority in charge of the situation, or the system administrator.

```

{
  "first_name" : "John",
  "last_name" : "Doe",
  "email" : "j.doe@gmail.com",
  "age" : 76,
  "photo" : "http://85.223.98.99/users/images/john_doe.png",
  "current_position" : {
    "lat" : 35.353233,
    "long" : 24.482689
  },
  "volunteers" : [
    "Esteban Pena",
    "Jorge Sherman",
    "Anna Hines"
  ],
  "volunteer_radius" : 5000,
  "first_response_info" : "User suffers from dementia
    and must be approached with extreme caution"
}

```

Figure 17 Profiling service response – List of volunteers

The screenshot shows the PyCharm IDE with the following components:

- Code Editor:** Contains the Python code for `profilingService.py`. Key lines include:


```

from flask.ext.http import ExtDeprecationWarning
import warnings
warnings.simplefilter("ignore", category=ExtDeprecationWarning)
from flask_mongoengine import MongoEngine
from flask_security import Security, MongoEngineUserDatastore, UserMixin, RoleMixin, login_required, roles_required
from flask_gravatar import Gravatar
from werkzeug.utils import secure_filename
from flask_mail import Mail
from .models import User, Role, db
from datetime import datetime, timedelta
import os
from pprint import pprint

# Create the APP
app = Flask(__name__)
app.config['DEBUG'] = True
app.config['SECRET_KEY'] = os.urandom(16)
app.config['SECURITY_TRACKABLE'] = True
app.config['SECURITY_CONFIRMABLE'] = False
app.config['SECURITY_REGISTERABLE'] = True
app.config['SECURITY_SEND_REGISTER_EMAIL'] = False
app.config['SECURITY_POST_LOGIN_VIEW'] = '/users/'
app.config['SECURITY_POST_REGISTER_VIEW'] = '/users/welcome'
app.config['SECURITY_RECOVERABLE'] = True
app.config['SECURITY_CHANGEABLE'] = True
app.config['SECURITY_PASSWORDLESS'] = True
app.config['SECURITY_PASSWORD_HASH'] = 'sha512_crypt'

```
- Run Console:** Shows the execution output:



```

C:\Python27\python.exe C:/Users/yannis/Desktop/Pycharm_projects/Thesis/ThesisAPELE/profiling_service/profilingService.py
* Restarting with stat
* Debugger is active!
* Debugger pin code: 321-722-976
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
[{"state": "OUT", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}]
PROFILE RECEIVED USER LOCATION
127.0.0.1 - - [30/Nov/2016 14:40:18] "POST /service/profiling HTTP/1.1" 200 -
127.0.0.1 - - [30/Nov/2016 14:40:21] "POST /service/profiling/get_full_profile HTTP/1.1" 200 -
SENT FULL PROFILE
127.0.0.1 - - [30/Nov/2016 14:40:22] "POST /service/profiling/get_limited_profile HTTP/1.1" 200 -
SENT LIMITED PROFILE
[{"state": "OUT", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}, {"state": "NEARBY", "location": {"lat": 35.3274286, "lng": 25.1317211, "accuracy": 2262.0}, "user-ip": "10.0.3.95", "user": "j.doe@gmail.com"}]
127.0.0.1 - - [30/Nov/2016 14:40:23] "POST /service/profiling/get_nearby_volunteers HTTP/1.1" 200 -

```

Figure 18 Profiling service during phase 2 – Received user's location and acquired nearby volunteers

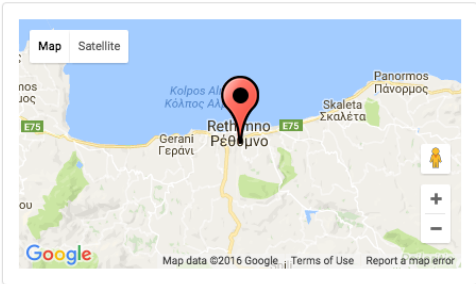
Alert Log Out



First Name: John
Last Name: Doe
Age: 66
Current Position:
 Lat: 35.353233,
 Long: 24.482689
Volunteer Radius: 5km

First Response Information +

User suffers from dementia and must be approached with extreme caution.



Emergency Number: 112

Figure 19 Limited profile banner for volunteers

4.7 Interaction Sequence

Figure 20, illustrates in detail the interaction sequence between all the system services. In overall, we observe that the service-logic acts as the system's mediator. Firstly, the embedded device continuously probes the positioning service to calculate the user's position. The user can be classified as IN or OUT. In both scenarios, the service-logic constantly updates the user's profile status (IN or OUT). In the scenario where a user is classified as OUT, the service-logic receives the user's outdoor position by the positioning service. Consequently, the service-logic acquires the geographically nearest PSAP and user's full profile, by querying the LoST service and profiling service respectively, and sends the PSAP an alert message containing the user's full profile and location. Additionally, the service-logic collects a list of all available nearby volunteers by querying the profiling service and alerts them with an alert message containing the basic profile and location of the user.

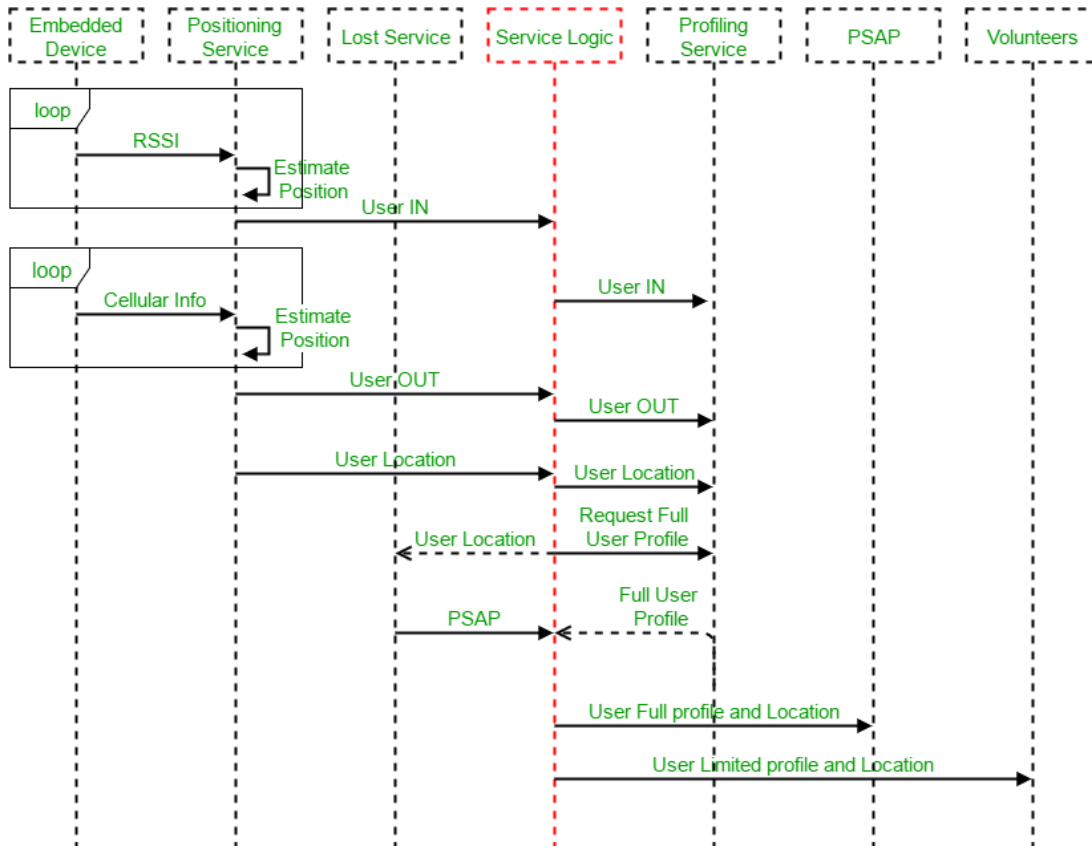


Figure 20 Sequence diagram

5 Evaluation

A performance evaluation was performed, under controlled-conditions environment, to export qualitative conclusions and assess system's efficacy. The experiment was implemented in two different deployment scenarios (Local Server—Pasiphae and Cloud –GRNET-Okeanos). In the first implementation scenario (Local Server), a Wi-Fi communication channel was utilized for the interaction between the embedded device and our system and a 3G GSM-based data channel was utilized for the second implementation scenario (Cloud).

5.1 Assumptions

For the purposes of the evaluation, we assume the experiment takes place during the second phase of our use-case scenario, where a patient has wandered-off, beyond the radius of the in-house Wi-Fi access point. Thus, the embedded device has deduced the patient is “*unsafe*” and started the emergency procedure, wherein it collects all available cellular information (CID, MNC, LAC etc.) of the surrounding serving base-stations and sends it to the positioning service. The moment the embedded device sends the cellular information to the positioning service for the first time was considered as the starting time of the experiment. To assess our system responsiveness, during each implementation scenario, we probed our system with ten (10), one hundred (100) and one thousand (1000) simultaneous user requests respectively.

5.2 Variables

During the experiment, four values were measured to assess our system's responsiveness. Firstly, the time needed for the system to retrieve the nearest PSAP from the LoST server (time for LoST) and the time needed to retrieve the list of nearby volunteers from the profiling service (time for volunteers) were measured. Additionally, system's CPU usage during the LoST request and the CPU usage during the request for volunteers were also measured (CPU usage for LoST and CPU usage for volunteers).

5.3 Experiment

The experiment was performed in two implementation scenarios under controlled-conditions to extract measurements concerning the system's responsiveness (Local server and Cloud).

As described in 3.1.4, the embedded device performs a JSON request to the positioning service, containing the cellular information (CID, MNC, LAC etc.) of its surrounding serving base-stations. A timestamp was added to the request, to be used as a starting point (starting time) for the measurement of the execution time of the tasks.

For the system to acquire the nearest PSAP, three tasks must be executed. Firstly, the positioning service acquires the users geographical position by probing a geolocation API with the received cellular data. Secondly, it sends the location of the user to the Service Logic and finally the service Logic probes the LoST server with the user's location to acquire the nearest PSAP URL.

For the system to acquire all available nearby volunteers, two tasks must be executed. Firstly, the positioning service acquires the users geographical position by probing a geolocation API with the received cellular data. Secondly, the Service logic. The sequence diagram below (Figure 21) depicts the execution sequence of the tasks.

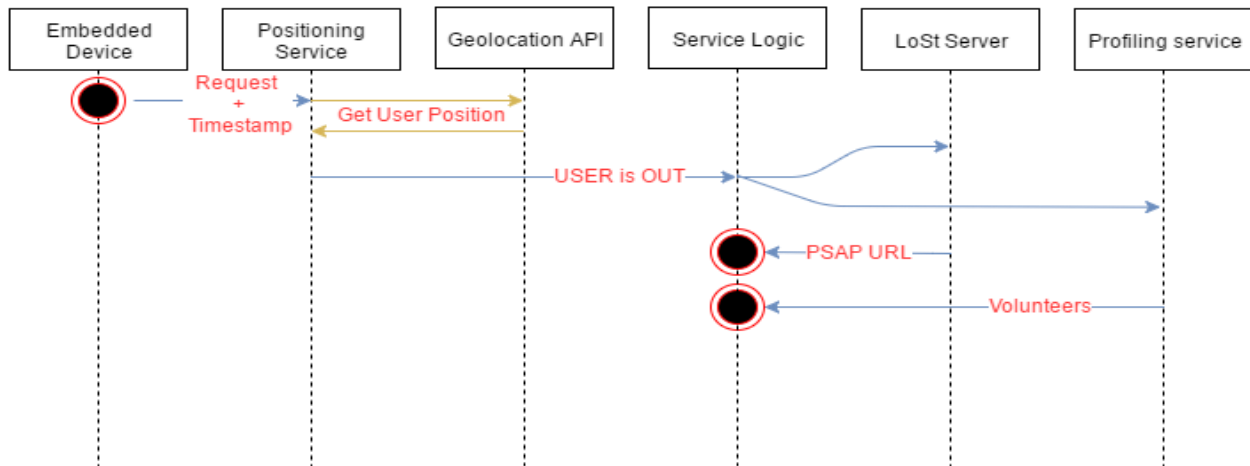


Figure 21 Sequence Diagram of the task execution sequence.

5.3.1 Ten (10) User Requests

During the first set of measurements, the system, was probed with ten (10) simultaneous user requests. The charts below, depict the values measured in both implementation scenarios.

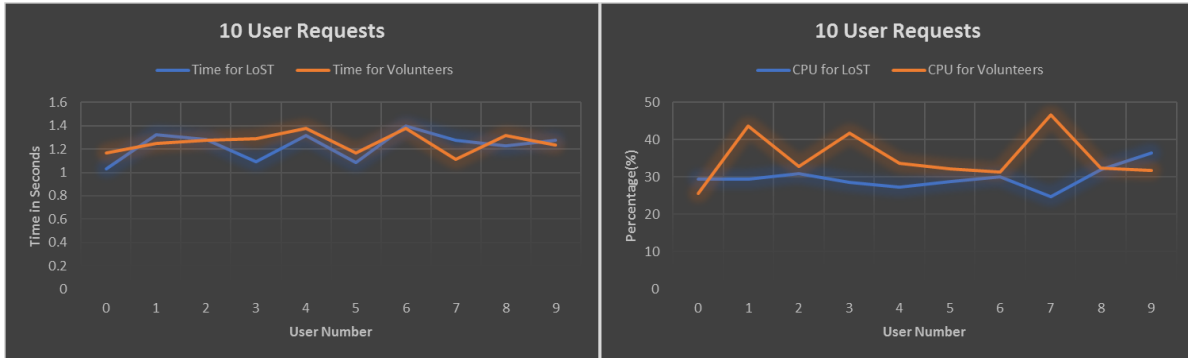


Figure 22 10 user requests - Fog Implementation

Mean Time for LoST	1.22
Mean Time for Volunteers	1.26
Mean CPU for LoST	29.8
Mean CPU for Volunteers	35.24

Table 1 10 user requests – Fog Implementation

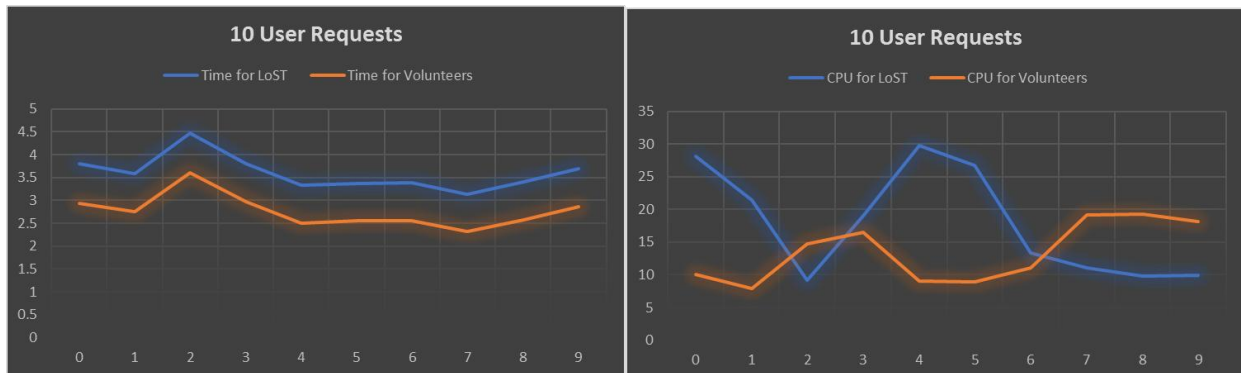


Figure 23 10 requests – Cloud Implementation

Mean Time for LoST	3.6
Mean Time for Volunteers	2.76
Mean CPU for LoST	17.88
Mean CPU for Volunteers	13.49

Table 2 10 user requests – Cloud Implementation

5.3.2 One Hundred (100) User Requests

In the next set of measurements, the system was probed with one hundred (100) user requests. The charts below, depict the values measured in both implementation scenarios.

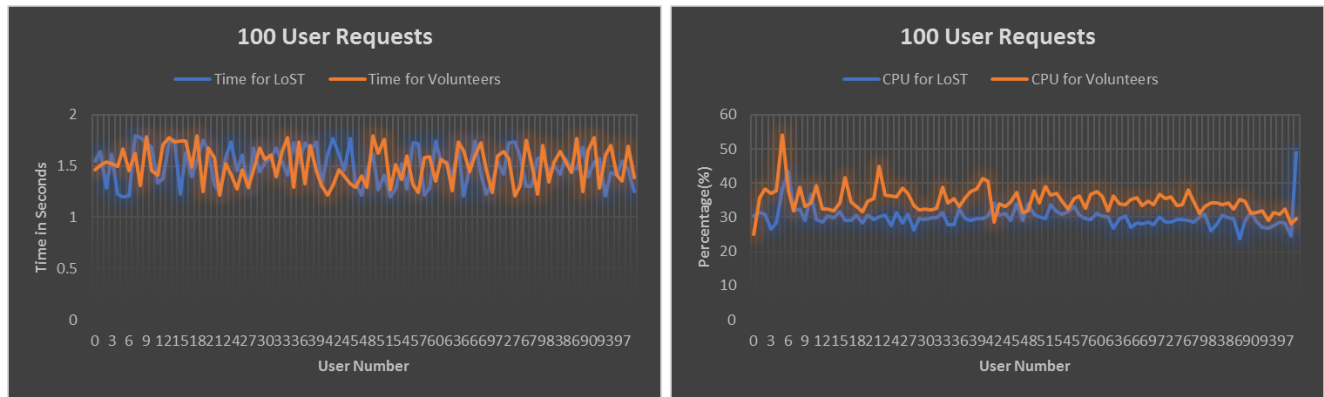


Figure 24 100 user requests - Fog Implementation

Mean Time for LoST	1.50
Mean Time for Volunteers	1.51
Mean CPU for LoST	30.13
Mean CPU for Volunteers	34.85

Table 3 100 user requests - Fog Implementation

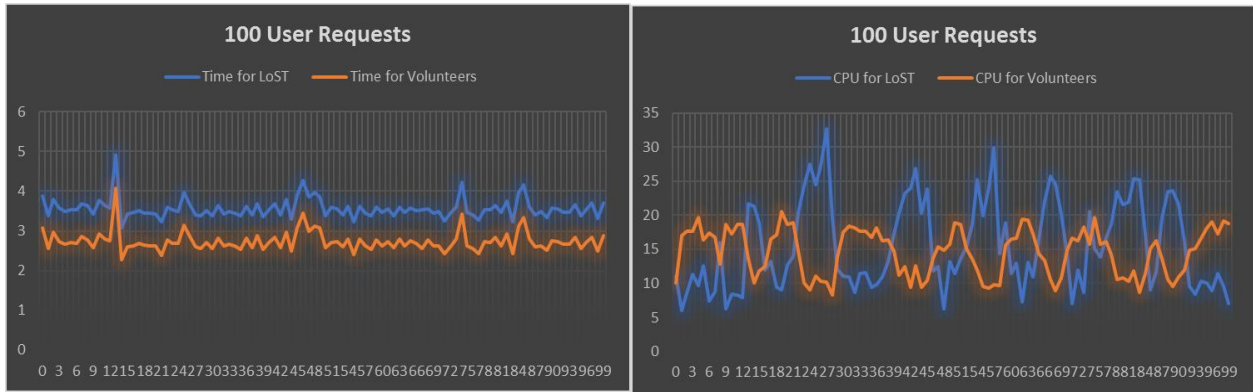


Figure 25 100 user requests - Cloud Implementation

Mean Time for LoST	3.56
Mean Time for Volunteers	2.74
Mean CPU for LoST	15.58
Mean CPU for Volunteers	14.60

Table 4 100 user requests - Cloud Implementation

5.3.3 One Thousand (1000) user Requests

In the last set of measurements, the system was probed with one thousand (1000) user requests. The charts below, depict the values measured in both implementation scenarios.

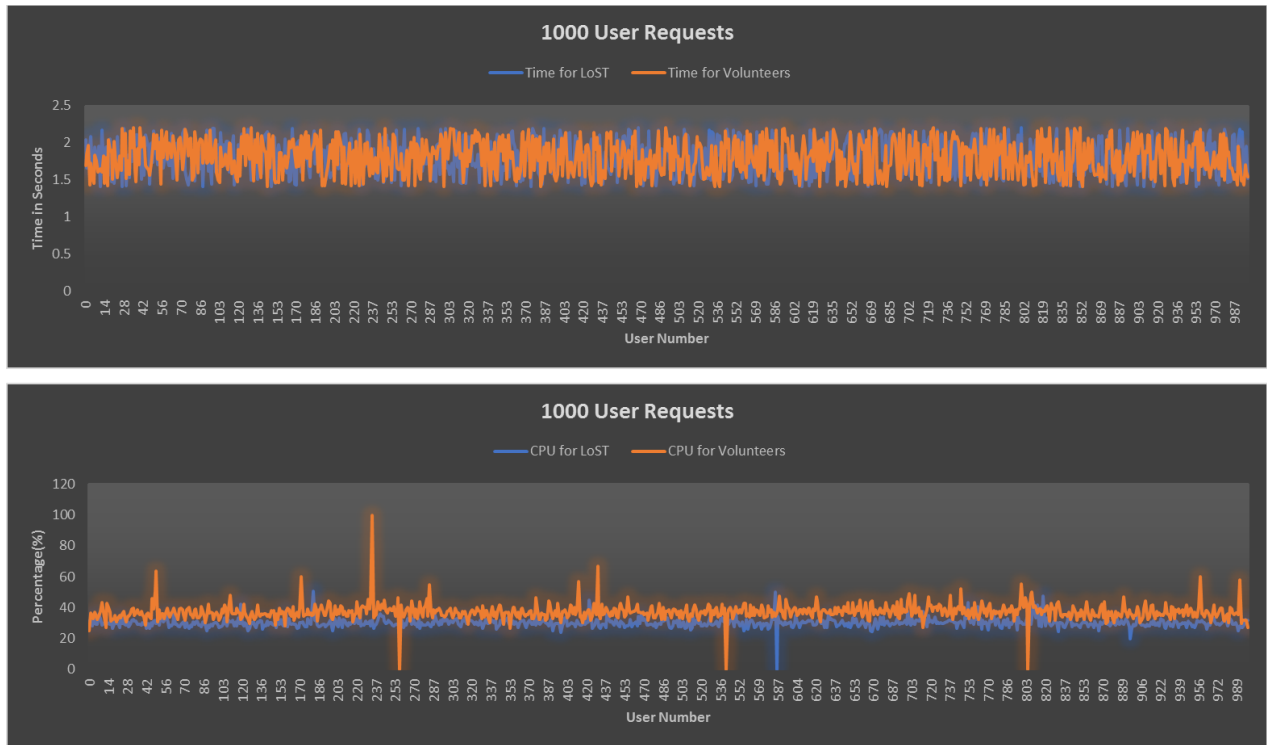


Figure 26 1000 user requests - Fog Implementation

Mean Time for LoST	1.80
Mean Time for Volunteers	1.79
Mean CPU for LoST	30.24
Mean CPU for Volunteers	37.24

Table 5 1000 user requests - Fog Implementation

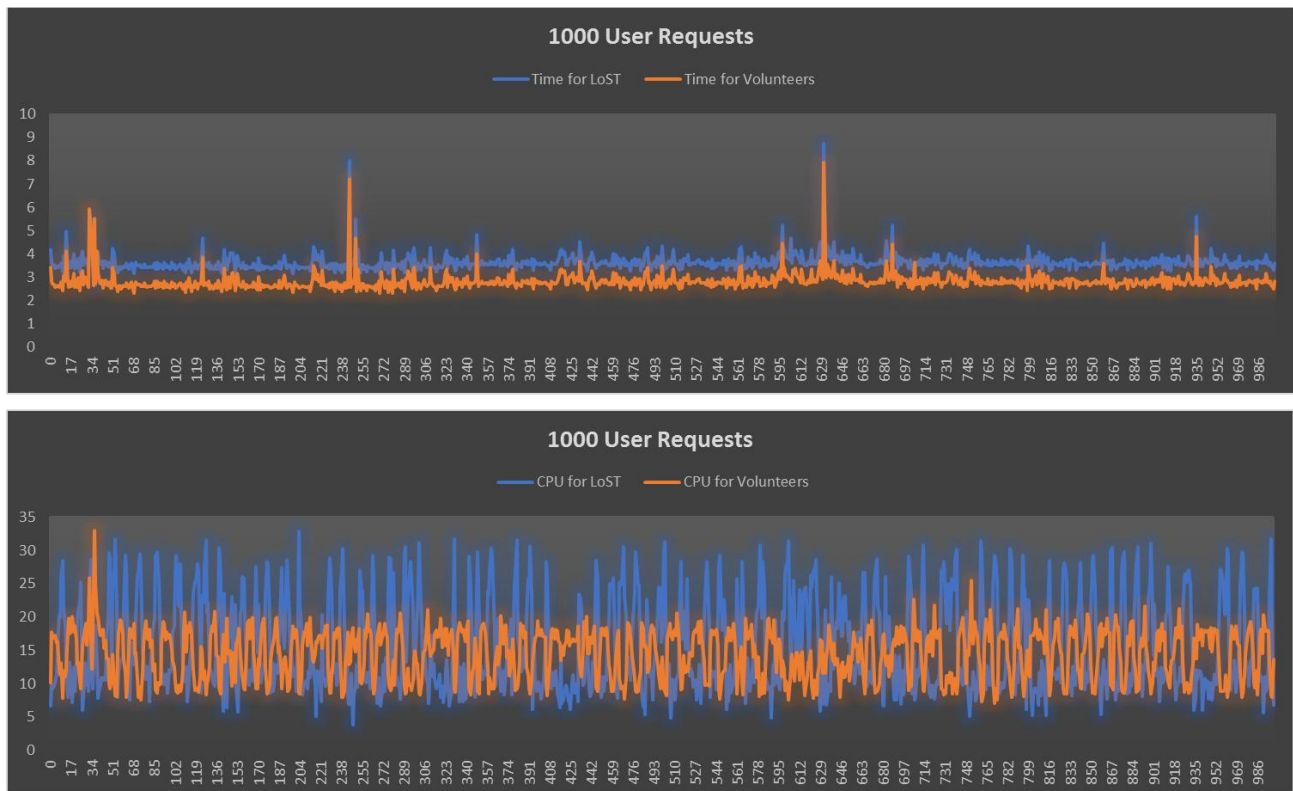


Figure 27 1000 user requests - Cloud Implementation

Mean Time for LoST	3.64
Mean Time for Volunteers	2.83
Mean CPU for LoST	15.83
Mean CPU for Volunteers	14.45

Table 6 1000 user requests - Cloud Implementation

5.4 Discussion

By examining the measurements of the above experiments, we come across some peculiar results that lead us to certain observations. Firstly, in the local server implementation, the system is faster. Yet, it seems that its' overall response time is dependent to the number of user requests. On the other hand, when the system is deployed in the cloud, the response time is rather stable, and does not seem to be influenced by the increase in user requests.

By observing the CPU usage charts, we deduct that the system consumes less computing resources (~14 - 15%) when implemented in the cloud. Contrary to that, the mean CPU usage of the local server implementation, is more than two times greater than the above value in average

(~30 - 37%). This behavior, is anticipated owing to the fact that the local server is a static-resources machine, able to perform certain non-resource-demanding tasks, whereas cloud servers are dedicated machines orchestrated by high-end entities, able to perform resource-intensive tasks in a more normalized and efficient manner.

To summarize, we observe that the local server implementation is faster, but the response time is unstable and dependent to the number of user requests. The mean CPU usage of the local server during the experiment is over thirty per cent (30%). On the other hand, the cloud implementation is slower, but the response time is not dependent to the number of user requests and remains stable throughout the whole experiment.

6 Conclusion – Future Work

In this thesis, we designed and implemented a virtualized, Cloud-based AAL system, to enable constant monitoring of indoor and outdoor position of patients, by health-cares and first-responders. Also, an emergency mechanism to alert those in charge (authorities, first responders) was proposed. Various techniques were discussed, such as proximity and cell identification, to make positioning possible. An emergency protocol was utilized for the retrieval of the nearest PSAPs and first responders a SIP-based VoIP communication channel was used for the alerting of those entities in the case of a patient drifting away from his/her premises.

The proposed system, can play a significant role in the AAL initiative. Nevertheless, the adoption of such a system, raises numerous implementation and coordination issues and challenges, that need to be tackled. The system's functionality relies on two basic entities, the LoST service and the Geolocation service, whose performance and robustness must be guaranteed. The former should be provided by a national authority, and the later, should be provided by an eligible application provider such as Google or Amazon. Additionally, international humanitarian organizations, such as the red cross, could provide the system with trained volunteers, explicitly qualified for emergency situations.

As a future objective, we will focus on adding telemedicine functionalities into the presented system, providing biometric measurements such as pulse, oxygen in blood (SPO₂), airflow (breathing), body temperature, electrocardiogram (ECG), glucometer, galvanic skin response (GSR - sweating), blood pressure (sphygmomanometer), patient position (accelerometer) and muscle/electromyography sensor (EMG), enhancing the patient's context, helping the system to evolve into a dangerous activity or health decline prediction system.

Bibliography

- [1] R. Li, B. Lu, and K. D. McDonald-Maier, “Cognitive assisted living ambient system: A survey,” *Digit. Commun. Networks*, vol. 1, no. 4, pp. 229–252, 2015.
- [2] J. L. Falcó, E. Vaquerizo, L. Lain, J. I. Artigas, and A. Ibarz, “AmI and deployment considerations in AAL services provision for elderly independent living: the MonAMI project,” *Sensors (Basel)*, vol. 13, no. 7, pp. 8950–8976, 2013.
- [3] L. Burzagli, L. Di Fonzo, and P. L. Emiliani, “Services and Applications in an Ambient Assisted Living (AAL) Environment Design of Smart Environments : The Present Situation,” in *Universal Access in Human-Computer Interaction*, 2014, pp. 475–482.
- [4] F. O. vergaard Hansen, “Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes,” *Sensors (Basel)*, vol. 14, no. 3, pp. 4312–4341, Mar. 2014.
- [5] N. K. Vuong, S. Chan, C. T. Lau, and K. M. Lau, “Feasibility study of a real-time wandering detection algorithm for dementia patients,” *Proc. First ACM MobiHoc Work. Pervasive Wirel. Healthc. - MobileHealth '11*, p. 1, 2011.
- [6] T. Hardie, A. Newton, H. Schulzrinne, and H. Tschofenig, “LoST: A Location-to-Service Translation Protocol,” *Rfc*, no. 5222, 2008.
- [7] H. Sun, V. De Florio, N. Gui, and C. Blondia, “Promises and Challenges of Ambient Assisted Living Systems,” in *Sixth International Conference on Information Technology: New Generations*, 2009, pp. 1201–1207.
- [8] D. Norman, Arthur, *The Invisible Computer : Why Good Products*. Cambridge,: MIT Press, 1999.
- [9] J. Wan, C. Byrne, G. M. P. O’Hare, and M. J. O’Grady, “OutCare: Supporting dementia patients in outdoor scenarios,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6279 LNAI, no. PART 4, pp. 365–374.
- [10] P. Remagnino and G. L. Foresti, “Ambient Intelligence: A New Multidisciplinary Paradigm,” *IEEE Trans. Syst. Man, Cybern. - Part A Syst. Humans*, vol. 35, no. 1, pp. 1–6, Jan. 2005.
- [11] P. Rashidi and A. Mihailidis, “A survey on ambient-assisted living tools for older adults,” *IEEE J. Biomed. Heal. Informatics*, vol. 17, no. 3, pp. 579–590, May 2013.
- [12] S. Hanke, C. Mayer, O. Hoeffberger, H. Boos, R. Wichert, M.-R. Tazari, P. Wolf, and F. Furfari, “universAAL - An Open and Consolidated AAL Platform,” in *Ambient Assited Living 4 Deutscher AALKongress*, 2011, pp. 127–140.
- [13] S. Hanke, C. Mayer, O. Hoeffberger, H. Boos, R. Wichert, M.-R. Tazari, P. Wolf, and F. Furfari, “universAAL – An Open and Consolidated AAL Platform,” in *Ambient Assisted Living*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 127–140.

- [14] G. Virone and a Sixsmith, “Monitoring activity patterns and trends of older adults,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 2008, pp. 2071–4, 2008.
- [15] G. Virone and A. Sixsmith, “Activity Prediction for In-home Activity Monitoring,” *Intell. Environ. 2008 IET 4th Int. Conf.*, pp. 1–4, 2008.
- [16] A. Sixsmith, S. Meuller, F. Lull, and M. Klein, “SOPRANO—an ambient assisted living system for supporting older people at home,” in *Ambient Assistive Health ...*, no. June 2008, 2009, pp. 233–236.
- [17] D. López-De-Ipiña, S. Blanco, I. Díaz-De-Sarralde, and X. Laiseca, “A platform for a more widespread adoption of AAL,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6159 LNCS, pp. 250–253, 2010.
- [18] J. Rivero-Espinosaa, A. Iglesias-Pérez, J. A. Gutiérrez-Dueñas, and X. Rafael-Paloub, “SAAPHO: an AAL architecture to provide accessible and usable active aging services for the elderly,” *ACM SIGACCESS Access. Comput.*, no. 107, pp. 17–24, 2013.
- [19] J. Wan, C. Byrne, G. M. P. O’Hare, and M. J. O’Grady, “Orange alerts: Lessons from an outdoor case study,” *2011 5th Int. Conf. Pervasive Comput. Technol. Healthc. Work.*, pp. 446–451, 2011.
- [20] S. Fudickar and B. Schnor, “KopAL - A mobile orientation system for dementia patients,” in *Communications in Computer and Information Science*, 2009, vol. 53 CCIS, pp. 109–118.
- [21] J. E. M. H. Van Bronswijk, W. D. Kearns, and L. R. Normie, “ICT infrastructures in the aging society,” *Gerontechnology*, vol. 6, no. 3, pp. 129–134, Jul. 2007.
- [22] Ico, “Privacy by design,” *Meta*, vol. 3, no. March, pp. 267–274, 2008.
- [23] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities,” in *2009 International Conference on High Performance Computing & Simulation*, 2009, pp. 1–11.
- [24] V. Koufi, “Ubiquitous access to cloud emergency medical services,” ... *Appl. ...*, pp. 1–4, Nov. 2010.
- [25] M. Satyanarayanan, “The Emergence of Edge Computing,” *Computer (Long. Beach. Calif.)*, vol. 50, no. June, pp. 30–39, Jan. 2015.
- [26] H. Chang, A. Hari, S. Mukherjee, and T. V. Lakshman, “Bringing the cloud to the edge,” *Proc. - IEEE INFOCOM*, pp. 346–351, 2014.
- [27] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, “Edge-centric Computing: Vision and Challenges,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.
- [28] Y. Nikoloudakis, S. Panagiotakis, E. Markakis, E. Pallis, G. Mastorakis, C. X. Mavromoustakis, and C. Dobre, “A Fog-Based Emergency System for Smart Enhanced

Living Environments,” *IEEE Cloud Comput.*, vol. 3, no. 6, pp. 54–62, Nov. 2016.

- [29] Wikipedia, “Javascript Object Notation,” 2016. [Online]. Available: <https://en.wikipedia.org/wiki/JSON>. [Accessed: 19-Jan-2017].