Commonwealth Scientific &
Industrial Research Organisation
PERCCOM Master Program

Master's Thesis in
Pervasive Computing & COMmunications
for sustainable Development

Alexandre De Masi

LOAD BALANCING IN P2P SMARTPHONE BASED DISTRIBUTED IOT SYSTEMS

2015

Supervisors: *Prof. Arkady Zaslavsky (CSIRO)*
Dr. Prem Prakash Jayaraman (CSIRO)
Prof. Jari Porras (LTU)

Examiners: *Prof. Eric Rondeau (University of Lorraine)*
Prof. Jari Porras (Lappeenranta University of Technology)
Prof. Karl Anderson (Luleå University of Technology)

This thesis is prepared as part of an European Erasmus Mundus program
PERCCOM - **P**ervasive **C**omputing & **COM**munications for sustainable development.

Co-funded by the
Erasmus+ Programme
of the European Union

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).

Successful defence of this thesis is obligatory for graduation with the following national diplomas:

- Master in Complex Systems Engineering (University of Lorraine);
- Master of Science in Technology (Lappeenranta University of Techology);
- Degree of Master of Science (120 credits) Major: Computer Science and Engineering, Specialisation: Pervasive Computing and Communications for Sustainable Development (Luleå University of Technology).

ABSTRACT

Commonwealth Scientific &
Industrial Research Organisation
PERCCOM Master Program

Alexandre De Masi

Load balancing in P2P smartphone based distributed IoT systems

Master's Thesis - 2015.

80 pages, 26 figures, 8 table, and 2 appendices.

Keywords: IoT, Sensor, energy, P2P, edge, Android, load balancing, publish/subscribe

Context: With the new age of Internet of Things (IoT), object of everyday such as mobile smart devices start to be equipped with cheap sensors and low energy wireless communication capability. Nowadays mobile smart devices (phones, tablets) have become an ubiquitous device with everyone having access to at least one device. There is an opportunity to build innovative applications and services by exploiting these devices' untapped rechargeable energy, sensing and processing capabilities. **Goal:** In this thesis, we propose, develop, implement and evaluate LoadIoT a peer-to-peer load balancing scheme that can distribute tasks among plethora of mobile smart devices in the IoT world. **Method:** We develop and demonstrate an android-based proof of concept load-balancing application. We also present a model of the system which is used to validate the efficiency of the load balancing approach under varying application scenarios. **Results:** Load balancing concepts can be apply to IoT scenario linked to smart devices. It is able to reduce the traffic send to the Cloud and the energy consumption of the devices. **Conclusion:** The data acquired from the experimental outcomes enable us to determine the feasibility and cost-effectiveness of a load balanced P2P smart phone-based applications.

ACKNOWLEDGEMENT

During the redaction of this document, the construction of this project and my two years of master I had many times to work harder than ever before. I was lucky enough to always have being supported by great people in this process.

I would like to thanks the PERCCOM consortium to allowing me be part of this master. All the travels, the universities, the professors, the lessons and the expertise acquired during those two years have changed me.

All my gratitude to Professor Arkady Zaslavsky & Doctor Prem Jayaraman for their guidance, responses and help during this work.

Merci infiniment Maman, Papa et frérot pour votre soutien.

Special thanks to Baptiste and Stefanos. You helped me to change and becoming someone better.

PERCCOM was once in a lifetime event. I had the pleasure to meet and share wonderful moment with incredible people from the four corners of the world, you will never be forgotten. Thank you Rohan, Zainie, Vicky, Fia, Iqbal, Chandra, Dorine, Mohaimen, Vlad, Maike, Khoi, Fisayo, Stefanos and Baptiste.

Alexandre De Masi

CONTENTS

Acronyms

CoAP Constrained Application Protocol.

CPU Central Processing Unit.

CUPUS CloUd PUblish Subscribe.

DHT Distributed Hash Table.

DoS Denial of Services.

DTLS Datagram Transport Layer Security.

FIFO First In Firs Out.

GLONASS GLObal NAvigation Satellite System.

GPS Global Positioning System.

GSM Global System for Mobile Communications.

HTTP Hypertext Transmission Control Protocol.

IoT Internet of Things.

JVM Java Virtual Machine.

LE Low Energy.

LILO Last In Last Out.

LTE Long-Term Evolution.

M2M Machine to Machine.

MQTT Message Queuing Telemetry Transport.

NFC Near Field Communication.

OSGi Open Services Gateway initiative.

P2P peer-to-peer.

PAN Personal area network.

POC Proof Of Concept.

QoS Quality of Services.

RESTful Representational State Transfer.

RPC Remote Procedure Call.

SOA Service Oriented Architecture.

TCP Transmission Control Protocol.

Ubicomp ubiquitous computing.

UDP User Datagram Protocol.

UMTS Universal Mobile Telecommunication System.

UPS Uninterruptible Power Source.

URI Uniform Resource Identifier.

UUID Universally Unique IDentifier.

X-GSN Extended Global Sensor Networks.

1 Introduction

According to Gartner [1] the current number of connected object as part of smart cities is estimated to be 1.1 billion . The number of connected "Thing" is said to rise to 9.7 billion by 2020. However this devices will be most use in smart homes, smart commercial building, transport, utilities and future industry. The world of the future will be composed of billion of devices connected to Internet continuously transmitting data.

Smart cities infrastructure will be equipped to query devices, analyse the data and make decision, e.g. the table 1 present a future smart world scenario from Libelium [2]. Connected object will generate a large amount of data. Nowadays, most systems passively send everything to the Cloud to process and analyse the information there. However, the future IoT world will be a major source of Big Data creating huge data streams and will demand immediate responses. Edge data analysis and filtering is essential to manage the enormous amount of data generated by these devices in order to enhance efficiency in bandwidth, increase response time and conserve energy.

1.1 Problem Statement

Internet of Things is the new disruptive trend in the era of the smartphones and Internet. New devices with low energy communication abilities start to be integrated in everyday lives and activities. Some of them are used to track our day-to-day activities while others capture/monitor environmental phenomenon. For example, devices such as Fitbit and smart watches monitor user's activities to help them live a healthy lifestyle while other devices such as mobile smart phones, Raspberry Pi's, wireless sensor networks and similar smart devices are used for real-time cost efficient monitoring of environmental phenomenon such as temperature, humidity etc. It is a well known fact that the IoT world is set to be filled with tens of thousands of devices according to recent estimates by Gartner [1] (9.7 billion by 2020).

IoT has the potential to lead the next technological revolution laying the foundation for future smart cities and smart industries In the next ten years, all the new smart cities and smart factories will be equipped with IoT devices. The data acquired from the devices will help creating sustainable cities and better managed factories. IoT has the potential to improve human life. A number of e-health application are being developed for monitoring the human body and use this data to predict the need of medication, e.g. smart insulin

Table 1. Smart World Scenarios

Scenario	Description
Air Pollution	Control of CO2 and toxic gases emissions from : factories, cars and farms.
Offspring Care	Control of the young animal in farms to ensure health.
Sportsmen Care	Monitoring performance and vital signs.
Structural Health	Monitoring of building, bridges and historical monument vibration.
Smartphones Detection	Detect smartphones with the help of wireless technology.
Radiation Levels	Distributed system monitoring of radiation levels in nuclear power plant.
Traffic Congestion	Monitoring of traffic to optimise driving and walking route.
Smart Roads	Automatic alerts for accident, traffic jam or extreme climate conditions
Smart Lighting	Street light able to change lighting depending of the weather conditions.
Water Quality	Monitoring of rivers and seas.
Water Leakages	Monitoring of water pressure and liquid presence along pipes.
Waste Management	Detection of waste level in containers to optimise trash collection.

pump connected to a smartphone for creating a artificial pancreas [3] .

Smartphones and other smart devices have become a ubiquitous in the IoT world and are densely available. Today smartphones and smart devices come embedded with numerous on board sensors with different sensing capabilities. Most of the time, these devices have access to rechargeable energy. In most cases the energy, sensing and processing capability of these devices remain untapped. The smart device's excellent sensing and processing capabilities and the distributed nature of the device can be exploited to develop innovative applications and services such as crowd-sensing in smart cities.

Currently more than half of the global population resides in megacities. The transformation of megacities to smart cities [1] fuelled by IoT has already enabled the interconnection of the masses in a previously unseen scale. Smart cities will be equipped with billions of

[1] http://www.smartsantander.eu/

things with the potential to transform the planning and management of smart cities via the IoT bottom-up paradigm. The enormous amount of data produced by smart cities will be a major source of Big Data, creating huge data streams, that will demand immediate and context aware responses. Currently many smart city applications (e.g. crowd-sensing) rely on data and services hosted on remote clouds. However, in the future the amount of smart devices including new devices such as Google Glass will only increase, augmenting the amount of data going to the cloud for processing. Pushing data to the Cloud is expensive and can causes multiple problem, eg. data explosion. It will also push the existing network bandwidth and processing demands significantly. Enhancing and upgrading existing resources to overcome this challenge come at a significant cost. Because of this new challenge, local data analysis and filtering is essential to manage the enormous amount of data generated by smart IoT devices in order to enhance efficiency in bandwidth, increase response time and conserve energy.

There are many challenges in exploiting smartphones to built complex and innovative IoT applications. Given the sheer increase in number of devices contributing to Big Data, one of the key challenge is to enable cost-efficient sensing and processing of data on mobile smart phones. This resources can be used without a significant cost of the battery usage for working on tasks. One of those task is data mining with the help of IoT devices, it is call edge mining [4]. The mining is done on the edge of the network, which is comprised IoT object and other smart devices. Data mining for the IoT world present different challenges as the survey from Tsai et al. [5] discuss. The development of pre-filtering process on IoT object is similar to the signal processing problems already know, e.g. data fusion, data abstraction and data summarising. Goura et al. [4] propose an algorithm to only transmit unexpected information from IoT smart devices, the General Spanish Inquisition Protocol (G-SIP). Their work is base on the Spanish Inquisition Protocol of Goldsmith et al. [6]. A number of model based transmission method for wireless sensor network can be applied to limit the communication of resource constrain smart devices.

Recently, *Edge Computing* and *Mobile Edge Computing* [7] have been proposed as a promising technology to overcome the above challenges. Edge computing [8] aims to lower cost and achieve energy efficiency by processing data closer to the data source. Edge computing is also refereed as fog computing by Cisco [9]. The definition of *mobile edge computing* is in the context of telecommunication networks where the processing of data originating from things are processed at the base stations. However, this definition fails to capture the true sense of our notion of mobile edge computing in an IoT context. Hence, we use the term mobile far-edge computing (MFEC) where the processing is further pushed into the *things* layer. The figure 1 is a comparison between cloud

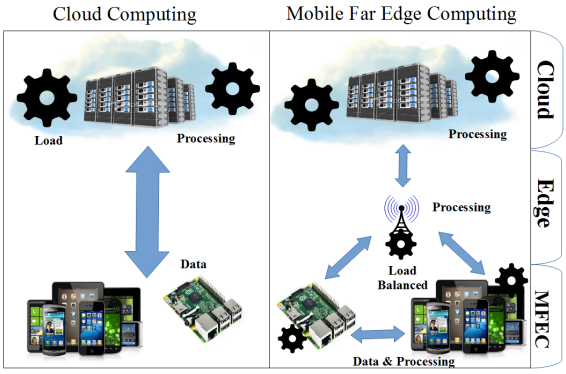


Figure 1. Cloud & Mobile Far-Edge computing architecture

computing architecture and our mobile far-edge computing architecture. In classic cloud architectures, the data is sent to the cloud for processing, contrary to mobile far-edge computing architecture where processing is distributed across three layers namely cloud, edge devices (network operator) and mobile far-edge devices (things producing data).

In edge system a middleware is often used for facilitate the communication between platforms of smart device. The middleware solutions for IoT integrate energy management program in the cloud responsible for computing and making decisions on which end devices information needs to be accessed. Such architectures are based on the client/server paradigm. Most of the time, the smartphones in the IoT world are used to control or access sensor information but not for processing the data. Such approaches do not take advantage of the smartphones advanced technological capabilities. We move further into far-edge devices which include end devices such as smartphones, sensors, Raspberry Pi and other smart devices.

The amount of protocol created or extended for the IoT world is always going up. Fragmentation and use of private licensed protocol is one of the challenge that IoT is facing. The standardisation of protocols and framework is a time consuming effort that require cooperation among many organisation.

The ability to use the power of a cluster of mobile smart devices for far-edge computing to process data from and on smart devices will be a add-on to the world of tomorrow. A load balancing scheme is needed to optimise the use of each devices depending of their resources, particularly in far-edge computing scenario with mobile resource constrained smart devices.

1.2 Objective

In the far-edge layer, smart devices come embedded with numerous on board sensors with multiple sensing capabilities. Most of the time, these devices have access to rechargeable energy sources and in most cases the devices have excellent sensing and processing capabilities. In this work, we propose LoadIoT, a peer-to-peer energy-efficient load balancing scheme for mobile smart devices distributed in the edge and far-edge layers. The distributed nature of the device offers the convenience in developing a load-balanced platform to achieve specific tasks closer to the source of data. The proposed LoadIoT scheme employs a publish-subscribe message exchange pattern over a peer-to-peer (P2P) network.

1. A novel energy-efficient load-balancing algorithm LoadIoT to distributed tasks between a peer-to-peer mobile smart devices in IoT ecosystems
2. A proof-of-concept implementation of LoadIoT on android platform
3. Experimentation and evaluation of LoadIoT's energy efficiency and performance

1.2.1 Research question

The aim of this thesis is addressing is the development of load balancing technique for P2P network of mobile smart devices in the Internet of Things paradigm allowing load-balanced distribution of tasks among peers. The exchange of information between the different entities in this project is one the higher concern in the study. Energy efficiency is a important point of consideration during the design and development of the thesis's system. The drains of the battery mobile smart device is a factor taken into account in the architecture of the work.

1.2.2 Research method

In the first part of this master thesis, we are presenting and commenting the state of the art research in the domains of the project. The domains of the project are P2P network, edge computing for Internet Of Things, messaging pattern and load balancing in distributed system. In each of those we analyse the current stage of the art to identify gaps and propose our noble solution. We propose LoadIoT a load balancing scheme taking into

consideration energy efficiency and other specific factor for load balancing on resource constrained smart devices.

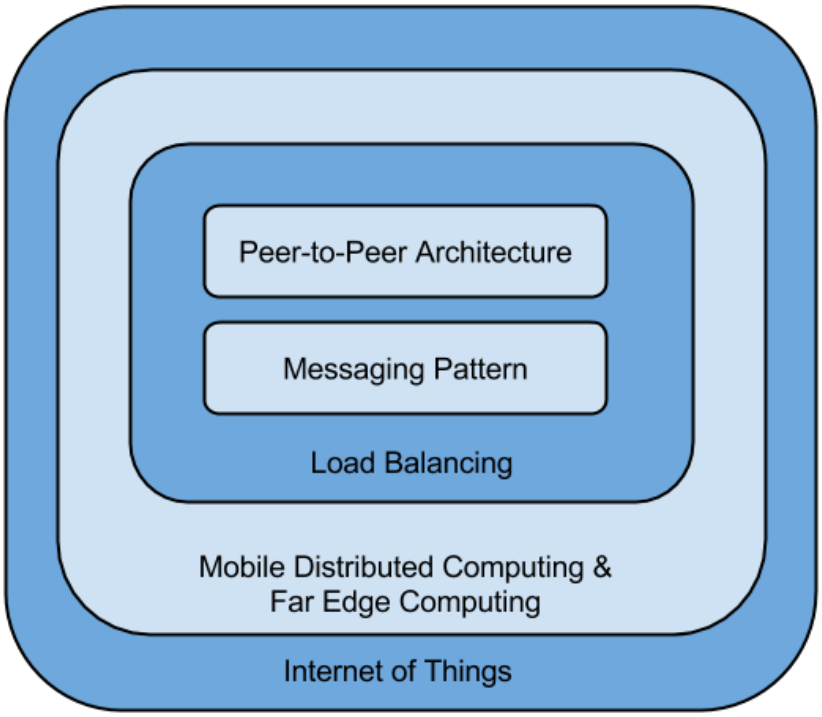


Figure 2. Domain of the study

We implement a proof of concept application tested with the load balancing scheme. To obtain a well define scope and problem space of the study, we assume and impose limitation during the implementation of the proof of concept application. We experiment to validate the efficiency of our work. The collection of the data output will help us to understand if this work can add to the state of the art and improve edge computing for smart devices in the IoT world. The system is put in a specific testing environment with a definite number of peers. We propose a model of the load balancing algorithm to validate the feasibility of the system and estimate the message complexity of the approach. We then use the model to estimate the performance of bigger scale scenarios.

1.3 Scenario

Edge computing on smart devices can use load balancing scheme in two main different scenarios such as :

- Distributed processing for Big Data application
- Energy efficient mobile wireless sensor

1.3.1 Facial Recognition

The number of devices generating data connected to the Internet is growing everyday. One of the application of edge computing is facial recognition in crowdsensing scenario. The authorities are searching for a individual in a crowd, no security camera are available. All the people present in the crowd use their smartphones to take a video of their surrounding. Because uploading and processing the information on the cloud takes time, every person is connected to Internet via a wireless 3G/4G connection to the same antennas. Depending on the network operators, the possibility of bottleneck is high. The load balancing scheme can distribute the video processing on the smart devices with in the peers network. One of the processing task could be pre-filtering the videos extracting the faces from the frame of the video and only sending this information to the cloud for further analysis and comparing the faces to find the individual.

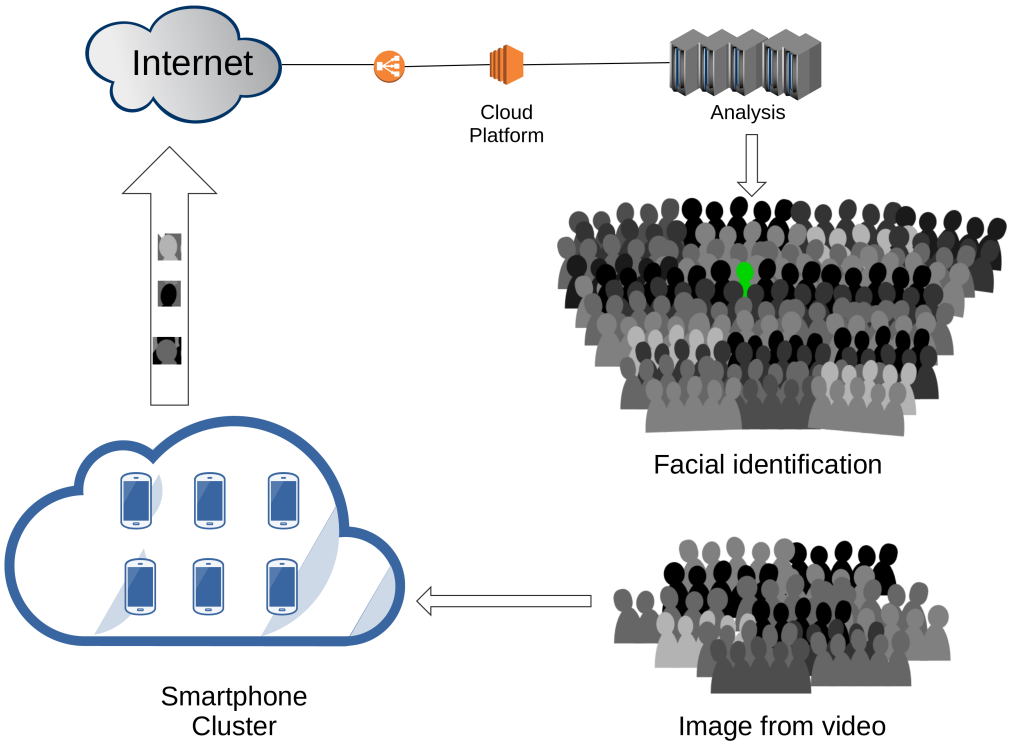


Figure 3. Crowd pre-filtering processing for facial recognition

1.3.2 Air Monitoring

Crowdsensing is part of the next generation of cities. Also, air monitoring is one of the challenges of the future world. Air quality in cities can cause health problem, e.g. pulmonary deceases. However in the future, smart devices will be equip with different sensor type, e.g. air sensors. Crowd with smartphones with this type of sensors are able to get air quality information depending on their surrounding. A important amount of data being generated by the sensors can become a problem, the load balancing scheme is able to use the smart devices with more energy available. Also, pre-filtering applications are possible to reduce the amount of data send to the cloud, e.g. Big data processing. MapReduce on a cluster of smartphone is a possibility, the load balancing scheme allows better managing of the resources in this case.

1.4 Thesis Structure

In chapter 2, we present the related work. Chapter 3 introduces LoadIoT, the scheme for load balancing in P2P distributed system for Internet of Things (IoT). The chapter 4 presents LoadIoT proof-of-concept implementation. Chapter 5 contains the experiments with the Android application implementing LoadIoT and evaluates the performance of the LoadIoT algorithm in term of message exchange and energy efficiency. The final chapter concludes the paper.

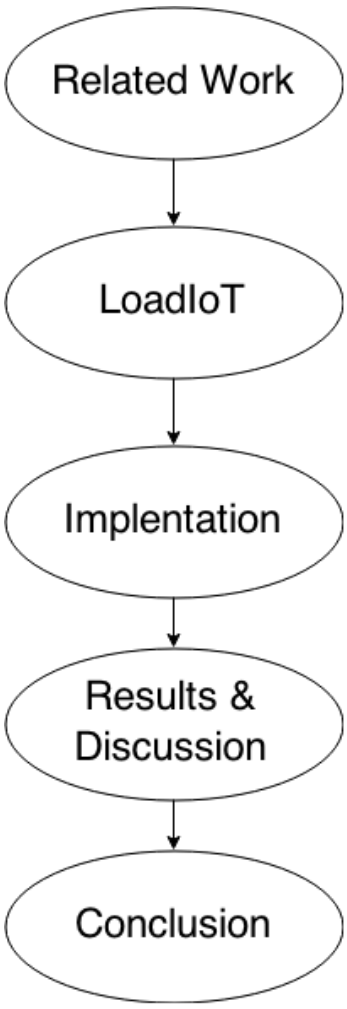


Figure 4. Thesis Structure

2 Related Work

Internet of Things (IoT) was coined by Kevin Ashton [10], during the dot com bubble in 1999. He used it to explain how everyday object connected to Internet will change the human life. IoT is now used to define the network of physical object, also call "things" for non denominate object, that is able to connect and transport information on the Internet. These devices have two main characteristics namely sensing/actuation and communication capability. An IoT device can be a sensor or an actuator or both. The sensor will generate data and the actuator will change the state of the system. Smart IoT object are able to make decision base on sensors reading, e.g. smart cart in smart cities can use different path to reduce road congestion [11] . They are the base for Future Internet and services offered in the future smart homes and smart cities. Building and home automation are fields were IoT object are present nowadays. One of the goal of IoT is to enable Machine to Machine (M2M)[2] communication and the creation of autonomous smart system that can manage the cities and its resources of the future. In this chapter we present the state of this art in multiple discipline relative to our work.

2.0.1 IoT World

Smart object are becoming more available every days. The market of IoT wearable, smart object that can be worn on a person are now mainstream, from smartwatch and smartband to smartglass. The smartwatch and smartbands are use for tracking physical phenomenon like heartbeat, number of step taken and other information. Recently, these devices have contributed to the development of innovative commercial health applications. The person wearing the device has access to a panel of information that he or she did not have before. From this data, it is possible to know the physical and health status of the person.

IoT non wearable devices are also starting to change the life of household. The smart thermostat [12] are enabling a better management of resource, e.g. the Google thermostat (Nest). It can interfere in people activity by inter-connecting to a Cloud platform to change the temperature from anywhere.

This smart thermostat couple with other smart object like smart meters for electricity, water and gas connected together create the house of the future. The house is able to detect leak of resource by comparing the data from the last month with the current consumption.

[2]Technologies allowing autonomous systems to communicate with each over.

Because of this data, the people living in the house will have a better understanding of how much energy they consume everyday in real-time. Currently, nobody is looking at the house water meter after a shower to know how much she or he consumed. If there is a simple widget on a smartphone or a tablet, the user will have a better way to visualise his/her own consumption.

IoT has much wider applications cutting across numerous domain. The industry of transportation can use IoT system to enable smart traffic control, electronic toll collection systems, logistic and fleet management, vehicle control and smart parking. IoT is also a tool for smart industries, intelligent manufacturing through performance traceability and digital agriculture. Monitoring of the environment is one of the other use cases for IoT. The most common use case is air quality monitoring. Smartphones are are one the major player in the IoT World, which include any type of smart devices. One of the challenges is how to manage the data generated by the IoT object. Big Data paradigm resolve this issue for processing and managing the data, e.g. Hadoop and MapReduce offer advantage for this kind of scenario as discuss by Demchenko et al. [13].

The research subject in the IoT World are multiple, one of them is crowdsensing. is a vast area of research for multiple laboratory in the world. One of the project to construct and to standardise a framework for the Internet Of Thing is call OpenIoT [14].

Constrained Application Protocol (CoAP) is a Representational State Transfer (RESTful) [3] protocol create for the Internet Of Thing, it use User Datagram Protocol (UDP) or Datagram Transport Layer Security (DTLS)[4] to connect to a sensor and requesting information. DTLS is a secure way to communicate using datagram. This protocol was influence by the HTTP protocol, the use of URI for making a query and sensing a response back. The actor requesting the data can also use the CoAP extension name Observe [15] to have the sensor itself send the data to the actor without the need of constant request on its part. CoAP can be use to create sensor / actuator binding solution with the need of a Internet connection or a Cloud Service. The RESTful advantage of CoAP provide the binding creation and execution for this kind M2M system [16]. The protocol provide a simple way to discovery the data available on sensor, by accessing a specific Uniform Resource Identifier (URI) the response from the sensor will contain the list, type and other information about the devices (well known URI). The Java implementation of CoAP call Californium oppose to state-of-the-art HTTP Web servers in benchmark, CoAP as a throughput of 64 times higher than HTTP on the same hardware [17] There is overwhelming evidence corroborating the notion that CoAP is a lightweight protocol for the IoT world. The use of

[3] Guidelines and best practises for creating scalable web services
[4] UDP with a security layer

UDP instead of Transmission Control Protocol (TCP) like Hypertext Transmission Control Protocol (HTTP) is one of the reason that make CoAP a lightweight protocol. But Message Queuing Telemetry Transport (MQTT) was also create for the same use, a M2M communication protocol for IoT. One of the fundamental difference between both of them is in the MQTT design, the protocol use a publish/subscribe messaging transport to exchange information and is base on TCP [18]. De Caro et al. [19]provide evidence in their comparison of MQTT and CoAP that there is no best protocol between the two of them. They each have their own particularity that can be useful depending on the scenario and the resources available. One of the other subsystem part of OpenIoT for data collection is CloUd PUblish Subscribe (CUPUS).

2.1 Distributed Computing

Distributed Computing is the use of hardware or software resources connected to a network to execute a job, contrary to parallel computing system as discuss by Coulouris et al. [20]. The figure 5 represent both architecture type. The job is derived in tasks and send to the computing entities on the network also call nodes or peers. The entities exchange messages about the running tasks and their characteristics. When all the task are done, the job is finished and a result is provided. It is a distributed system. From banking to e-commerce, to manufacturing and other industry, those systems are nowadays everywhere [20]. The main motivation for creating distributed computing system is resource sharing. Resources can be web pages, Central Processing Unit (CPU) time, network bandwidth or database record.

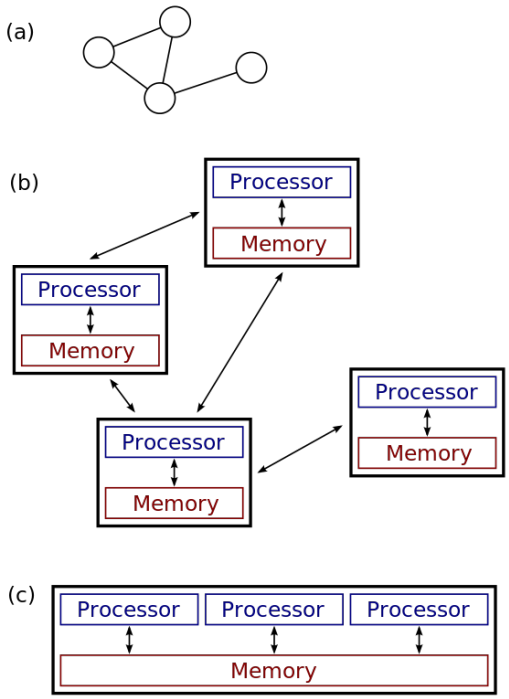


Figure 5. a & b : distributed system
c : parallel system

However the construction of this systems create challenges as describe by Coulouris et al. [20]: heterogeneity, openness, security, failure handling, concurrency, transparency and quality of service. Also this challenges apply to the computing paradigm use in IoT World.

2.1.1 Challenges

The heterogeneity of a distributed system can be found in the different type network (Wifi, Ethernet, 3G, ...) , operating system, hardware of the entities and programming language in place to create a distributed computing application. The openness of this system can make the difference between an extensible system, simple to add component and functionality to a close one where it is impossible to integrate change. The security concern for the systems running on public and open network can solve with encryption, cryptography with modern computer is simpler because it has been added to the CPU hardware. But this does not protect the system again traditional network attack e.g. Denial of Services (DoS) A distributed system is only scalable if the cost of adding an entity is less than the resources that the entity provide to the entire system. If an entity fail in the system, measure needs to be in place to respond to the failure and try to obtain the same

level of services as before the failure. In a distributed system the resource are shared between the nodes. During the design and implementation of a distributed middleware all the exceptions must be define and handle so that the application using the middleware does not have to worry about the underling layer abstraction. The Quality of Services (QoS) in distributed system is the guarantee that the services respect a predetermined agreement via defined parameters, e.g. performance, reliability and security.

The research into distributed system is always evolving, cyberforaging and mobile distributed system are now a reality.

2.2 IoT Application

The application domain of IoT are multiple because of the wide range of services for smart cities and the world of tomorrow that IoT can offer. Smartphones are widely available and contains communication capabilities and processing power. Also aggregating a cluster of smartphone on a specific IoT related task has the potential to accelerate the evolution of cities to smart cities without the need of developing new platform to compute and transfer the data from IoT object.

One of the challenge of mobile distributed system is the energy parameter to take into account. Because it functioning with a battery, it has a finite energy resources contrary to other system that are connected to a power grip and/or a Uninterruptible Power Source (UPS)[5] system. Hao Qian et al. propose Jade [21] a computation offloading system for wireless ad-hoc networked mobile devices. The mobile devices are tablet and high end smartphone that will do the computing. Their present the integration of Jade for different Android application. The runtime engine can schedule where the code will be execute and incorporates energy and performance parameters for a heterogeneous cluster of devices. Trobec et al. [22] describe in their paper the problem of energy efficiency in large scale distributed system, they explain the two type of measurement possible to know how much cost energy wise : hardware-based or software-based. It reinforce the fact that energy is one complex problem in mobile distributed computing. Chen et al. [23] discuss the challenges of resource allocation in wireless distributed computing networks. The authors claims that there is two parameters to examine : the communication and the computing power consumption. From it is possible to predict the cost of offloading and chose the best power-rate ratio for maximising the processing capability and minimising the power consumption.

[5]Battery banks with or without fuel generator

2.2.1 Edge Computing in IoT

Before edge computing and cloud computing, cyberforaging was the solution to process computing load from resource constrained devices on powerful server. The term cybergoraging was first described by Satyanarayanan [24] in 2001 as part of the core of pervasive computing also name ubiquitous computing (Ubicomp)[6]. Cyberforaging is a technique where mobile devices with small resources send tasks to heavier machine close to them. In this paper the author asses the challenge of cyberforaging : context awareness, privacy and trust in distributed systems and mobile computing. Some scenarios of the use of cyberforaging are also describe, in every scenario mobile devices with low computing hardware use surrogate, stronger computing system in their close environment to execute process. The result of the process is after send to the mobile device.

Distributed system in cyberforaging are design with a traditional client/server architecture, however a using a peer-to-peer architecture can be implemented. Skodzik et al. [25] present DuDE a is a distributed computing system using a decentralised peer-to-peer environment. It integrate peer discovery using Kademlia protocol. The author also present a data sharing algorithm using a Distributed Hash Table (DHT) ring implementation. DuDE was create for high performance distributed computing. It is one of many P2P distributed system. Bourgeois et al. [26] propose a simulation tool for P2P distributed system that aims to predict the performance and the execution time of a distributed application before its finalisation. It is one of the criteria to take into account during the development of this kind of system. Performance and execution time have a great importance in distributed system for the selection of the surrogate. The performance can be for CPU and memory usage, bandwidth capacity or remaining battery for mobile nodes. One of the recurring question in cyberforaging and mobile offloading is there any benefits to not send the task to the surrogate and to to process locally. The work of Datla et al. [27] list the challenges of a wireless distributed computing. They use a standard scenario with real-time data capture, the processing of the data and their dissemination. At the end if the energy needed to send the task to a surrogate is higher than the energy needed to process the task locally, the task should not be send. Of course other parameters are to be examine, e.g. the processing on the mobile device has to the transparent for the user. Cost-benefit analysis need to be done to determine the benefits, Kondo et. al [28] did it for Cloud Computing versus Desktop Grids. Scheduling and workload allocation are the main challenges.

Middlewares are often used to implement a distributed system has an underling layer of a existing application. The AIOLOS Middleware from Verbelen et al. [29] enable cy-

[6]Transparent computing everywhere and at anytime

bergforaging for java application using offloadable classes with Java Open Services Gateway initiative (OSGi) framework. In the paper presenting the middleware the authors want to improve the performance of mobile application through cyberforaging. In this system they use a cloud infrastructure has surrogate to handle the task processing. Ou et al. [30] propose the similar middleware with offloading code approach. Their algorithm work integrate the cost of CPU cycle, memory and bandwidth resources. Different middlewares component handle all the scheduling and allocation required. Offloading classes of code can be a simple way of implementing cyberforaging for an existing application. However the limitation is, the classes are often tied to a programming language. The difference in system and setup is a problem. There is no interoperability between system. Arslan et al. [31] propose offloading of java classes for their distributed computing infrastructure using smartphones. The smartphones are the nodes doing the processing. In their work they propose to use in smartphone given by enterprise to the employees during the night when it is charging, in the idle period. The authors show a system with less energy consumption. However, this fact is base of the evolution of the smartphone processors with ARM architecture. For the same computation ARM use less energy that a server on other architecture. The issue that has not been addressed in the paper is the communication between the smartphone and the company server.

Kristensen et al. [32] create an entire library and application for cyberforaging. It is written in Python (Stackless). The mobile application running of the smartphone have in their program a call to a surrogate to execute a process, if a surrogate is available in the close environment the task is send to the surrogate. The surrogate return the result after the end of execution. Their example is a image filtering application. The result of their experiment prove the advantages of offloading the process when it must filter high quality image. The application name is Scavenger [33], it use Remote Procedure Call (RPC) to communicate between the surrogates and the mobile application. The scheduler in the program decides whether to do the job locally or remotely. It is a dual-profiling scheduler using adaptive history-based profiling and the microthread feature offered by Stackless Python. Those components enable the scheduling system to create an execution plan later executed by the runtime. Scavenger scheduler use batch scheduling [34] to obtain better performances. It also present the advantage of remote execution for saving energy on mobile devices in [35]. Kristensen propose to improve pervasive positioning using three-tier cyberforaging in another paper [36]. Position is important in mobile environment, in cyberforaging it can determine the proximity and availability of surrogate for offloading. The method presented in the paper use Global System for Mobile Communications (GSM) properties to find the position. Nowadays, mobile devices, smartphones and tablet all integrate one or many satellite receiver for positioning e.g. Global Positioning System

(GPS), GLObal NAvigation Satellite System (GLONASS), Galileo ...

Busching et al. [37] propose the first prototype of Android smartphones cluster. Their work offer a first view of the capability of this type of system. For facilitating the Proof Of Concept (POC), the authors did not create a application for the smartphone but use a traditional distributed Linux application running into a chroot on all the smartphones part of the cluster. It does not offer any discovery, automation or native Android application.

2.2.2 Crowdsensing

Crowdsensing and crowdsourcing are two applications domains that are built on the IoT paradigm. The application Crowd Out running on smartphone, seen as a sensor device, from Aubry et al. [38] enable the user to report road safety issues and road offence to create a better and smarter circulation in the cities. The author also point that during the presentation of their work a privacy and security issue were raised. There has been an inconclusive debate about whether it will be used for creating smarter cities, building and other without impacting the privacy of the people. For example, a lot of concern were raised when smart electricity meter were installed in France. The electricity company would be able, with the data from the smart meter, to determine the daily schedule someone, if the person was in the house or not.

The paper from Cardone et al. [39] present McSense with the main with the following aspects: time, location,social interaction, service usage, and human activities. The platform is a sensing platform for smart cities . According to the authors, the main difference between McSense and the other crowdsourcing platforms can be found in the type of crowds it use, mobile for McSense, fixed for the others. Further, they are unable to exploit the sensors available onboard mobile devices ,e.g. smartphones, and do not have context-aware mechanisms, which are necessary for effective mobile sensing. Other research work as describe by Xiping Hu et al. in their paper [40] created a crowdsensing application case for Smart Cities running on Android OS. They use ontology-based matching instead of classical keyword matching. The data are acquired by the smartphone application and send to the Cloud for processing. The literature from Sherchan et al. [41] shows that the smartphones in crowdsensing scenario are often use to relay the data to the Cloud. However they have other properties that would enable them to run distributed algorithm for sensing, processing and computing information. The authors present a crowdsensing framework for location based social networking and citizen surveillance. In this framework the energy to get the data, the amount to transfer and the occurrence of the informa-

tion are taken into account to provide a solution with great energy efficiency of the sensing mobile device. This is done by using scheme to diminish the amount of data to transfer. One of the scheme is smart selecting of the devices that will provide the data. The result shows that by not selecting all the sensors for gathering the data, the mobile device have a greater battery life and the accuracy of the information are preserve. The survey from Madria et al. [42] presents the issues in dynamic data management in mobile P2P network for creating crowdsourcing application. The survey also asses the issues in non traditional point to point network, e.g. data cache and replication. A peer can disappear from the network, however caching and replication can be use to reduce the probability of loosing data. However, the survey also develop concern for privacy in service discovery scheme on mobile P2P network.

Antonic et. al. [43] propose a cloud based publish/subscribe middleware for a mobile crowdsensing ecosystem. This middleware CUPUS is part of the OpenIoT project. The publish/subscribe model is one of the method use for data gathering and filtering for crowsensing. The general principle is showed in the figure 6. It represent the logic of CUPUS in a sequence diagram, from the OpenIoT deliverable. In In this approach, matching subscription and publication is done in the cloud. This paper present a simple architecture with one main broker, also call CPSP Engine for Cloud-based publish/subscribe processing engine, less powerful mobile broker relay running on cloud instances and smartphones. The mobile element is presented as a relay between the sensors output and the subscribers. It will only process the matching algorithm from a local subscription list send by the cloud broker. CUPUS is one of the most advance middleware for crowsensing using a publish/subscribe model as delivery mechanism, but this system is centralised and does not take into account a scenario if the main broker can not be access. The mobile brokers are not aware or connected to each other.

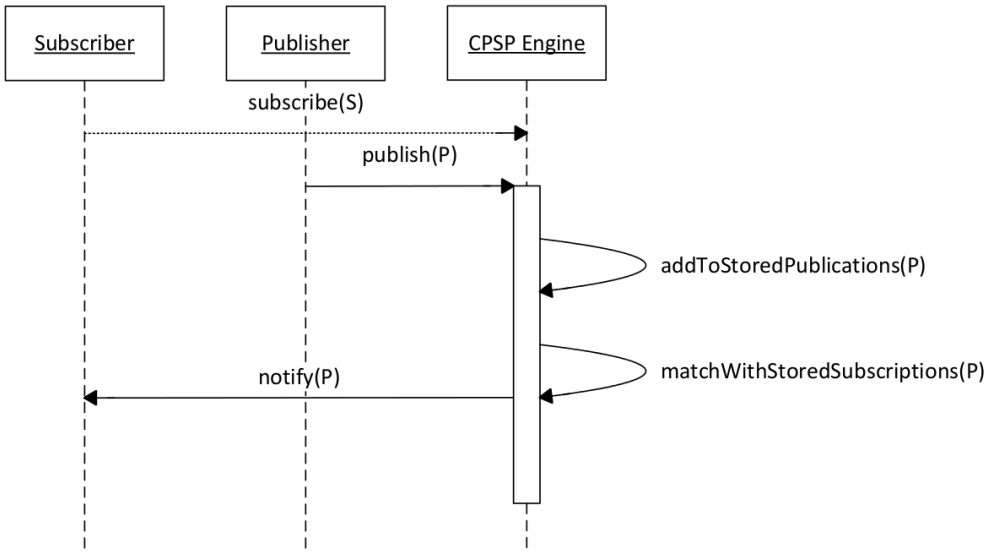


Figure 6. Sequence Diagram of CUPUS logic

Antonic et al. the authors of CUPUS did a demonstration of the system with OpenIoT in a real life scenario [44]. The experiment was call :"Sense the Zagreb Air". Android smartphone were connected to air quality sensors via Bluetooth. The reading were sent to the Cloud broker for matching and the data were send to the OpenIoT platform for presentation and analyse. However in [45] the authors add a level of management to CUPUS using a quality-driven sensor function middleware. The QoS Management Function (QSMF) has the ability in certain identified cases to reduce the energy consumption of the mobile broker connected to the Cloud engine by applying a sensor management scheme. The sensor management software knows the localisation of the mobile broker, with this data it can determine to which sensor to subscribe with a specific accuracy factor for a amount of time . The management platform will select another sensor in the delimited environment for the next subscription and so use only one to n sensor(s) at a time.

2.2.3 Challenges

The challenges of edge-computing are the same that any distributed system as discuss by Coulouris et al. [20]. heterogeneity, openness, security, failure handling, concurrency, transparency and quality of service.

2.2.4 Peer-to-Peer

peer-to-peer (P2P) systems have always been difficult to define. The RFC number 5694 [46] from November 2009, argues that a system is peer-to-peer only if the element of the system request services and provide services for and from the other elements. The systems respecting this definition have been around before 2009, e.g. Bittorrent [47] , Kazaa [48] and Gnutella [49] are the most common P2P protocols use for the last 12 years. The paper classify different type of P2P application : content distribution, distributed computing, collaboration and platform. The author claims a popular view about peer-to-peer distributed computing. It present the basic definition of a distributed system as a computing task divide into subtasks, send to peers for processing, once the work is done, the results of all the subtasks are return to carry out the main task. The author advice the choice for using a P2P architecture is case-by-case, depending of the application requirement, the security implementation and the trade-off that this kind of system can offer. AN important part of the P2P application running on smartphone as of today are only implementation of the some application available on a laptop, desktop or server. One of the first peer-to-peer network for mobile devices was described by Porras et al. [50]. In this paper, the authors use the bluetooth technology for creating network connection between peers and discuss the purposes of this type of system based on the mobility factor. They implemented a protocol named PeerHood based on devices in a Personal area network (PAN) connected via Bluetooth. The authors define the basic functionality for a node to integrate a P2P network has :

- Device discovery
- Service discovery
- Connection establishment
- Data transmission

This four functionalities are essential to create a peer-to-peer protocol.

2.2.5 Load Balancing

Load balancing in a peer-to-peer system can have a multitude of meaning, the term load can reference to process load, network traffic load or data load. Before going into load balancing and the algorithms that can be useful to implement.

Load balancing in a peer-to-peer system can have different definition depending of the context of the system, as describe by Felber et al. [51]. In a P2P system a load is the capacity of an object, e.g. bandwidth, storage space or processor. The author present in this survey a collection of methods for load balancing in P2P system with DHT. DHT is a distributed system of key value pair. Each node part of the DHT can retrieve the value (the data) with the given key. The mapping from keys to values is done by the nodes part of the network. It offers scalability, failure handling and extremely large numbers of nodes. The authors compare the load balancing mechanism for object placement, routing and the underlay protocols. The underlay is define as the network topology supporting the system, a load imbalance is probable between the peers depending of the path the messages take. A routing imbalance is created when the nodes have too much messages to route between the peers. There is a rapidly growing literature on DHT in P2P system, this is one of the main method to create a peer-to-peer network without the use a control entity or main directory, containing the shared object name and the peers who have it. DHT algorithm are implemented in most of the object placement P2P network like Bittorrent. The principle of the DHT is based on a hash table compose of a key value pair.A hash function is use on an identifier of a resource, e.g. a filename, to create a key. This key is paired with the resource to share and put in a message. The message is sent to the nodes participating in the DHT network until it reach the node(s) responsible for the key. The ownership of the key is split between the nodes. If a peer want a resource it will request the data from the network using the key, the overlay will route the request to the peer who has the resource wanted. The paper conclude on the fact there is none perfect load balancing mechanism in P2P DHT system for any kind of load.

Rao et al. [52] present three load balancing schemes build around DHT. The mechanisms presented use the notion of virtual server. A node can have multiple virtual server for different parts of the keyspace. One of the main plus of virtual server in a P2P DHT system is the implication for load balancing. The virtual server can be move or split around the network to a peers with more capacity to handle the load, e.g. a popular keyspace host by a poor resource server will be transferred to one with more capacity in term of bandwidth, storage and CPU. The three scheme are :

- One-to-One : transfer of virtual server from a node to another node
- One-to-Many : transfers of virtual server from a node to multiple nodes
- Many-to-Many : transfers of virtual server from multiple nodes to other nodes

The simulation result provide confirmatory evidence of the schemes effectiveness for load

balancing in this kind of system, from 80 % of the optimal value for the simpler scheme to 95 % for the more complex.

Godfrey et al. [53] proposed load balancing scheme in dynamic P2P system. The scheme proposed is based on virtual server and the combination of two previous scheme : One-to-One and Many-to-Many. The simulation of the new algorithm prove that the scheme is able to achieve load balancing for dynamic system with high use. The scalability of a heterogeneous system is easier to improve by reducing the number of virtual servers per node. Other approach aims without virtual servers for resolving the issue of load balancing in peer-to-peer system are define by Xu et al. [54] and with virtual server by Steele et al. [55]. Xu et al. describe the main elements responsible for effectiveness of load balancing applied to P2P DHT system. They are : peer heterogeneity, file access behaviours, and P2P overlay network topology. By maintaining an access history of the file their scheme is able to predict the future access behaviours. Using this information they are able to accurately split the workload when a new peer connect to the system. In this scheme the load redistribution is only made when necessary and not continuously, this way it reduce the overhead. The last part of the scheme is really interesting for mobile environment, where zone are defined. Depending of the probability of the future access to a file by a group of peers, the file is replicate in a zone close to the group. Doing so will balance the load on multiple peers. In the second article the author implement a parameter free approach for load balancing in P2P system. In load balancing algorithm one point is often the same, which peer is going to have more load and the reciprocity, which one will have less. The selection mechanism is not parameter free, it depend of the knowledge for each peer of the global capacity of the system for heterogeneity. The author resolve the problem by using a method to informs the peers about the capacity of each other randomly. With this data, each peer create a fixed size window of the recent sample to compute the selection parameter. After implementing the solution in the Swaplinks algorithm, they obtain simulation result proving the efficiency of this method.

Ledlie et al. [56] discuss load balancing around routing, instead of object placement contrary the other papers. They propose k-Choices, a load balancing algorithm who differ from the other by making the workload assignment explicit. This algorithm is able to give the right amount of work to new peer joining the system. This paper assess different load balancing method, but their method is better in realistic condition where the other offer poor performance.

Most of the load balancing method are based on the type of load to balance. There is a need for multidimensional load balancing approach. The survey from Felber et al. [51]

is an example of the challenges of load balancing. In a peer-to-peer system of any kind, the architect needs to define what is the most important load to balance, from that point a decision can be made of the algorithm to implement. In resource constrained environment, more factor will appear, e.g. energy consumption. None of the previous papers assess the energy needs of the algorithm .

P2P technology can be use media streaming, particularly for video streaming as describe by Lu et al. [57] and by Wichtlhuber et al. [58]. The authors of the first paper claim that a content provider peer-to-peer hybrid technology is more efficient to deliver live media that traditional P2P scheme. The load balancing here is not done by the peers but it is made by a central authority. In the second paper the authors try to determine a better way to obtain a energy efficient mobile streaming supported by a P2P network. The energy consumption of each devices participating in the streaming are taken into account. The devices that share the media and not only view it will consume more energy. One of the interesting part of this system is how is selected the bootstrapping host. As previously explain, a node with the capacity to join a P2P network can only do it if knows at least one address of an already peer member. In the paper they use one of the mobile communication technology available on most of the new smartphones :Near Field Communication (NFC) to send the bootstrap host address. NFC is nowadays normally use for contact less payment or other information exchange of short size. For becoming a peer and getting access to the P2P network, the devices are held back-to-back to exchange the bootstrapping information, other method exist , e.g. service discovery using multicast DNS, Siljanovski et al. [59] discuss the challenges in constrained network. The last method do not need human interaction on the user smartphone. It is preferable for transparent crowdsensing application.

2.2.6 Message exchange pattern

A message exchange pattern is in telecommunication the pattern of messages needed by a communication protocol to use a communication channel [60]. This domain as been research for Service Oriented Architecture (SOA) application. Two major message exchange patterns exist : request-reply and one-way. With request-reply the first entity send request message for information to a second entity. The second entity respond to the request with a reply. Such exchange are similar to a HTTP transaction for loading a web page. The one-way pattern does not need a response, e.g. UDP. A publish-scribe messaging channel pattern can be implemented with a set of one-way message. Shahnaz et al. [61] provide the advantages of this messaging channel pattern such as Many-to-

Many messaging, configurable QoS and loose coupling.

3 LoadIoT

Mobile far-edge computing is a novel approach to address the challenges posed by the emerging trend of IoT, particularly the Big data generated by *things*. We propose a load balancing scheme that works in the edge, far-edge layers enabling load-balanced processing of data closer to the source. The devices are able to take part in the processing if they satisfy conditions such as sufficient battery level, storage and CPU capability to perform processing. Firstly we presents some definitions that we used within the scheme of LoadIoT.

- *Load* A task that will be distributed among a set of mobile smart peer devices for execution.
- *Peer:* A smart device taking part in load-balancing operation. The assumption is, when a peer participates in the operation, it has sufficient power (energy, processing and memory) to handle the load.
- *Node:* A smart devices that will request information from the peer network but not a participant of the network.
- *Service:* Each peer participating in the load-balancing operation offer a list of services (e.g. task as sensing temperature using on-board temperature sensor, processing data) to other peers.
- *Smart Head* Peer responsible for managing and handling the load.
- *Subscription* Request of a service.

3.1 Load Balancing in Peer-2-Peer Mobile Smart Devices: An Illustrative Example

We consider a mobile crowd-sensing application scenario that captures the noise level in different locations in a smart city environment. Given the growth of IoT, its likely, in the future, there will be many devices providing this data. Collecting data from all the devices will be expensive due to unnecessary communication to cloud devices. We take a load-balanced approach to address this issue. A group of peer mobile smart devices (e.g. within the same location and connected to the same access point or the network) periodically

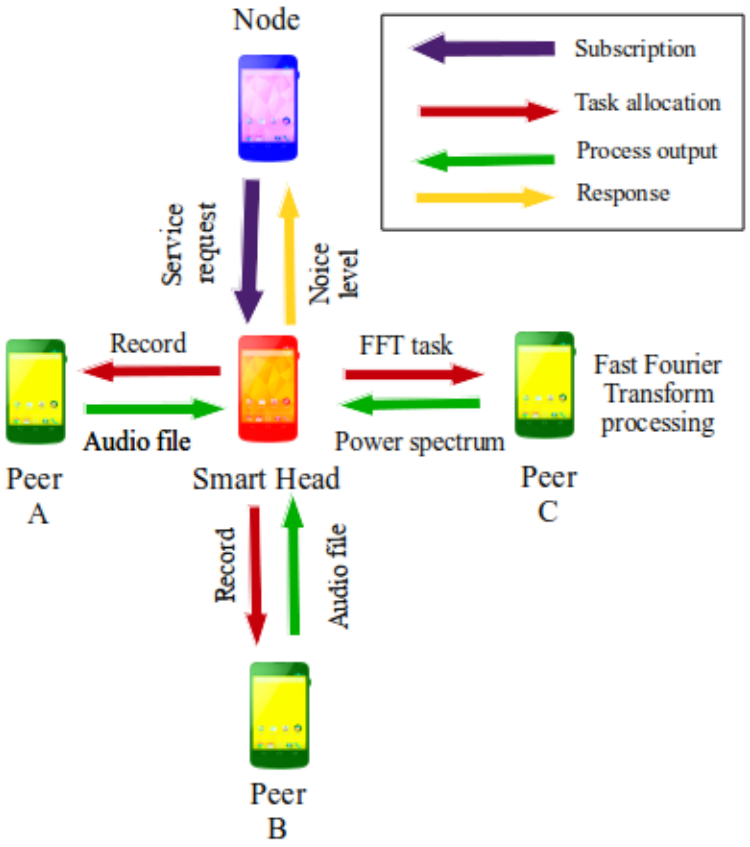


Figure 7. Noise level example

elect a smart head (leader) using the proposed load balancing scheme (LoadIoT). When requests from the users (subscription from node) is received, the smart head distribute the tasks to different peer each of which is responsible to process and respond to the subscribers request. We use the term producers to classify peers that provide services (data or processing capabilities).

The Figure 7 represent the scenario. A node (user) request the noise level in the surrounding environment. The smart head allocates the task to the peers (producers) A & B. Both will send to the smart head an audio file originating from their respective microphone. The smart head will then send it to peer C for processing. After peer C applies a Fast Fourier Transform to both audio signal, it will emit a report containing the noise level and the frequency to the smart head which is then forwarded to the node.

3.2 Basic

Peer-to-peer can have different meaning, in this project it us the way to express that each smartphone taking part in the cluster offer and use different services on and from the network peer-to-peer system can be really complex and difficult to implement. Nowadays peer-to-peer protocols most used are the ones for sharing files between people from different part of the globe using Internet. Bittorrent implemented a well design way to share information without the need of a centralised server. The protocol is use to connect billion of nodes to each other.

In our approach, we use a efficient way to exchange messages on the peer-to-peer network. We minimise the sending of information to reduce the energy consumption. It is possible due to the limitation imposed on the system.

The aim is to give to one smartphone, call the smart head, the task of allocating process. If the task process return a value in a specify range of accuracy, the smart head send the result of the process. The load can be share between the different peers part of the network. Each task available through services, e.g. in crowdsensing scenario the task is the access to data, e.g. temperature, humidity or noise. The task can be process locally by the smart head or send to other peers. The load balancing scheme is able take into account different factors, e.g. the number of present tasks, the battery usage, the CPU load of each peer, ... The smart head is elected for a specific slot of time. After the end of a timer, a new smart head is elected. This does not stop until all the task have been process or the peers are not able to fulfil the factor parameters, e.g. smart devices battery are empty or their threshold has been exceeded.

There is multiple type of messaging pattern for exchanging information in the networked world. One of them is the publish/subscribe mechanism, as previously described, it offers advantages and disadvantages. Generally the system has a broker to interconnect the sender and the receipt of the data. The broker distribute the responses and manage the core of the system, it is not a IoT sensor or a mobile component. The goal of the application is to be an energy efficient mobile publish/subscribe peer-to-peer system using a load balancing algorithm to retrieve IoT sensors data.

The publish/subscribe messaging pattern has been use in the past by other project in relation with IoT. It offer a two main advantage : loose coupling and scalability. The scalability of this type of system offers a lower level of failure, parallel operation processing, message catching and non continuous connection.

The load balancing scheme is able to work on any kind of smart devices. We are making the assumption that in the future a important number of things with sensors and processing capabilities will be available. Already, the ones on smartphones are becoming common. Even low end Android devices have a minimum of two to three types of sensors :

- Motion : accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.
- Environmental : barometers, photometers, and thermometers.
- Position : orientation sensors and magnetometers.

Those smartphones are equipped with Wi-Fi and Bluetooth technology, it enables them to connect to WAN and to *things* with Bluetooth Low Energy (BLE) capability, part of the Bluetooth 4.0 specification. This technology provide a network communication interface to low powered devices. The devices can run for a few months on button size battery, depending of factors : processing power, battery capacity, use of light and deep sleep mode. In the last two years a number of IoT devices arrived on the market, most are connected to Internet via a smartphone. Hardware manufacturer and big tech brand started to make smart watches (Moto 360 , Samsung Gear), connected light bulb (Hue), sport and health band (Fitbit, Jawbone, Fuel). Those devices are part of the Internet of Things, most of them rely on BLE and a smartphone for accessing Internet.

3.3 Architecture

3.3.1 Network Topology

Because this study is concentrate on the load balancing and not on the context awareness and positioning, the system use the define zones of the OpenIoT project. Every area has its own cluster of smartphone.

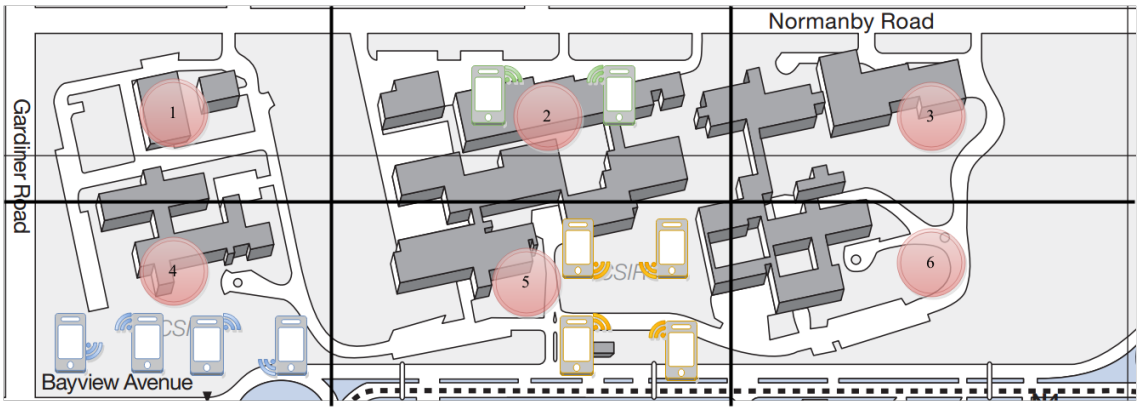


Figure 8. Smartphones in zones

In the figure 8, we can observe smartphones in delimited zones. A zone is space surface on a grid bonded by 3 to n point.

The type of communication network that support the system can be multiple :

- Bluetooth
- 3G/4G
- NFC
- Infrared
- WiFi

The type of communication and the network have to support the traffic load that the application will generate. Each of this communication technology consume energy at a different level and co : range, topology and accessibility.

Two of the communication technologies have by default a small range and bandwidth. For NFC, the smartphones needs to be back-to-back to exchange information. The NFC antenna is always in the back of a smartphone. For infrared communication the sender and the receiver need, as with NFC, to be place in a special way to obtain a direct line of sight for enabling the communication link to transmit and receive.

4G Long-Term Evolution (LTE)[7] and 3G Universal Mobile Telecommunication System (UMTS) network technologies offer a large range and a gateway to Internet. But this

[7]Standard for wireless high-speed data communication

technologies consume a lot of energy and their operator impose a number of different rule concerning their usage. They offer a simple way to communicate but have too many constraints.

Bluetooth in its last commercial version, Bluetooth 4.0 Low Energy (LE) , has the best ratio between limit range, size of the messages and energy consumption. Wifi is also a solution for the system. The smartphones can be connected together via a mesh or an access point via Bluetooth or Wifi. The energy consumption of Wifi is higher than Bluetooth but it offers a higher range of transmission and a higher bandwidth.

3.3.2 Smart Head

A peer-to-peer system offers a modularity and scalability that a traditional system can not. By designing the system as a peer-to-peer distributed platform it has a built in scalability factor. The system resources can be multiply without rewriting or adding new code to the application. Just by providing a new peer to the network, the system will be able to handle more load, in this case sensor information. However the complexity of the design and development of a peer-to-peer system is higher than a classical server client application.

Because of the augmentation of interest of mobile smart devices for services in the future, linked to new age of smart cities and smart world, a large amount of information will need to be process. One of the biggest concern can be found the communication in constrained resources system.

In this system we elect a smart head for each services and tasks handle by the system. The load is distributed to the peers part of the network. The load attribution depend of the result of the load balancing algorithm. A peer can only become the smart head after an election. The election is the moment when every peers who has the capabilities to become the next smart head send its candidacy on the network. The candidacy integrate the important information that the load balancing algorithm will need. Because we want to create a lightweight system to reduce the energy consumption, all the peers run the election process and determine if they are or not the new smart head.

The process of election is compose of 3 phases :

- Phase 1 : Send candidacy

- Phase 2 : Receive candidacy.
- Phase 3 : Run the load balancing algorithm to elect the smart head.

An election happen when a node wants to access services from peers and no smart head is in place. The LoadIoT scheme operate in two modes: *reactive* and *proactive*. The proactive mode triggers the election process at a predefined intervals time. The reactive mode starts the election process only when a smart head is required i.e. on a subscription. The proactive mode offer failure handling and can elect a smart head if the current one suddenly disappear after a timeout. The reactive mode can use the maximum of a peer before starting a new election.

In most advance load balancing algorithm, a scheduling policies is needed for queueing the load to the handling entity, e.g. First In Firs Out (FIFO), Last In Last Out (LILO). Queueing theory is an entire domain of knowledge, in this work we use a simple first in first out queue.

For presenting the service discovery and the smart head election, we propose to focus on the zone 5 of in figure 8. The figure 9 represent the trigger that start the election process in the standard mode of the system. The service discovery is available through the multicast network. A node asking for services send a multicast request, all the peers respond with a list of services available.

The node N1 has a task to distribute to a group of smartphones.No smart head is in place. After every peer receive the request of the services via a subscription, the election process begin. During the election state, shown with in orange colour in the figure 10, all the peers send their candidacy and wait to received the ones of the other electable peers, it stop to listen to candidacy after a timeout. The system create a list of peers candidacy. This list is only updated during election proceeding. The peer-to-peer system discuss in Related Work, have a list of active peer maintain by the system.

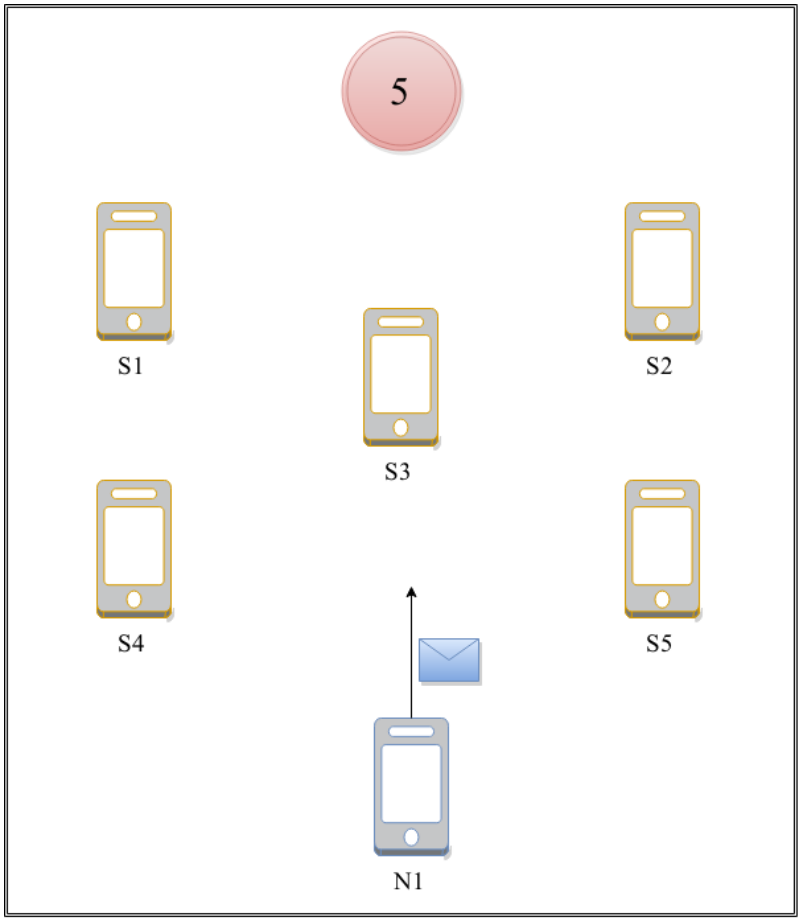


Figure 9. Subscription send by $N1$

Once the peers have all the candidacies, the load balancing algorithm determine the new smart head. In this example, $S4$ is elected. The other peers know who is the smart head for service wanted by $N1$.

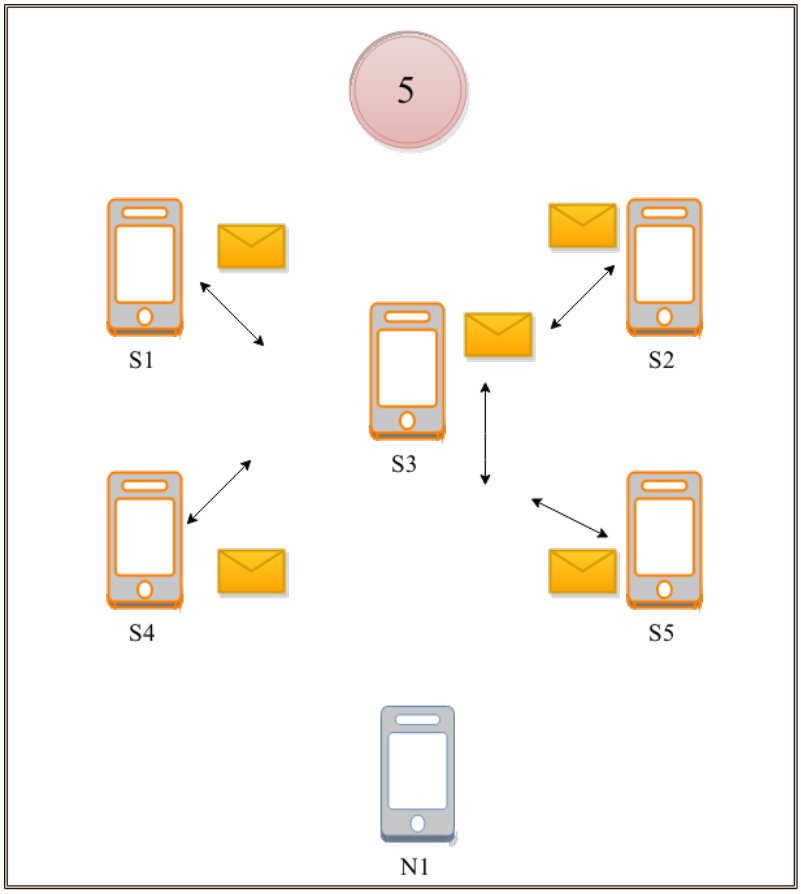


Figure 10. Election State

The figure 11 represent the state of the system after the election, in red the smart head and in green the other peers in a standby mode, waiting for the next election. $S4$ is the peer managing the load and sending the process result in a notification to $N1$.

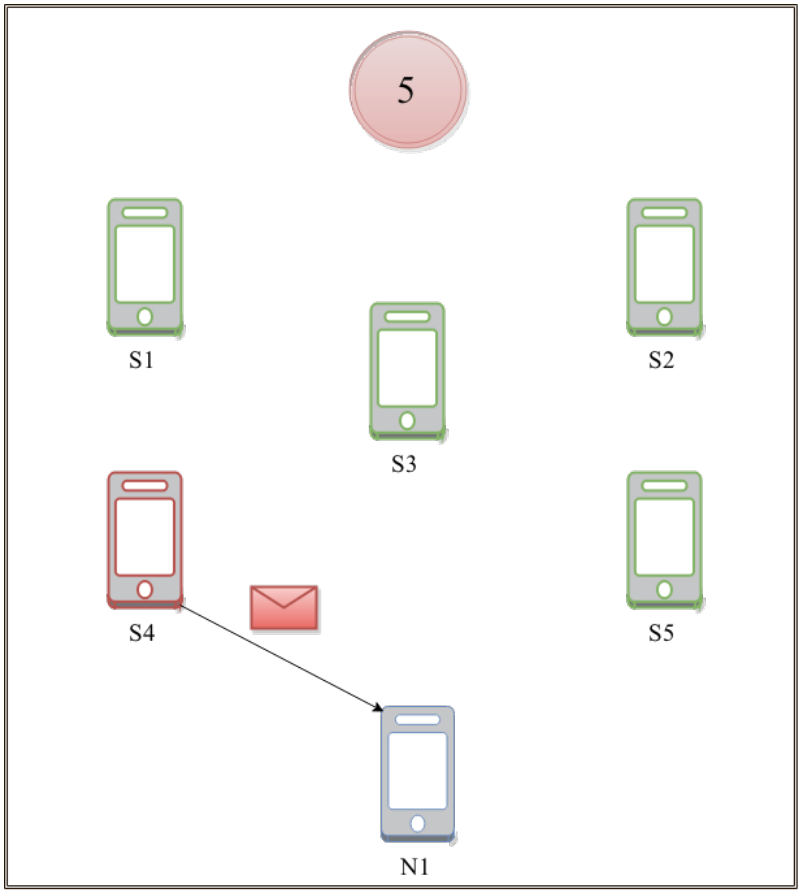


Figure 11. $S4$ send a notification to $N1$

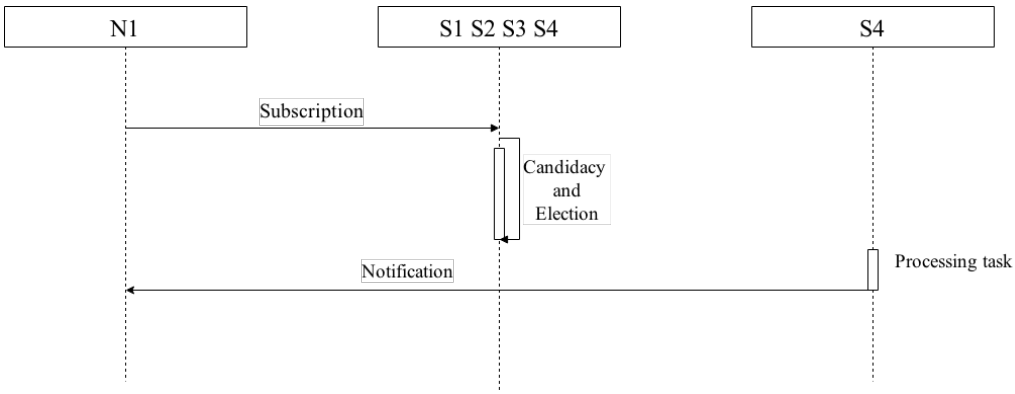


Figure 12. Sequence Diagram of

3.3.3 Messaging Channel Pattern

There are different messaging channel patterns for exchanging information, the traditional one behind the point-to-point channel, e.g. client/server architecture where the client send a query to the server, the server send back a response. This method is best for static content that does not change rapidly. If the client want to access the data again it needs to send a query and only after getting a response back will know if the data has changed and offer any interest for the client.

A publish/subscribe channel offer advantage to classical key mechanism in an constrained environment. In the standard approach, a request is made to obtain an information variable at diverse time interval. Even if the data available has not change, the query will happen and will return the same value. If there is no restraint on CPU time, energy and bandwidth, the traditional mechanism is more efficient than a publish/subscribe model. In this model, an entity subscribe to a variable .

The subscription is traditionally managed by a main broker. When the data are available from a source, the publisher push the data to the broker entity. The broker is responsible to match the publication with the subscription list. If a publication match 1 to n element(s) in the subscription list, it send a notification to the subscribing entities. The notification is often sensor data with a timestamps. The validity of an information is managed by the publishing entity. A publication can be valid and present in the network for a specific amount of time. When a subscriber is not interested by a variable anymore, it can unsubscribe by sending a request to the broker.

In mobile smart devices scenario, for a small amount of data produce by a few number of things, the query method is better. It has less overhead contrary to a publish/subscribe system. But for a system with a big numbers of things, more data are generated, a publish/subscribe system offers benefits, without constantly sending query, it use less resource. The overhead can be dismiss because of the said benefits.

In classic system using the publish/subscribe message exchange paradigm a subscription is a message requesting data respecting conditions. A subscriber (node) is the entity emitting the subscription. The broker is the entity managing the subscription list and the delivery mechanism for notifying the subscriber nodes of the request result.

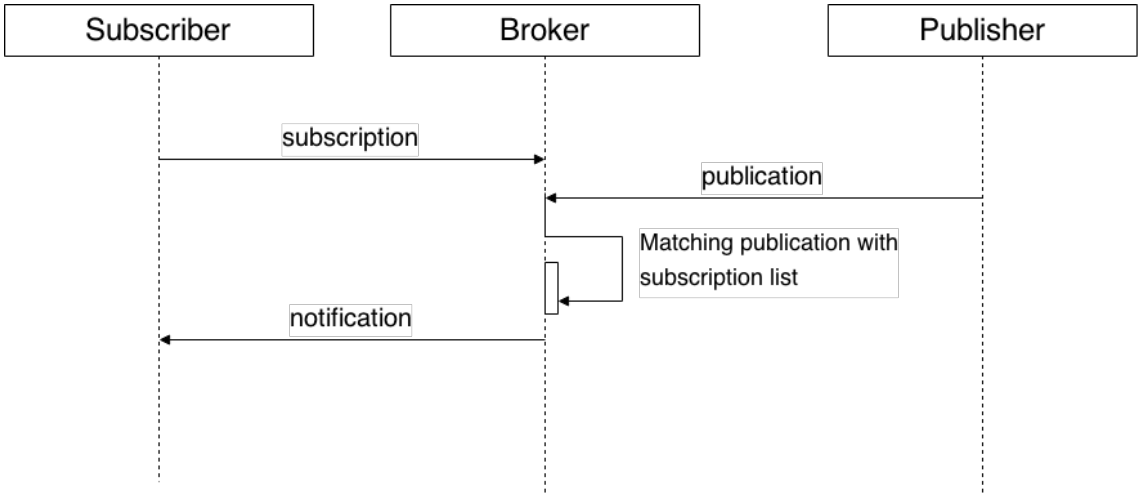


Figure 13. Publish/Subscribe Sequence Diagram

In the standard model the broker has a method that can be use by the subscriber to unsubscribe a previous subscription as shown by figure 14. When the broker receive an unsubscribe request, it will remove the subscription from the list. In the traditional paradigm of client/server, the broker is the server and the clients are the publisher and the subscriber. A broker is able to serve multiple client. We use a timeout on the validity of the subscription to reduce the communication of the entities part of the system.

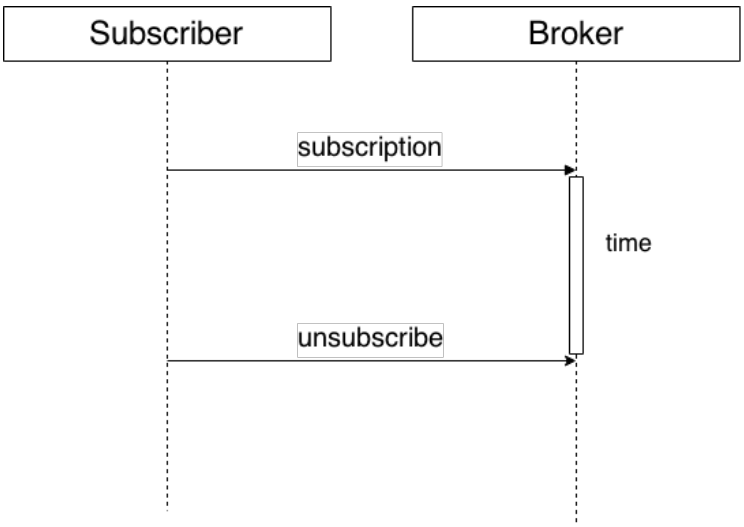


Figure 14. Unsubscribe Sequence Diagram

We decided to fusion the broker and the publisher in the same entity : the smart head. The node subscriber send within the subscription the specific condition to trigger a notification

message. The subscription contains 6 information for a crowdsensing scenario :

- Expiration time
- Service
- Time interval
- Operation
- Accuracy (range)
- Condition for operation (optional)

The expiration time is generated at the creation the subscription. The peer generator use the date and time at the moment of the creation and add the validity time to this date. The result is a future expiration time. The type of task (service), e.g. crowdsensing application it correspond to the type of sensor. It is the service that every peers share. The time interval is use by the smart head to schedule the task. The operations are computation process to execute. The condition object is an optional parameter to the operation, e.g. in a crowdsensing scenario, it is a value to include in the operation process superior five means that the operation is superior and the condition is five. The accuracy is a range, it allows the smart head to test the new value obtain by the peer output against a previous one. If the difference between the old and new value is not in the range accuracy, a notification is sent by the smart head to the node requesting information.

3.4 Operation

Our proposed load balancing scheme is responsible for distributing tasks and reducing the overall resource usage among mobile smart devices. The load in the scope of the project is a task that comprises a request for data or processing. The aim is to give one smart device (e.g. smartphone), called the smart head, the task of allocating process according to specific queries (subscriptions). The list of all the queries comprises the subscription list. The load needs to be shared between different smartphones within the peer-to-peer network. Each services, e.g. crowd-sensing scenario the temperature, is a task allocated by the smart head. We define different parameters as input for the load balancing scheme as presented in table 2.

Table 2. Parameters of Load Balancing scheme

Factors	Definitions
X_0	Battery stage : charging, discharging, critical, ...
X_1	Battery usage : percentage of the energy left
X_2	Processor load : activity the cores
X_3	Time delay
X_4	Task running
$X_{...}$	
X_n	

During the election the peers exchange these parameters via messages called candidacy. The candidacy integrates the important information (parameters) that the load balancing algorithm will need. The LoadIoT process running on each peers is able to determine if it has the capability to become the smart head. If multiple peers have the same value for the first parameter X_0, the scheme will compare the next parameter in the list.

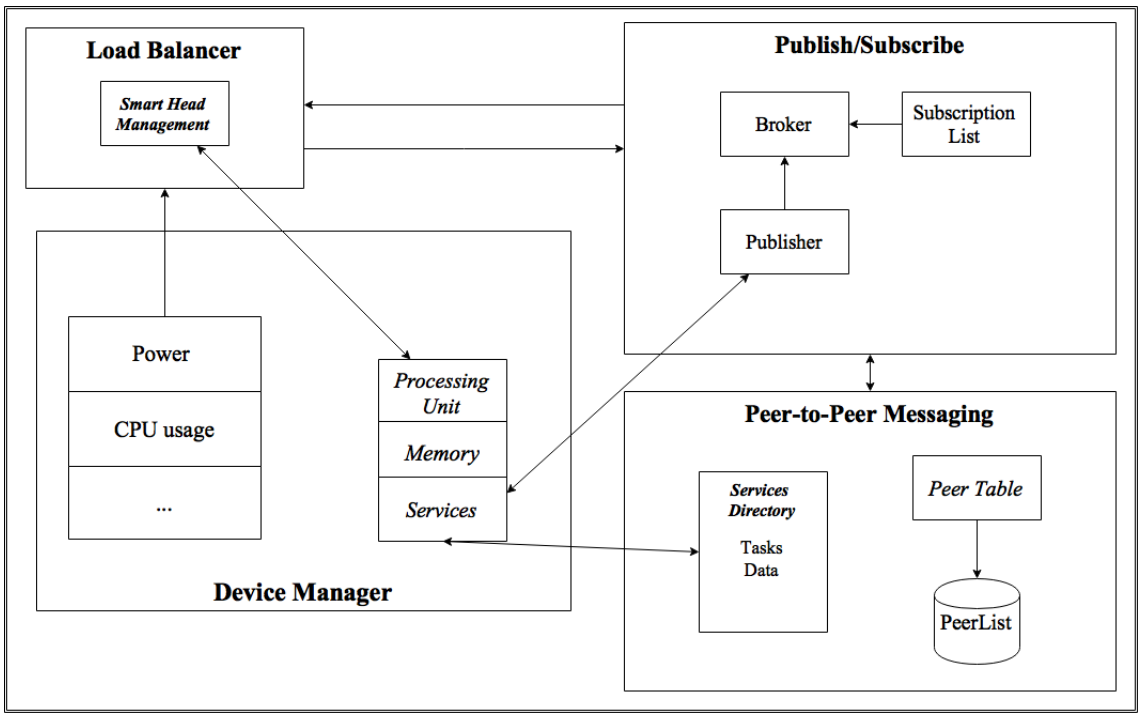


Figure 15. Block Diagram of the LoadIoT scheme

Each block represents one of the core component of the LoadIoT scheme. The smart head management is responsible of the smart head election. The publish/subscribe bock

manage the list of subscription. The peer-to-peer block allows the system to be scalable and to easily add or remove a peer depending of the capacity and resources of the network. The last block is the smart devices operating system, it provides hardware information through an API, e.g. power usage, CPU usage, network connection, ... Also it contains the different tasks that the system knows how to process. The tasks (services) change depending of the implementation of LoadIoT. In a crowdsensing scenario, the task can be the access to sensor information and localisation data.

LoadIoT scheme use one smart head at a time. If a peer has the need for a bigger and more complex task, it is able divide the operation and allocate the tasks to peers available on the network. The appendix 1 present this kind of possible implementation of this scenario.

3.5 Smart Head Election

In the standard model, without any load balancing scheme, when a node try to obtain data from the network, after sending the query, every peers will send a response. This generate mass messages and is highly resource wasteful. The use of a load balancing scheme to allocate tasks to access data is beneficial for the energy consumption of the entire peers network. The smart head is elected using the load balancing algorithm, it takes into account the list of all the peers and the factor list.

The Max function compares the parameters (factor) of two candidacies and determines the smart head using the value attributed to the parameter. The line 2 assigns the value to the factors depending on the application scenario in use. The line 4 sends the candidacy to the peer-to-peer network. The next line waits for the candidacy of the other peer participating in the smart head election. The function of line 8 is loop through each candidacy and determine which is the best peer for handling the load.

3.6 Peer-to-Peer

In distributed computing a well known architecture is peer-to-peer. It offers scalability and decentralization, considering those advantages, we choose a peer-to-peer approach for our design. In this work, peer-to-peer is the way to express that each devices taking part in the processing offer and use different services from the peer network. P2P systems can be really complex and difficult to implement. Nowadays, the most commonly used P2P

Algorithm 1 Smart Head Election

1: Assign value to factor $X_0, ..., X_n$.
2: **for** $i = 0 \rightarrow n$ **do**
    $\mathbf{AssignValue}(X_i)$
3: Send all factor value in candidacy.
4: **for** $i = 0 \rightarrow n$ **do**
    $\mathbf{Send}(\sum_{i=0}^{n} X_i \rightarrow C)$
5: Receive candidacy from all the peers.
6: $\mathbf{loopRec}(C)$
7: Determine smart head.
8: **function** BESTCANDIDACY(ListCandidacy)
9: best = $ListCandidacy_0$
10: $i = 0$
11: **while** $i \neq ListCandidacy_{end}$ **do**
12: best = $\mathbf{Max}(ListCandidacy_i ListCandidacy_{i+1})$
13: $i++$
14: SmartHeadName = $\mathbf{BestCandidacy}(C)$

protocols are the ones for files sharing in the Internet, eg. Bittorrent. It implements a well designed architecture to share information without the need of a centralised server. The protocol is used to connect billion of nodes to each other via the Internet. In our approach, we use efficient way to exchange messages. We minimise the sending of information to reduce the energy consumption.

The peer-to-peer protocol allows the system to be scalable. In mobile smart devices scenarios, a new devices can appear and disappear of the network for plenteous reason. By using a timeout in the smart head management, the system can quickly elect a new smart head if needed. In other cases, a new peer can be added to the system by starting a election or waiting for the next one. In the figure 16, we present a simple scenario. $P1$ is the only peer of the network, no tasking are currently being process. $P2$ wants to become part of the system. It start by sending a join candidacy message. $P1$ send its own candidacy message and the election of the new smart head start. $P1$ is elected. After sometime, $P1$ has to disconnect of the network. Because it is the smart head is has the responsibility to start a new election process. It then send a candidacy quit message. Starting a new election that $P2$ win.

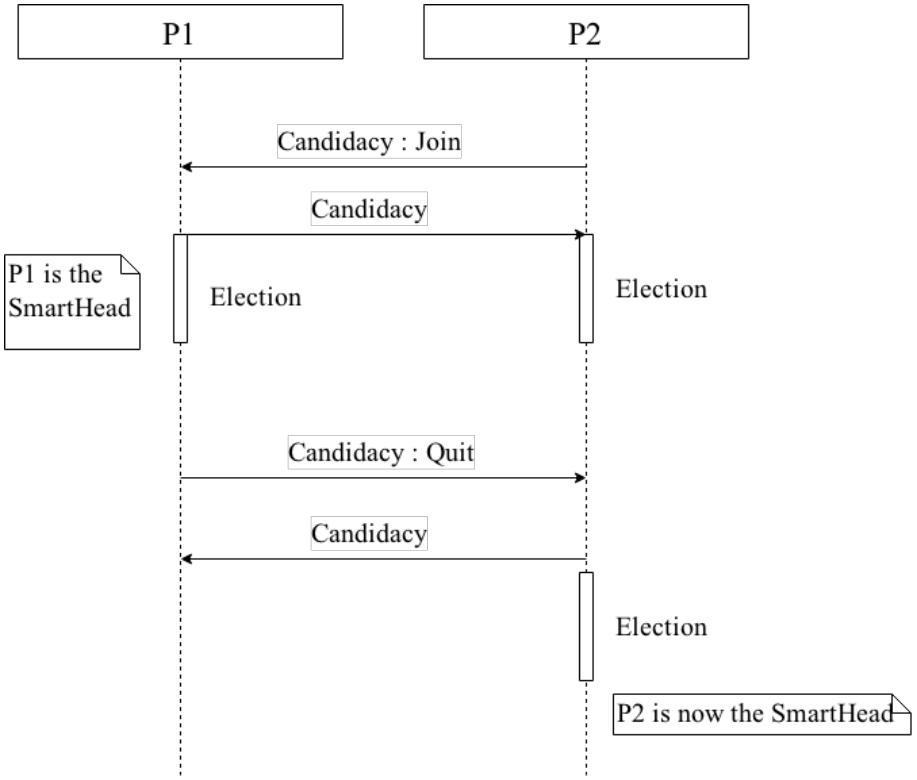


Figure 16. Sequence Diagram Peer Join and Quit

4 Implementation

We named the implementation presented in this thesis of LoadIoT as LoadIoT'App. LoadIoT'App is an Android proof-of-concept application for mobile crowd-sensing. The interface integrate different UI components to show and register information. One of the component is the subscription maker. The maker takes the input parameters for a subscription. The interface provides a graphical view that enables the user to visually see the crowd-sensing data through time.

As explain in the previous chapter, two types of communication system can be use for implementing LoadIoT: Wifi and Bluetooth. We discarded 3G, as shown by the study of Kalic et al. [62], the energy consumption of 3G communication on Android based smart devices is higher than Wifi for the same amount of data. The best communication technologies to use is Wifi, Bluetooth providing to much constrain regarding multicast communication. It provide a simple way to exchange information and we assume that the configuration to the Wifi network has already been done by the user.

The peer-to-peer architecture implementation in LoadIoT'App solution is use multicast technology for exchanging information. It compensate the need of establishing a full stateless P2P stack. Multicast is base on one-to-many communication type, one packet send to an address is received by many entities. The receiver has the responsibility of handling the packets and determine its usability.

The implementation of LoadIoT'App was written in Java for the Android platform. The code use calls to the Android API to obtain information about the battery status. However, because this implementation is Java base, it can run with some modification on other Java Virtual Machine (JVM).

For supporting the message exchange in the P2P distributed system we use IP with multicast UDP. UDP is the best choice because it offers a small footprint compare to TCP. UDP lacks of error detection, retransmission and the other advantage of TCP but provide a simple protocol to send information on a network. Because we use the publish/subscribe model and send result only if a condition is respected, we reduce the emission of packet. UDP has a limit the packet size of 65535 bytes, all of the messages sent by the system are inferior to 150 bytes.

In the next sections we describe the messages sent by the system and the inner working of the scheduling method that start the load balancing algorithm and the election of a new

smart head. The appendix content an other implementation of LoadIoT for distributed image processing.

4.1 Assumption

The environment is defined as a surface on a map. It is devise in different zones. The system has only one instance of the application running per zone. The zone are a geographical area defined by coordinated. The mobile smart devices are part of the same zone if their position is in the zone space as seen in figure 18. We assume that OpenIoT is taking care of space indexing. The scope on this implementation is reduce at one zone.

OpenIoT is a project co-funded from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n^{o} 287305, with the participation of research laboratory and companies from the entire world : University of Zagreb Faculty of Electrical Engineering and Computing (Croatia), Commonwealth Scientific and Industrial Research Organisation (Australia), Across Limits (Malta), SENSAP Microsystems AE (Greece), Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (Germany), The École Polytechnique Fédérale de Lausanne (Switzerland), Athens Information Technology (Greece), The National University of Ireland, Galway (NUI Galway).

The OpenIoT software platform and middleware are developed by the OpenIoT Consortium and available on Github. All the software code is licenced GPLv3, it is a open source platform. It is the only IoT open source project for real life scenario. The OpenIoT Architecture is composed of seven element :

- Sensor Middleware
- Cloud Data Storage
- Scheduler
- Service Delivery and Utility Manager,
- Request Definition
- Request Presentation
- Configuration and Monitoring

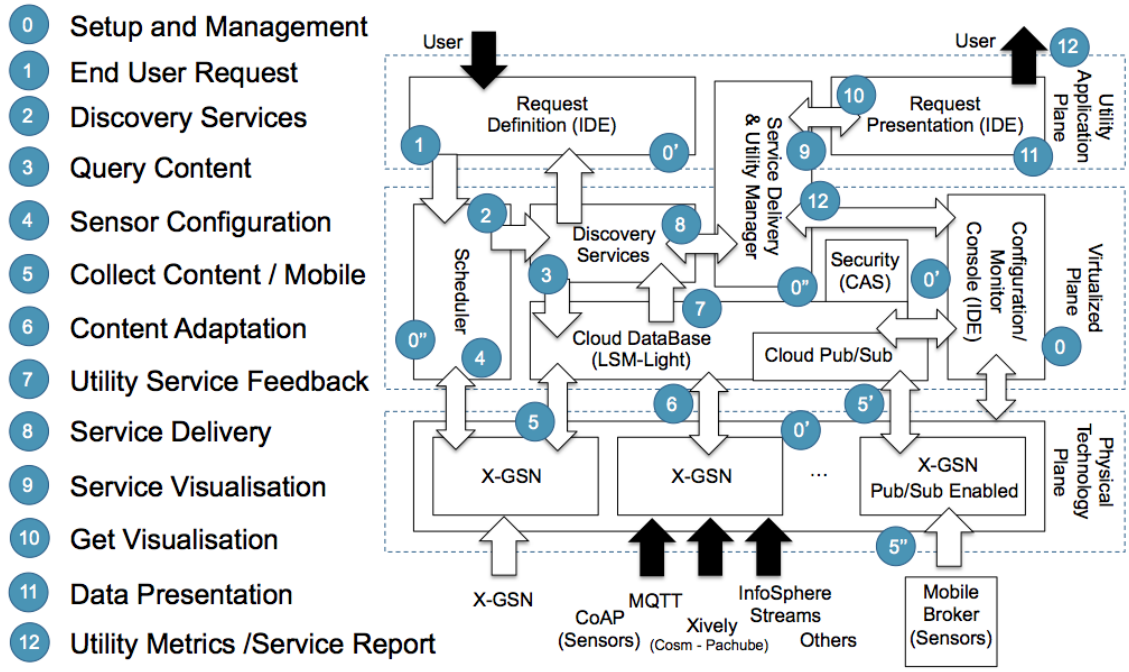


Figure 17. OpenIoT Main Core Components Functional Blocks

The figure 17[8] is the OpenIoT main core components functional blocks and represent the organisation of flow of data. The elements Collect Content / Mobile and Content Adaptation are the interesting components concerning the data acquisition of the *things* (sensors). It use the middleware Extended Global Sensor Networks (X-GSN) for accessing the sensors data. The IoT protocols CoAP[63] , MQTT[18] and other extension using a publish/subscribe messaging pattern are the most use.

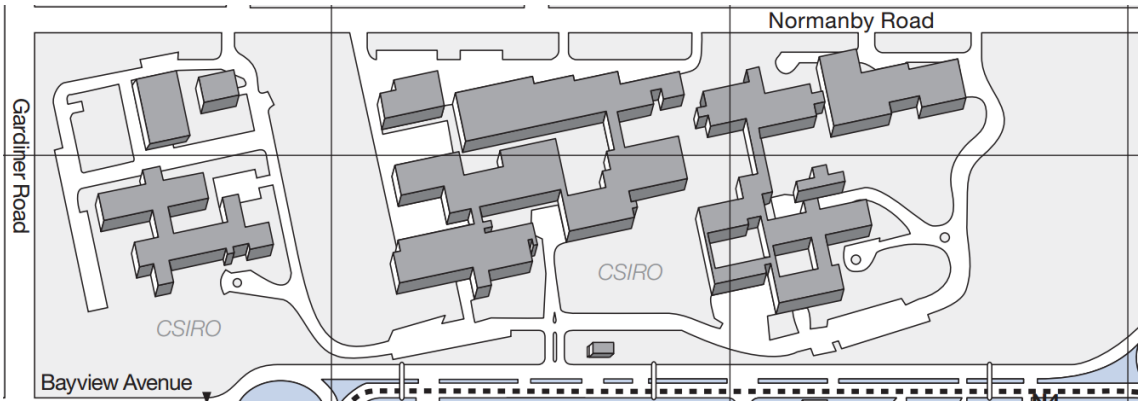


Figure 18. Logical view of zones in an environment

[8]Reproduction allowed by the OpenIoT Consortium

4.2 Messages

Messages are the inner part of every communication protocol. They content information, the payload and the recipient(s) Universally Unique IDentifier (UUID)(s). Because the system use UDP multicast, every peer receive all the traffic. We use recipients UUIDs has to be route the messages. It is needed for special message with an unique recipient. A notification contains the result of a task is always send to one node, this message integrate in the message header the identification information of the receiver.

4.2.1 Smart head Election

During the smart head election, the candidacy of the peers are send on the network. Because the payload is compose of different object, we serialise the information using the JavaScript Object Notation (JSON) format. This format facilitate the exchange of payload in data objects consisting of attribute–value pairs. JSON is compact and the parsing is faster than other techniques of serialisation.

The converted data objects are human readable. The message are encode in ASCII. The first letter of the message is the header. It allow the application to determine the type of the message.

The candidacy message is constructed of :

- peer id : UUID string of 128 bits, generate at each start of the application.
- battery percentage : the remaining battery of the peer.
- CPU load : the state of the peer processor.

e"battery":99.0,"id":"35c2802f-6e0d-42d3-a2a6-0e498292e2d9","load":"0.6225"

Figure 19. Candidacy Message

The figure 19 is an example of candidacy message.

4.2.2 Publish/Subscribe Information

The LoadIoT'App use two type of packets, the subscription (task request) and the notification (task result). The system use the same scheme than previously with JSON formatting. The subscription is send as a JSON payload. The notification is simple and contain only two fields in the payload : the task result (sensor data) and the UUID of the receiver (node). The messages are differentiate using a header at the beginning of the packet (simple ASCII). In each message the first character is a pseudo header. However, the notifications are short and simple information, they do not need to be formatted in JSON. They are also compose of a header, the UUID of the receiver and the data value.

4.3 Proactive

The proactive mode start a new election after a 5 minutes timer, even if no smart head is in place. After the first election at the start of the application, the smart head will start the new election at end of the timer. The newly election smart head will do the same and start a timer.

4.3.1 State Diagram

The proactive mode allow a regular reelection of the Smart Head, this enable the system to detect the status of each peer of the network. It offers a chance to the new connected peer to be a Smart Head.

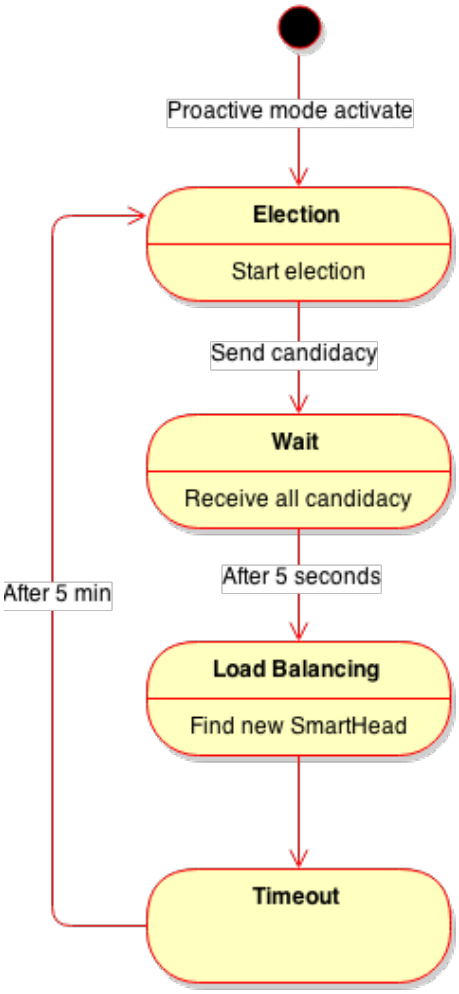


Figure 20. Proactive Stage Diagram

4.4 Reactive

The reactive mode is the default of the application. After receiving the first task request, it start the election of the smart head. A new election happen in two cases : the remaining battery of the Smart Head got to the battery threshold or all the task have been process.

4.4.1 State Diagram

The reactive mode will allow the application to use the peer resource for undetermined amount of time. This duration depend of the task processing. If a peer process multiple tasks the same time and exceed their validity time, a new election can happen because of the battery threshold and stop the processing. The user can change the battery threshold.

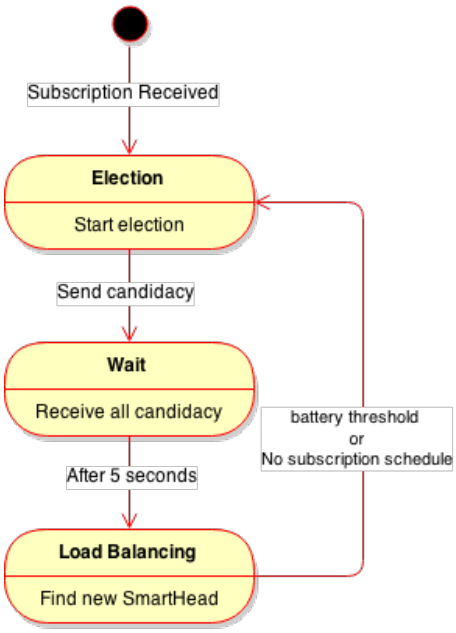


Figure 21. Reactive Stage Diagram

4.5 Interface

The application interface is presented in figure 23, each of the different element are explain in this list :

1. A : Graphic representing the sensor data
2. B : Indicator for the smart head
3. C : Number of peer taking part in the system
4. D : Number of message send and received from the device
5. E : Sensor type
6. F : P2P functionality
7. G : Accuracy of the data wanted
8. H : Reactive or proactive use of the load balancing algorithm
9. I : Subscribe button
10. J : Battery threshold

11. K : Subscription details

12. L : Discover the sensor type in the zone

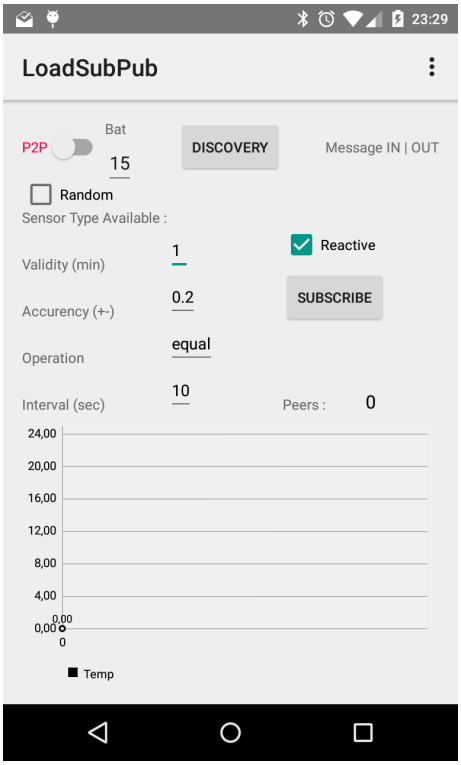


Figure 22. Android Application

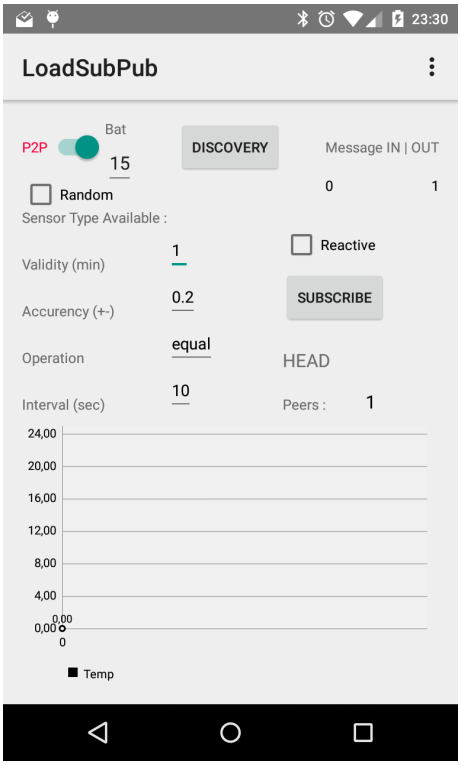


Figure 23. Application after election

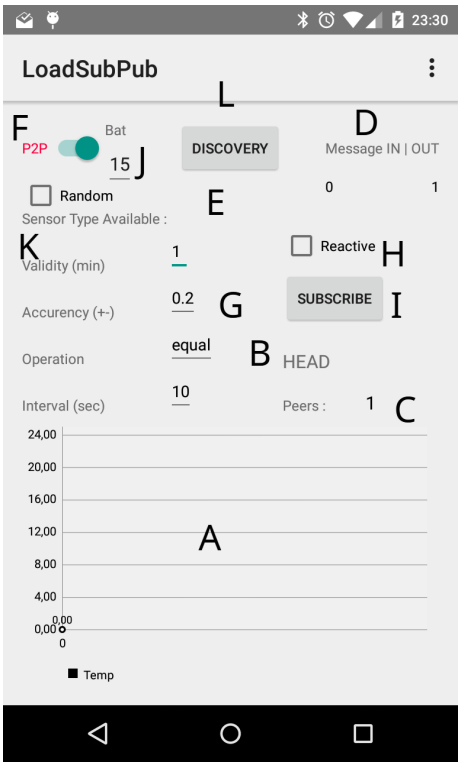


Figure 24. Interface

4.6 Code

The source code of the prototype Android application is available on GitHub[9]. The prototype Android application required Android OS compatible from API 21.

4.7 Development Environment and Tools

We coded the application using the Android Studio IDE, the application platform is Android Lollipop 5.0. The programming language is Java with JDK version 7. The last version of the Android Development tools including the Android SDK were use for creating and debugging the software. To convert Java Object to JSON payload and vice versa, we use Gson a Google Java library.

[9]https://github.com/SheepOnMeth/LoadSubPub

5 Results & Discussion

In our test we use 3 smart devices for testing the system as seen in the table 3. The goal is to compare the number of message generated in different scenario of the multiple configuration system. Also, we aim to validate a model with the output of the experiment setup.

Table 3. Devices information

Constructor	Model	Operating System	Type
Motorola	Xoom	Android 4.4.4	Tablet
LG	Nexus 4	Android 5.1	Smartphone
ASUS	Nexus 7	Android 5.0.2	Tablet

Because the Android API does not give access to the hardware configuration, it is not possible to modify the power emission rate (TX) of the Wifi antenna to limit the transmission power [64]. The load balancing scheme limits the number of message exchanged. This can be applied to other smart devices running different operating system.

The devices are directly connected to a Wifi access point using 802.11g technology. The Nexus 4 smartphone is a node requesting data and both tablets are peers. Also, we include in our experiment,a discovery scheme use by the node for finding the services (sensor type) available on the network.

5.0.1 Scenario

A basic scenario is played three times. A smartphone generates 3 tasks valid for 5 min with an interval of 10 sec. The first two tasks will overlaps during one minute. The third is standalone. The tasks are always producing a output. The tablets are running a sensor simulator outputting new task information every second. In the first scenario, the peers do not use the load balancing scheme. Every peer handle the task and send back the task result (sensor data) to the node. This experiment is use to compare the modes of the load balancing scheme : reactive, proactive and without the scheme.

However, we reproduce the experiment but the battery level of the peers have changed. Base on the load balancing scheme use in the application, we created a simple tool that

simulate the model and the scenario use in the experiment. After validating the model with the experiment result, we simulate bigger scenarios and discuss the advantage of the load balancing scheme.

5.1 Results

During the experiment 1, the Nexus 7 handle the load, because its battery level is superior to the one of the Xoom tablet. It is a constant during the three scenarios.

Table 4. experiment 1

Type	N7 TX packets	Xoom TX packets	Total Packet
Reactive	88	3	91
Proactive	93	4	97
No scheme	90	90	180

However, during the second experiment the smart head change because the Xoom battery has more energy than the Nexus 7.

Table 5. experiment 2

Type	N7 TX packets	Xoom TX packets	Total Packet
Reactive	62	32	94
Proactive	63	33	96
No scheme	90	90	180

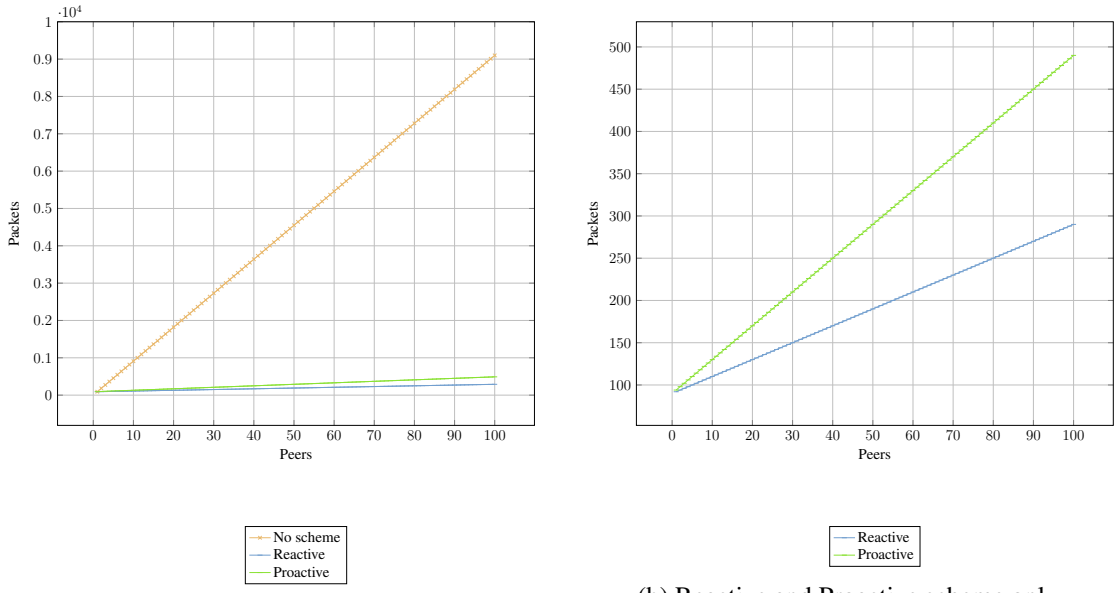
The result provided in the table 5 shows the reactive mode generate more messages than in the previous experiment.

Deriving from our result, we create a simple model to scale the architecture of the system. The python code of the model simulation is available in the annexes of the thesis. We obtain the following result for running the simulation of the experiment 2.

The disparity between the experiment result and the model can be explain by the method by the fact that only valid UDP packet were captured.

Table 6. Experiment 2

Type	Total Packets
Reactive	94
Proactive	98
No scheme	182

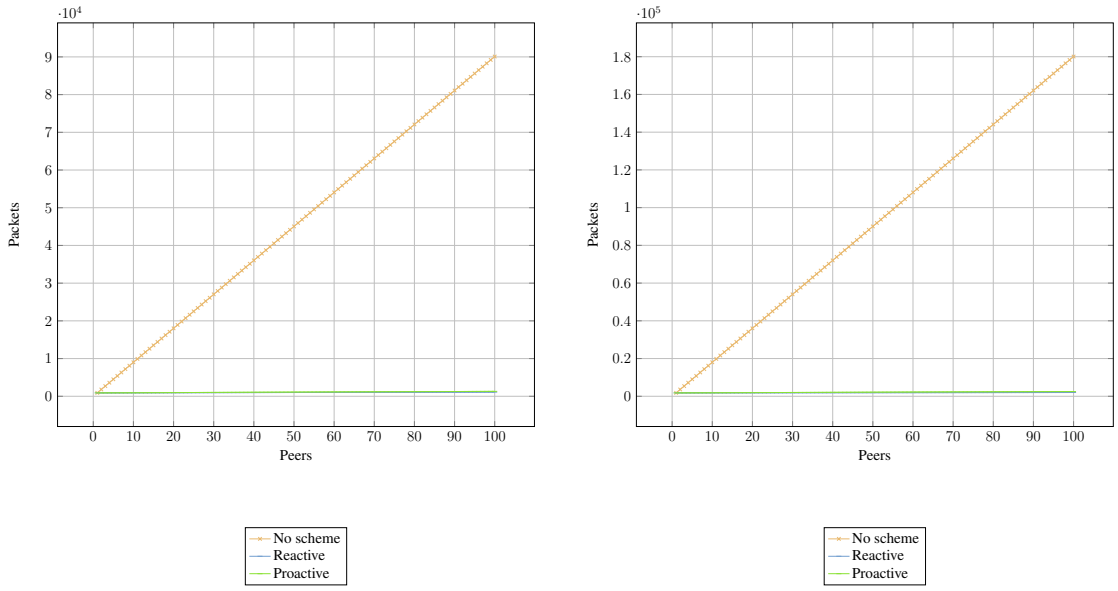


(a) First simulation

(b) Reactive and Proactive scheme only

Figure 25. 3 Tasks 15 min

The result enable us to validate the model and simulate bigger system. The figure 25 present the messages generated for a smart devices cluster of different size, from 1 to 100 peers, on a 15 min duration and 3 tasks, as in the first experiment. In fig 26, we simulate the same experiment as previously but we change the number of tasks and the duration. The load balancing scheme provide a real advantage for a network with important number of peers. It reduce the traffic needed to exchange the information between smart devices.



(a) 30 Tasks 15 min (b) 60 Tasks 30 min

Figure 26. Simulations of different duration and subscription.

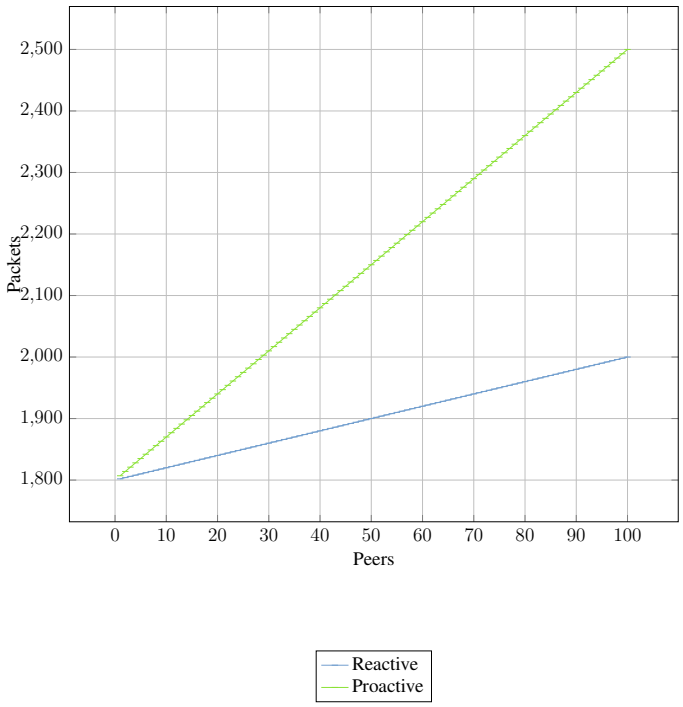


Figure 27. 120 Tasks 60 min

The last figure 27 show that a reactive approach offer better result in term of message economy.

5.2 Summary

The overall result of the simulations shows that less messages are use with the reactive method. The amount of election needed offer a large saving of energy communication.

Table 7. Economy of Messages

Peers	No scheme	Reactive	Difference
10	910	110	800
50	4550	190	4360
100	9100	290	8810

The energy needed for sending a packet on a wireless medium is less that for receiving a message, particularly on mobile smart devices. Android OS use a scheme to deactivate and reactivate the radio hardware rapidly for reducing the energy consumption in a idle state [64]. This energy needed is divided in two parts, the radio antenna on the wireless network chip and the energy use by the CPU and memory to process the tasks [65]. However, the energy for sending a message is not the same for every smartphones and smart devices. From the energy model provided by Balasubramanian et al. [66] we determine that the scheme exchanging less data will use less energy for same size packets.

5.3 Environmental Contribution

This work has the potential to have an enabling effect on sustainable development. Knowing that the problem exist and obtaining basic information about the issue is one step closer to find a way to solve it. LoadIoT implementation could be use for accessing and processing information from air pollution sensors and finding the source of the pollution.

LoadIoT enable mobile far-edge computing in mobile smart devices system, it allow a better management of resources and an saving of energy. A LoadIoT implementation in smart cities and environment monitoring scenarios can be foundation for limiting wastefulness of future system and application.

The LoadIoT implementation are multiple in future edge computing platform. The main effect of this type of system in future smart cities and smart factories will have an impact regarding smart management.

6 Conclusion

Our work has proposed a load balancing scheme for distributing tasks onto cluster of smartphones and multiple scenario. We call this mobile far-edge computing for the IoT World. We presented the theory foundation bounding our research. We then pressed by creating a prof of concept implementation on Android based smartphones. We performed extensive evaluation to validate the feasibility, energy efficiency and the scalability of this work. Our result show that the proposed scheme work well. Particularly, we reduce the number of message exchange in crowd-sensing IoT scenario using Android smart devices.

Mobile smart devices offer a platform for the expansion of the far-edge computing paradigm in smart cities. The application of LoadIoT are multiple has presented in this work, it enables load balancing between devices. Processing data closer to the source can reduce the response time for specific scenarios. LoadIoT has provide a way to diminish the number of message exchanged for data acquisition and so save energy on smart mobile devices.

REFERENCES

[1] Gartner, Inc. Gartner says smart cities will use 1.1 billion connected things in 2015, 2015. [Online; accessed 11-May-2015].

[2] Libelium. Libelium smart world infographic – sensors for smart cities, internet of things and beyond, 2013. [Online; accessed 11-May-2015].

[3] Claudio Cobelli, Eric Renard, and Boris Kovatchev. The artificial pancreas: a digital-age treatment for diabetes. *The Lancet Diabetes & Endocrinology*, 2(9):679 – 681, 2014.

[4] E.I. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic. Edge mining the internet of things. *Sensors Journal, IEEE*, 13(10):3816–3825, Oct 2013.

[5] Chun-Wei Tsai, Chin-Feng Lai, Ming-Chao Chiang, and L.T. Yang. Data mining for internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):77–97, First 2014.

[6] D. Goldsmith and J. Brusey. The spanish inquisition protocol x2014;model based transmission reduction for wireless sensor networks. In *Sensors, 2010 IEEE*, pages 2043–2048, Nov 2010.

[7] Michael Till Beck Martin Werner Sebastian field Thomas Schimper. Mobile computing edge: A taxonomy. In *The Sixth International Conference on Advances in Future Internet (AFIN 2014)*, 2014.

[8] Pacific Northwest National Laboratory. Edge computing, 2013. [Online; accessed 10-May-2015].

[9] Andrew Froehlich. Iot: Out of the cloud & into the fog, 2014. [Online; accessed 11-May-2015].

[10] Arik Gabbai and Kevin Ashton. Kevin ashton describes the internet of things, 2015. [Online; accessed 11-May-2015].

[11] S. Djahel, R. Doolan, G.-M. Muntean, and J. Murphy. A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches. *Communications Surveys Tutorials, IEEE*, 17(1):125–151, Firstquarter 2015.

[12] K. Sangani. The heat is on. *Engineering Technology*, 9(7):49–51, August 2014.

[13] S. Pandey and V. Tokekar. Prominence of mapreduce in big data processing. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pages 555–560, April 2014.

[14] M. Serrano, H.N.M. Quoc, D. Danh, M. Hauswirth, N. Kefalakis, J. Soldatos, P.P. Jayaraman, and A. Arkady. The stack for service delivery models and interoperability in the internet of things: A practical case with openiot-vdk. *Selected Areas in Communications, IEEE Journal on*, PP(99):1–1, 2015.

[15] Klaus Hartke. Observing Resources in CoAP draft-ietf-core-observe-08. Draft 16, RFC Editor, December 2014.

[16] G.K. Teklemariam, J. Hoebeke, F. Van den Abeele, I. Moerman, and P. Demeester. Simple restful sensor application development model using coap. In *Local Computer Networks Workshops (LCN Workshops), 2014 IEEE 39th Conference on*, pages 552–556, Sept 2014.

[17] M. Kovatsch, M. Lanter, and Z. Shelby. Californium: Scalable cloud services for the internet of things with coap. In *Internet of Things (IOT), 2014 International Conference on the*, pages 1–6, Oct 2014.

[18] U. Hunkeler, Hong Linh Truong, and A. Stanford-Clark. Mqtt-s x2014; a publish/-subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 791–798, Jan 2008.

[19] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali. Comparison of two lightweight protocols for smartphone-based sensing. In *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*, pages 1–6, Nov 2013.

[20] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[21] Hao Qian and D. Andresen. Jade: An efficient energy-aware computation offloading system with heterogeneous network interface bonding for ad-hoc networked mobile devices. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*, pages 1–8, June 2014.

[22] R. Trobec, M. Depolli, K. Skala, and T. Lipic. Energy efficiency in large-scale distributed computing systems. In *Information Communication Technology Electronics*

Microelectronics (MIPRO), 2013 36th International Convention on, pages 253–257, May 2013.

[23] Xuetao Chen, S.M. Hasan, T. Bose, and J.H. Reed. Cross-layer resource allocation for wireless distributed computing networks. In *Radio and Wireless Symposium (RWS), 2010 IEEE*, pages 605–608, Jan 2010.

[24] M. Satyanarayanan. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, Aug 2001.

[25] J. Skodzik, P. Danielis, V. Altmann, J. Rohrbeck, D. Timmermann, Thomas Bahls, and D. Duchow. Dude: A distributed computing system using a decentralized p2p environment. In *Local Computer Networks (LCN), 2011 IEEE 36th Conference on*, pages 1048–1055, Oct 2011.

[26] J. Bourgeois, J.-B. Ernst-Desmulier, and F. Spies. Evaluation of a performance prediction tool for peer-to-peer distributed computing applications. In *Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on*, pages 814–820, Dec 2009.

[27] D. Datla, Xuetao Chen, T. Tsou, S. Raghunandan, S.M. Shajedul Hasan, J.H. Reed, C.B. Dietrich, T. Bose, B. Fette, and J. Kim. Wireless distributed computing: a survey of research challenges. *Communications Magazine, IEEE*, 50(1):144–152, January 2012.

[28] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D.P. Anderson. Cost-benefit analysis of cloud computing versus desktop grids. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12, May 2009.

[29] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, 85(11):2629 – 2639, 2012.

[30] Shumao Ou, Kun Yang, and Jie Zhang. An effective offloading middleware for pervasive services on mobile devices. *Pervasive and Mobile Computing*, 3(4):362 – 385, 2007. Middleware for Pervasive Computing.

[31] M.Y. Arslan, I. Singh, S. Singh, H.V. Madhyastha, K. Sundaresan, and S.V. Krishnamurthy. Cwc: A distributed computing infrastructure using smartphones. *Mobile Computing, IEEE Transactions on*, PP(99):1–1, 2014.

[32] M.D. Kristensen and N.O. Bouvin. Developing cyber foraging applications for portable devices. In *Portable Information Devices, 2008 and the 2008 7th*

IEEE Conference on Polymers and Adhesives in Microelectronics and Photonics. PORTABLE-POLYTRONIC 2008. 2nd IEEE International Interdisciplinary Conference on, pages 1–6, Aug 2008.

[33] M.D. Kristensen. Scavenger: Transparent development of efficient cyber foraging applications. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pages 217–226, March 2010.

[34] Mads Darø Kristensen and Niels Olof Bouvin. Scheduling and development support in the scavenger cyber foraging system. *Pervasive and Mobile Computing*, 6(6):677 – 692, 2010. Special Issue PerCom 2010.

[35] J. Parkkila and J. Porras. Improving battery life and performance of mobile devices with cyber foraging. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, pages 91–95, Sept 2011.

[36] M.D. Kristensen, M.B. Kjaergaard, T. Toftkjaer, S. Bhattacharya, and P. Nurmi. Improving pervasive positioning through three-tier cyber foraging. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 135–140, March 2011.

[37] F. Busching, S. Schildt, and L. Wolf. Droidcluster: Towards smartphone cluster computing – the streets are paved with potential computer clusters. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 114–117, June 2012.

[38] E. Aubry, T. Silverston, A. Lahmadi, and O. Festor. Crowdout: A mobile crowdsourcing service for road safety in digital cities. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*, pages 86–91, March 2014.

[39] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. Fostering participaction in smart cities: a geo-social crowdsensing platform. *Communications Magazine, IEEE*, 51(6):112–119, June 2013.

[40] Xiping Hu, Xitong Li, E.C.-H. Ngai, V.C.M. Leung, and P. Kruchten. Multidimensional context-aware social network architecture for mobile crowdsensing. *Communications Magazine, IEEE*, 52(6):78–87, June 2014.

[41] W. Sherchan, P.P. Jayaraman, S. Krishnaswamy, A. Zaslavsky, S. Loke, and A. Sinha. Using on-the-move mining for mobile crowdsensing. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 115–124, July 2012.

[42] S.K. Madria and A. Mondal. Crowdsourcing: Dynamic data management in mobile p2p networks. In *Mobile Data Management (MDM), 2012 IEEE 13th International Conference on*, pages 364–367, July 2012.

[43] A. Antonic, K. Roankovic, M. Marjanovic, K. Pripuic, and I.P. Zarko. A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pages 107–114, Aug 2014.

[44] Aleksandar Antonic, Vedran Bilas, Martina Marjanovic, Maja Matijasevic, Dinko Oletic, Marko Pavelic, Ivana Podnar Zarko, Kresimir Pripuzic, and Lea Skorin-Kapov. Urban crowd sensing demonstrator: Sense the zagreb air. In *Software, Telecommunications and Computer Networks (SoftCOM), 2014 22nd International Conference on*, pages 423–424, Sept 2014.

[45] L. Skorin-Kapov, K. Pripuzic, M. Marjanovic, A. Antonic, and I.P. Zarko. Energy efficient and quality-driven continuous sensor management for mobile iot applications. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*, pages 397–406, Oct 2014.

[46] Gonzalo Camarillo. Peer-to-Peer (P2P) Architecture:Definition, Taxonomies, Examples, and Applicability. RFC 5694, RFC Editor, September 2009.

[47] Benbest et. al. Bittorrent — Wikipedia, the free encyclopedia, 2005. [Online; accessed 11-March-2015].

[48] Func et. al. Kazaa — Wikipedia, the free encyclopedia, 2004. [Online;accessed 11-March-2015].

[49] Archivist et. al. Gnutella — Wikipedia, the free encyclopedia, 2003. [Online; accessed 11-March-2015].

[50] J. Porras, P. Hiirsalmi, and A. Valtaoja. Peer-to-peer communication approach for a mobile environment. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pages 7 pp.–, Jan 2004.

[51] P. Felber, P. Kropf, E. Schiller, and S. Serbu. Survey on load balancing in peer-to-peer distributed hash tables. *Communications Surveys Tutorials, IEEE*, 16(1):473–492, First 2014.

[52] Ananth Rao, Karthik Lakshminarayanan, Sonesh Surana, Richard Karp, and Ion Stoica. Load balancing in structured p2p systems. 2003.

[53] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica. Load balancing in dynamic structured p2p systems. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2253–2262 vol.4, March 2004.

[54] Zhiyong Xu and Laxmi Bhuyan. Effective load balancing in p2p systems. In *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, pages 81–88, May 2006.

[55] Tyler Steele, Vivek Vishnumurthy, and Paul Francis. A parameter-free load balancing mechanism for p2p networks. In *IPTPS*, page 21, 2008.

[56] J. Ledlie and M. Seltzer. Distributed, secure load balancing with skew, heterogeneity and churn. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1419–1430 vol. 2, March 2005.

[57] ZhiHui Lu, XiaoHong Gao, SiJia Huang, and Yi Huang. Scalable and reliable live streaming service through coordinating cdn and p2p. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 581–588, Dec 2011.

[58] M. Wichtlhuber, J. Ruckert, D. Stingl, M. Schulz, and D. Hausheer. Energy-efficient mobile p2p video streaming. In *Peer-to-Peer Computing (P2P), 2012 IEEE 12th International Conference on*, pages 63–64, Sept 2012.

[59] A. Siljanovski, A. Sehgal, and J. Schonwalder. Service discovery in resource constrained networks using multicast dns. In *Networks and Communications (EuCNC), 2014 European Conference on*, pages 1–5, June 2014.

[60] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[61] A. Shahnaz, T. Nafees, and F. Azam. Domain based analysis of messaging patterns in service-oriented architecture. In *Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on*, pages 1341–1345, Oct 2012.

[62] G. Kalic, I. Bojic, and M. Kusek. Energy consumption in android phones when using wireless communication technologies. In *MIPRO, 2012 Proceedings of the 35th International Convention*, pages 754–759, May 2012.

[63] Klaus Hartke Zach Shelby and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, RFC Editor, June 2014.

[64] Li Sun, R.K. Sheshadri, Wei Zheng, and D. Koutsonikolas. Modeling wifi active power/energy consumption in smartphones. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, pages 41–51, June 2014.

[65] Yu Xiao, Yong Cui, P. Savolainen, M. Siekkinen, An Wang, Liu Yang, A. Yla-Jaaski, and S. Tarkoma. Modeling energy consumption of data transmission over wi-fi. *Mobile Computing, IEEE Transactions on*, 13(8):1760–1773, Aug 2014.

[66] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

List of Figures

List of Tables

Appendix 1. Use case : Image processing

Another use-case of LoadIoT is distributed image processing among crowds to identify specific pattern, e.g. colour dominance or faces. This use case is describe as follow : a user takes a picture with a smartphone. The image is then processed within the P2P network for identifying three dominant colours. The first step of implementation is to describe the different tasks part of the process. The micro-tasks and the macro-tasks compose a pipeline of the entire process. A pipeline example is given in figure A1.1.

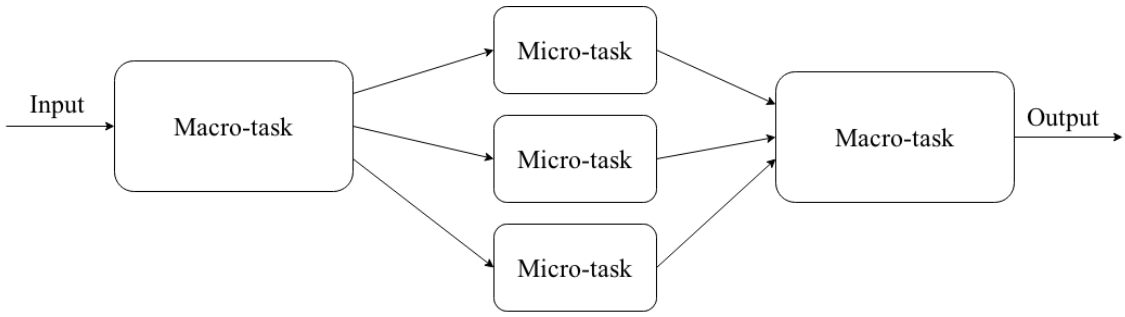


Figure A1.1. Pipeline Example

1.1 Pipeline

The first operation or macro-task is the first load locally processed. It divides the pictures into smaller chunks for processing. This task is handled by the smart head elected by the LoadIoT scheme within the P2P network. The smart head knows which peer of the network are able to analyse the chunks of the picture. The smart head is able to choose the size of each chunk depending on the number of peers available for processing and allocates the micro-tasks to different peers. After the chunks have been analysed by the peers, the result is sent to the smart head. The smart head compiles the result and transfers the data to the user. The table A1.1 present the tasks and their properties. The cost of the tasks is used during the smart head election as one of the parameters. A peer with more processing capabilities, multiple cores and high battery percentage, is able to handle more load than one with one core and small amount of power. The figure A1.2 represent the pipeline for this implantation.

The macro-task divide the image in two and the micro-task is a simple k-means clustering approach to identify the three main colours in the image.

(continues)

Appendix 1. (continued)

Table A1.1. Tasks Information

Task Name	Type	Description	Cost
Divide	Macro-task	Take a picture has a input and divide it in chunks.	High
Analyse	Micro-task	Find the dominant colour in a image chunk.	Medium
Report	Macro-task	Generate report about the colour dominance.	High

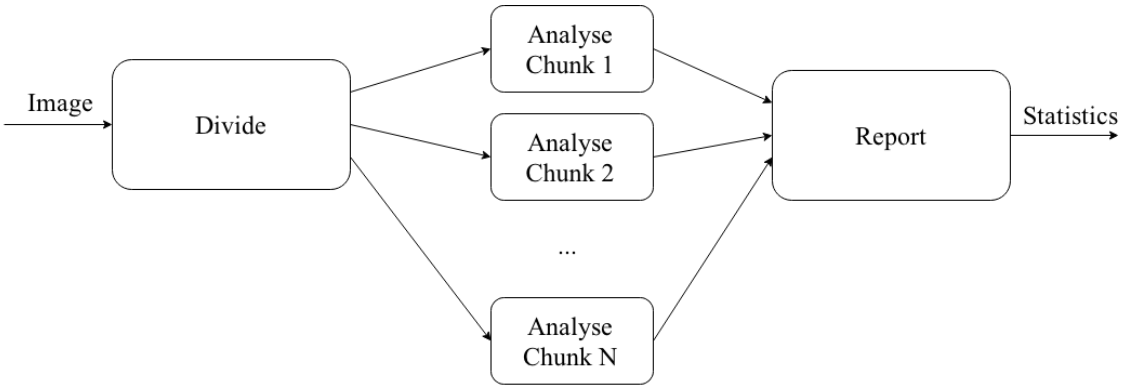


Figure A1.2. Pipeline Implementation

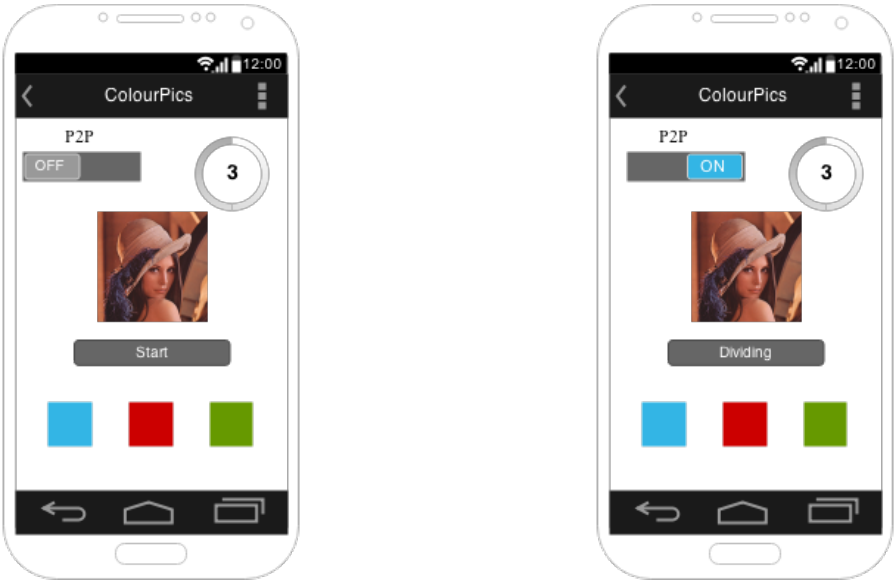
1.2 Smart devices

The smart devices employed in this use case are Android based. The user must be able to take a picture and send it to the network for processing. The figure A1.3 present the android interface of the application with an image loaded. The processing can be done with Python for Android[10] and PIL[11]. A node produce the image and the peers process it. The number of participating peers in the process is on the right up corner of the screen. The slide button below "P2P" enable or disable the availability to process a task. The focus of this application is to demonstrate the feasibility to LoadIoT implementation for distributed image processing algorithm. The details of the actual image processing algorithm are out of the thesis's scope.

[10]https://github.com/kivy/python-for-android
[11]Python Imaging Library

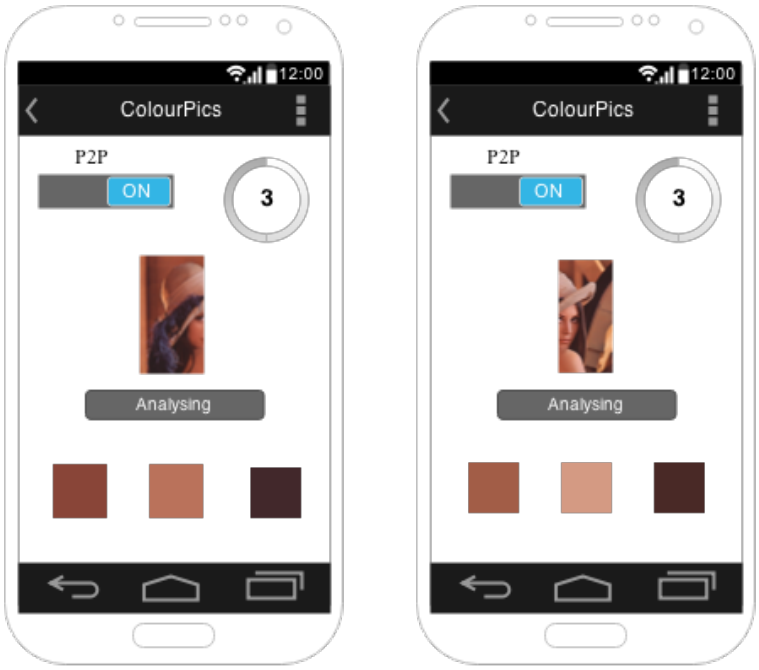
(continues)

Appendix 1. (continued)



(a) Node with picture loaded (b) Peer dividing the image in chunks

Figure A1.3. User interface and macro-task

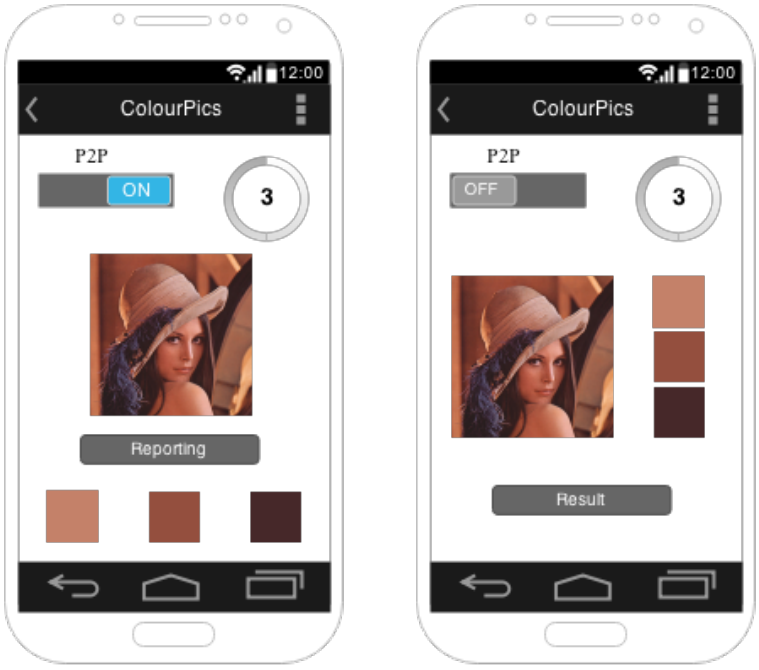


(a) Analysing left side (b) Analysing right side

Figure A1.4. Micro-task process on two different peers.

(continues)

Appendix 1. (continued)



(a) Reporting final data (b) User screen with information

Figure A1.5. Final macro-task and result.

(continues)

Appendix 2. Simulation

Listing 1. Python Model

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# All the subscription have the same specification and will
# trigger a notification

import argparse
import matplotlib.pyplot as plt
import numpy as np

if __name__ == "__main__":

        parser = argparse.ArgumentParser()


        parser.add_argument("-p",dest="peer", help="Number of peer in the simulation")
        parser.add_argument("-s",dest="sub", help="Number of subscription")
        parser.add_argument("-d",dest="duration", help="Duration of the simulation")


        args = parser.parse_args()
        duration= int(args.duration)
        peersmax = int(args.peer)
        sub = int(args.sub)

        output= "Peer_"+str(peersmax)+"_Sub_"+str(sub)+"_Dur_"+str(duration)+".dat"


        print("Modelisation Tool")

        print("Sub "+str(sub))

        #sub generate a notification each time
        # 1 sub valid 5 min generate a average of 6 notifications in one minute.
        n =  6*5*sub;

        #Discovery and response

        print("Base Model")

        # First i represent the discovery mechanism

        base = [i+i*n for i in range(1,peersmax+1)]



        print("Load Balancing")
        #reactive
        # 0 overlapping subscription
        election = 1
        load1 = [i+election*i+n for i in range(1,peersmax+1)]

        #proactive
        #Find the number of election in the entire duration
        # election every 5 min
        election = duration/5

        load2 = [i+election*i+n for i in range(1,peersmax+1)]

        savepacket = list(np.array(base) - np.array(load2))
        print("Packet Diff :"+str(sum(savepacket)))

        data= open(output,"w")
        data.write("peer"+" "+"base"+" "+"reactive"+" "+"proactive"+"\n")
        i=0

        while i < len(base):
                data.write(str(i+1)+" "+str(base[i])+" "+str(load1[i])+" "+str(load2[i])+"\n")
                i+=1
        data.close()

        ymax = max(base[-1],load1[-1],load2[-1])
        plt.plot(base,'bs',load1,'bo', load2,'g^')
        plt.axis([1, peersmax, 0, ymax])
        plt.ylabel('Packets')
        plt.xlabel('Peers')
        plt.show()
```

(continues)