Universidade do Minho
Escola de Engenharia
Departamento de Eletrónica Industrial

Ricardo Manuel Santos Martins

Secure and High Performance Framework for Smart Cities Based on IoT

November 2018

Universidade do Minho
Escola de Engenharia
Departamento de Eletrónica Industrial

Ricardo Manuel Santos Martins

Secure and High Performance Framework for Smart Cities Based on IoT

Master dissertation
Master Degree in Industrial Electronics and Computers Engineering

Dissertation supervised by
Sérgio Lopes

November 2018

ACKNOWLEDGMENTS

In my life, I always considered three very different moments, the Yesterday, the Today and the Tomorrow...

Yesterday, I was a young sweet baby, guided by the experience of two great parents and a unique brother, that taught me much more than any kid deserve. Yesterday, I went to school, I met friends, and I learned with my teachers. But nothing compared with the daily true friends I always had at home.

Yesterday, I met a girl... No, a woman... No, a lady... First time in my life that I truly understood the ″that woman drives me crazy″...

Yesterday, I saw my brother getting married (with such a woman, I must confess) and leaving home... I thought it would be a ″small end″ for me, but instead it was a ″big begin″.

But Yesterday was yesterday, just before Today. Although I regret nothing in my past, it is important to let it go, keeping their best memories forever. From the Yesterday, I made a list of what I will never forget:

- the partnership from by parents and brother;
- the knowledge of my paternal grandmother;
- the family spirit of my maternal grandmother;
- and the engineering vein from my paternal grandfather.

Although they are not all Present here today, I thank them for leading me from heaven.

But Today... Well, Today I thank for my Yesterday...

Today, I am about to end my master's degree, and I want to thank to my supervisor Sérgio Lopes, and to my three colleagues/partners Nobo, Belho and Loira. To them, I thank all the funny and crazy moments we had, and my most sincere wish is to keep in touch with them.

Today I am a grown man (so they say). I had the privilege to meet an honest businessman as Dr. Fernando Moreira, I have a good job at DigitalSign, and I aim to build my own company.

Today, I thank to the great technological knowledge that my brother and father gave me. Thanks to them, I have a big advantage compared to many of my colleagues. Apart from love, family and health, this was from far the best gift I could ever had.

Today, I think of moving out with the sweetest Sassenach in the world. To Mafalda, I thank the fellowship and the gift of dreaming. Although Yesterday I was unable to dream,

Today I dream with her in my life, I dream with our family and pets, trips and old age... I deeply thank you for your heart.

However, the Today is not an infinite gift from my Yesterday. I will have to rely and invest on my Tomorrow to keep having a thankful Today...

So Tomorrow, I give my dedication, effort and gratitude to the Universe, hoping to receive in return strength, health and love to my nearby, making each Today better than Yesterday.

Yesterday is but Today's memory, and Tomorrow is Today's dream.

Khalil Gibran

ABSTRACT

Smart Cities can be defined as a technology based phenomenon capable of positively transforming the urban life style, improving mobility, sustainability, sociability and work conditions, among other aspects. This is accomplished by the intensive use of information captured, processed and made available in a very fast and massive way, through a plethora of digital devices interconnected by highly flexible communication networks which, together, are commonly referred to as IoT (Internet of Things) (Rathore et al., 2016; Hwang, 2015). To properly deploy such a large-scale system, it is desirable first to analyze and test different solutions, which demands for complex simulation systems, since it is not practicable to build prototypes with thousands or millions of ”smart devices” (Medwed, 2016).

Aiming to simulate what will become the city of the future (or cyberspace of the future), fully integrated in the IoT concept, this project proposes the creation of a ”mini laboratory” composed by numerous small devices/platforms with different communication capabilities (RFID, Bluetooth, Wi-Fi, among others). These embedded devices contain some sensors to acquire information from the environment, such as humidity, temperature, light, etc. (Weber, 2010). This information is then sent to a central server, which integrates all information, processes it and provides appropriate services for this emergent and challenging cyberspace. In summary, this IoT Lab aims to create a platform to:

- develop, test and experiment solutions for information security in IoT environments;
- promote the innovation underlying the IoT concept development and, in particular, stimulate security by design paradigm;
- promote a collaborative approach to explore the tradeoff between flexibility (main business requirements) and security/privacy non functional requirements, essential to ensure sustainability and user acceptance, of such systems.

As is already noticeable these days, Internet attacks are becoming recurrent and, with the introduction of IoT, this phenomenon is expected to increase. Indeed, several authors have been addressing the role of Information Security in this new environment, where the management of information naturally plays a central role, being widely accepted that security will be a major source of threats and, therefore, a major challenge for systems development (Zhou et al., 2017; Ijaz et al., 2016; Brambilla et al., 2014; Weber, 2015). Among other aspects, it is important to ensure that the generated and stored information is not changed unexpectedly (integrity) nor it is accessed by those who should not do it (confidentiality) (Bonomi

et al., 2012). In order to ensure these two fundamental properties, the ADSS platform will be used to manage and verify certificates.

The developed architecture is based on a hierarchical structure of small islands with hundreds, thousands, or even millions of devices (sensors and actuators), which cooperate in a relatively restricted environment (the Island), where safety has to be maintained, since these devices communicate and process information (eventually critical) in ”real-time”, with a common goal (or, at least, very close). Each Island contains some type of more powerful computing resource that centrally processes, stores and serves as interface to the outside world - the Cyberspace, or other Islands, in cooperation. In this way, each Island implements some type of service that will be available to the CyberIoT, where several of these Islands can be found. This architecture corresponds closely to the concept of fog computing used in typical IoT environments (Bonomi et al., 2017; Gubbi et al., 2013).

Each Island implements authentication protocols and integrity checkings, over transacted information, because a hostile environment is assumed where several attacking devices can exist, in a very small area. A main requirement in this context, was supporting the high rate of data transfer necessary for overall operation, ensuring the appropriate level of security. At another level, but now with lower performance requirements, each island also has to implement authentication and security assurance protocols. In both scenarios, the ADSS technology will be explored.

RESUMO

As cidades inteligentes podem ser definidas como um fenômeno baseado em tecnologia capaz de transformar positivamente o estilo de vida urbano, melhorando a mobilidade, sustentabilidade, sociabilidade e condições de trabalho, entre outros aspectos. Isto é alcançado pela utilização intensiva de informações capturadas, processadas e disponibilizadas de maneira muito rápida e massiva, através de uma infinidade de dispositivos digitais disponibilizados por redes de comunicação altamente flexíveis que, em conjunto, são frequentemente conhecidas como IoT (Internet of Things) (Rathore et al., 2016; Hwang, 2015). Para implementar adequadamente um sistema assim, de grande escala, é desejável primeiro analisar e testar diferentes soluções, o que exigem sistemas de simulação complexos, uma vez que não é possível construir protótipos com milhares ou milhões de "dispositivos inteligentes" (Medwed, 2016).

Com o objetivo de simular o que será a cidade do futuro (ou o ciberespaço do futuro), totalmente integrado no conceito IoT, este projeto propõe a criação de um "mini laboratório" composto por numerosos pequenos dispositivos/plataformas dotados de diferentes capacidades de comunicação (RFID, Bluetooth, Wi-Fi, entre outros). Esses dispositivos são construídos com alguns sensores para adquirir informações do ambiente, como umidade, temperatura, luz, etc. (Weber, 2010). Esta informação deve então ser enviada para um servidor central, que integra todos os dados, processa-os e fornece serviços de informação apropriados para esse ciberespaço emergente e desafiador. Em resumo, este laboratório IoT pretende criar uma plataforma para:

- desenvolver, testar e experimentar soluções para a segurança da informação em ambientes IoT;
- promover a inovação subjacente ao desenvolvimento do conceito IoT e, em particular, simular o paradigma "security by design";
- promover uma abordagem colaborativa para explorar o equilíbrio entre flexibilidade (principais requisitos de negócios) e requisitos de segurança/privacidade não funcionais, essenciais para garantir a sustentabilidade e aceitação dos utilizadores nestes sistemas.

Como já se nota nos dias de hoje, os ataques na Internet estão a tornar-se recorrentes e, com a introdução da IoT, esse fenômeno prevê-se que aumente. Na verdade, vários autores têm abordado o tema da Segurança da Informação neste novo ambiente, onde a gestão

da informação naturalmente desempenha um papel fundamental, sendo largamente aceite que a segurança será uma fonte importante de ameaças e, portanto, um grande desafio para o desenvolvimento de sistemas (Zhou et al., 2017; Ijaz et al., 2016; Brambilla et al., 2014; Weber, 2015). Entre outros aspectos, é importante garantir que a informação gerada e armazenada não seja alterada de forma inesperada (integridade) nem seja acedida por quem não o deva (confidencialidade) (Bonomi et al., 2012). Para garantir essas duas propriedades fundamentais, a plataforma ADSS será usada para gerir e verificar certificados.

A arquitetura a ser desenvolvida baseia-se numa estrutura hierárquica de pequenas ilhas com centenas, milhares ou até milhões de dispositivos (sensores e atuadores), que cooperam num ambiente relativamente restrito (a Ilha), onde a segurança deve ser mantida, uma vez que esses dispositivos comunicam e processam informações (eventualmente críticas) em ″tempo real″, com um objetivo comum (ou, pelo menos, muito próximo). Cada Ilha tem uma capacidade de computação mais poderoso que centralmente processa, armazena e distribui informação para o mundo exterior - o Ciberespaço ou outras Ilhas, em cooperação. Desta forma, cada Ilha implementa um tipo de serviço que estará disponível para o CyberIoT, onde várias dessas ilhas podem ser encontradas. Esta arquitetura corresponde ao conceito de ″Fog Computing″ usado em ambientes IoT (Bonomi et al., 2017; Gubbi et al., 2013). Cada Ilha deve implementar um tipo de protocolo de autenticação e verificação de integridade sobre as informações transacionadas, porque num ambiente assim é assumido que vários dispositivos podem ser atacados, numa área muito pequena. Um requisito principal, neste contexto, é suportar altas taxas de transferência de dados necessária para o funcionamento global, garantindo o nível adequado de segurança. Noutro nível, mas agora com menores requisitos de desempenho, cada ilha também precisa implementar protocolos de autenticação e segurança. Em ambos os cenários, a tecnologia ADSS será explorada.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 MOTIVATION

The Internet of Things can be seen as a parallel world, in constant evolution, where millions of small devices play an important role in the daily basis. All together and interacting, they form a huge network that attends all the needs, and goes far beyond human mind (Rathore et al., 2016). However, along with many devices, comes a huge amount of data that needs to be collected and processed, being necessary to ensure high performance, as well as information security (Hwang, 2015; Medwed, 2016; Weber, 2010; Zhou et al., 2017). Particularly in the field of smart cities, the security is a core requirement (Ijaz et al., 2016).

Assuming that all these IoT (Internet of Things) devices and heaps of information are coming (in the near future), it is important to ensure that there is a solid infrastructure that guarantees the processing and storage of the generated data. It means not only to ensure a high performance data transmission rate and proper data preparation, but also ″enough″ security, which is commonly neglected in performance offspring. Testing and experimenting solutions in this environment becomes very difficult at first because it seems very unlikely that someone or organization builds such a complex prototype without knowing its feasibility. To overcome this limitation, the development of a framework capable of simulate such a network is one of the motivation of this project.

Although this dissertation is focused on ″Smart Cities″, the same phenomenon will happen in many other areas, such as health and industry.

The development of this project takes place in a computer security laboratory focused on the field of smart cities. Its creation has two phases: (i) selection of all devices/platforms that constitute the desired infrastructure, as well as the respective communication and operation tests; and (ii) definition of requirements for the projects to be developed, including imposition of secure communication protocols, use of digital certificates and the Access Control framework and information sharing, including identity management.

This laboratory is called LabSecIoT, and is powered by DigitalSign, a qualified trust services provider company.

1.2 OBJECTIVES

The deployment of this vast network (described above) to a city, characterizes a ”Smart City”. Since the creation of such network is too complex to experiment a priori in a real city (Brambilla et al., 2014), the goal behind this project is to develop a simulator, where different scenarios can be tested and different technologies can be evaluated. To achieve that, the creation of a laboratory is crucial, where devices, architectures and solutions can be experimented. As mentioned, information security is a primordial requirement, assuming the confidentiality (information is not accessed by those who should not do it) and integrity (information is not changed unexpectedly) of the information must be kept under control (Weber, 2015). To ensure this and following the focus of the organization powering the Lab, digital certificates and PKIs will be used as primary solutions - not forgetting the role of cryptography on protecting those properties.

As a ”Smart City” involves a huge number of devices, it is already known that the cloud cannot handle all the connections and information pre-processing required, suggesting the necessity to introduce intermediate machines (concept of edge computing (Bonomi et al., 2012, 2017)). These machines have the responsibility to transform the low-level information (from devices) into high-level information (identifying adequate features), then able to be uploaded and used in the cloud where, most likely, Machine Learning algorithms will process it. Concerning this topic in particular, this dissertation intends to evaluate the capacity of an Edge concerning the number of devices, guaranteeing the security requirements. This is particularly important to estimate the number of Edges a ”Smart City” needs to have.

The global idea behind this project is to develop an Edge, designed to offer an integrated, solid, secure, trustable and user-friendly decision-making help desk.

Finally, the possibility to simulate different devices is essential, as a ”Smart City” requires that heterogeneity. The simulator must have the capacity to simulate a large variety of object/devices pairs, such as cars, buses, wearables, traffic lights, among innumerous others, with different controllers (ESP, Raspery, to name just two examples) as well as the capacity to simulate thousands or even millions of them. Alongside with this topic, some real devices need to be constructed, like small robots, to collect real data and perform some real tests, mainly aiming to validate simulated devices (Gubbi et al., 2013).

Succinctly, this laboratory aims to develop a ”Smart City” simulator where:

- different architectures can be tested;
- many devices are simulated, and consequently a huge amount of data is produced;
- different approaches about ”how much” security must be assured and provided as datasets, depending on the trust levels required.

1.3 RESEARCH HYPOTHESIS

At the end of this dissertation, it is supposed to exist an answer to the following research question:

> *Considering the "Smart City" architecture under development in this project, can it be used to evaluate and represent typical smart cities operations? Furthermore, considering integrity and confidentiality requirements, using authentication mechanisms typically provided by a PKI, is it possible to scale it to a huge number of devices?*

To answer this research question, it is important to clarify some assumptions, answering the following sub-questions:

> *What is a "Smart City", and what is the impact of IoT devices under its scope?*
>
> *How to determine if a smart city scenario is plausible under certain circumstances?*
>
> *Which are the integrity and confidentiality requirements in the scope of a "Smart City", given related trust levels?*
>
> *In real time and for a given security architecture, how many devices can be handled by a PKI (in particular, authenticated)?*

1.4 METHODOLOGY AND METHODS

The methodology followed by this investigation is based on Design Science research (DSR), usually adopted when the research output is a technology-based solution, emphasizing practical problems. This methodology provides some guidelines that, if followed, ensure that the work has scientific value. Among them, stands out that the study should be associated with a problem related to an organization. This dissertation intend to show that the usage of a "Smart City" simulator and architecture are necessary for a city (the organization) to evolve in the right and profitable way.

Regarding the research question, to evaluate if the simulator plays a fundamental role in simulating a real smart city, its performance will be compared with other similar ones. Furthermore, hopefully some smart cities scenarios will be tested in a real city, allowing, in this case, a direct evaluation of the simulator performance comparing it with the real implementation.

Concerning security problems, particularly integrity and confidentiality, a study about the "best" solution will not be performed. Mainly the performance of a well known procedure based on PKI to authenticate devices will be studied. To measure its performance, the number of devices that can be authenticated per time unit will be used. A priori, it is expected that no more than one thousand per minute can be.

About the definition of ”Smart City” and IoT device, some empirical study will be performed.

Finally, to determine whether a smart city scenario is plausible or not, only a qualitative answer will be given, as this will only result from the observation of the simulator response by stakeholders (in particular City Hall representatives). In a first approach, the operations expected to be evaluated are related to:

- temperature, noise and CO2 services;
- smart parking;
- smart traffic lights;
- smart garbage truck route.

2

STATE OF THE ART

In this chapter we start describing the main technological evolutions that gave rise to the IoT and Smart Cities paradigms, in parallel with the fundamental concepts involved and relevant for the context of this dissertation. In that evolutionary process some challenges and threats are highlighted, in particular concerning Information Security. Overall, the reasoning will support the design and implementation decisions. Since the fundamentals are very broad, not all details are discussed in this chapter, being postponed to subsequent sections, within the discussion concerning implementation, where those details make more sense. The chapter ends with a description of some related projects.

2.1 ICT EVOLUTION

Digital technology is an integral part of the contemporary lifestyle, spreading in an increasingly ubiquitous way over all possible application domains, and following an evolutionary logic, apparently well sustained and without signs of slowing down. At the center of this evolution it is important to recognize two assumptions: the microelectronic components, at the heart of digital systems, will maintain the same increasing rate of miniaturization and speed (a phenomenon commonly referred to as the Moore Law (Schaller, 1997)); and data communication networks will also maintain the same increasing rates, both in terms of speed (phenomena commonly referred to as the Edholm Law (Cherry, 2004))and bandwidth (phenomena commonly referred to as the Nielsen Law (Nielsen, 1998)). Altogether these observations allow us to foresee the possibility of effectively use microtechnology in an integrated network of billions of small devices, for diverse utilizations, such as environment sensing. Notwithstanding its influence, it is important to retain that those are not real laws, but only projections based on observations that serve i) to instigate the innovation capacity (what can we do in 10 years from now with the technology that will be available?), and ii) to put some form of development pressure on industries linked to the digital world (long term strategies considering those performance rates). These two facts are interrelated, and it is sometimes difficult to see when a particular development is part of a well-prepared strategic plan, or is only the result of the need to feed that powerful and increasingly important ecosystem – together, the industries linked to computers, data communications and information, form the Digital Economy with a recognized and significant impact on the World Economy.

Most digital systems integrate, as central components, microprocessors and micro controllers. These components play a fundamental role adding the programmable flexibility to this kind of systems, which allow them to be suitable for virtually anything that we can model in binary representations. To fulfill their tasks they are interlinked with memory components and I/O (Input/Output) devices. The general architecture of these powerful systems was defined in the 50's by John von Newmann, being still valid today. However, several generations of evolution, in particular at the micro architecture level, allowed microprocessors and microcontrollers components to achieve high performance-levels, mainly concerning i) speed, ii) efficiency (number of tasks per watt) and iii) size. Among those evolutions there are a lot of new technical design innovations like parallel execution, multitasking, multi-core, caching, virtual memory and pipelines (Furber, 2017).

Organizing these components to make them work together is part of another study area called Computer Organization. From the 80's Personal Computer (and ignoring previous generations, not so relevant in the context of this work), where each processor has a dedicated memory and I/O devices, to the actual Cloud architectures, where thousands or millions of such modules are aggregated in clusters or through some form of interconnection techniques to promote resource sharing, several innovations, or generations, came up, allowing today to start looking at processing and storing information as a commodity, available to everyone, everywhere (over the Internet) provided by Cloud companies in various flavours and for different purposes (e.g., Infrastructure as a Service - IaaS -, Platform as a Service - PaaS - and Software as a Service - SaaS) (Hennessy and Patterson, 2011).

Figure 1 illustrates this evolution, showing as starting points the two main paralization efforts, or platforms: HTC (High-Throughput Computing), more focused on general business high-flux computing applications (like Internet searches and web services), where the main performance goal is to execute the higher possible number of operations; and HPC (High-Performance Computing), more focused on scientific applications and speed requirements of new and complex algorithms, where the main performance goal is typically expressed in the number of *floating-point operations (flops)* executed by time unit - *GFlops* in the early 1990s, *Pflops* in the first decade of XXI century (Hwang et al., 2013).

Using different technologies and architectures, mainly promoted by notable advances in distributed systems, HTC and HPC converged for big computational and data grids, hiding its inner details and offering large amounts of computational resources in a distributed way. From that convergence point and keeping the same evolution rate (Hwang et al., 2013), i) the introduction of SOA (Service-oriented Architectures) gave rise to the Web 2.0 services and subsequent developments, transforming the Internet in an interactive and integrative

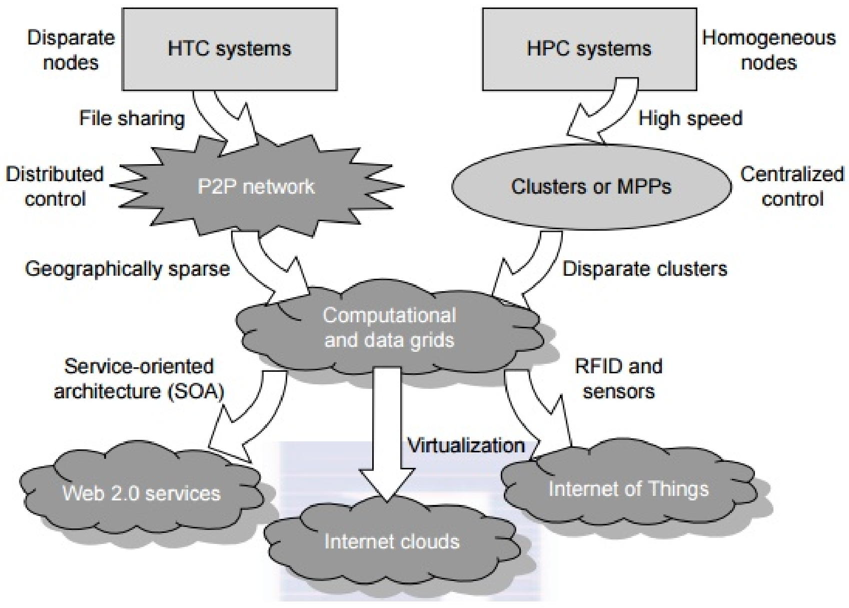


Figure 1: Evolution of Computer Organization platforms on the Internet era (extracted from Hwang et al. (2013)).

programming platform; ii) advances in virtualization techniques and tools, made it possible to see the growth of Internet Clouds and resource sharing, promoting a vision of *computing as a commodity* (Sánchez et al., 2014); and iii) the maturity of RFID (Radio-Frequency Identification), GPS (Global Positioning Systems), among other sensing technologies, promoted the development of the *Internet of Things* (IoT) paradigm, which is particularly relevant for this dissertation and will be addressed in more detail, concerning different aspects, in subsequent sections.

2.1.1 *Virtualization*

Virtualization is also an integral part of the ICT development, allowing to share physical resources among virtual instances, augmenting efficiency (better hardware utilization) and security (promoting isolation between virtual instances themselves, and between them and the physical hardware) (Sahoo et al., 2010). All these benefits came with some drawbacks, like overhead and power consumption. But Cloud providers are constantly investing in research and development, looking for new solutions to minimize those negative impacts, while keeping the advantages (Desai, 2016).

Virtualization is typically implemented through a Virtual Machine Monitor (VMM), or hypervisor, a layer of software that manages the physical resources sharing process, making it available to one or more Virtual Machines (VM). The VMM can implement several strategies (Sahoo et al., 2010):

- Full Virtualization: the VMM runs on top of an OS host, like a user application and creates its own version of virtual resources for all types of components, generic enough to be recognized as a standard machine. This is the most well known case, being used by any user to run, for instance, a different OS on the same machine. The main drawback is the performance since all components are software-based and do not implement optimized versions of real hardware.

- OS-Layer Virtualization: also known as Single Kernel Image, or **Container-based**, this alternative virtualizes the OS itself, allowing VMs to execute on top of the host OS and taking advantaged of all optimizations the host implements. As a result, VMs typically perform better and take less space (since they do not include a copy of an OS), but loosing some isolation capacity and not allowing to run different OSs on top of the same host, directly.

- Hardware-Layer Virtualization (HLV): the VMM runs directly on top of the hardware, implementing the sharing and scheduling mechanisms to allow multiple VMs. Concerning performance and isolation this is the best alternative, since there are no host OS, but each VM will need its own OS, which can be a waste of space, especially if all VMs use the same OS. The performance characteristics makes this the more frequent solution to implement virtualization servers (like in Clouds).

- Para Virtualization: this solution uses a modified version of a host OS, allowing to share the processor and associated components (like in HLV), but using a Full Virtualization approach for the rest of the components (namely I/O and storage). The VMM becomes simpler, performance is higher then with Full Virtualization (but obviously not so good as with HLV) and isolation is better than with HLV. However, each VM needs to have its own copy of an OS.

- Application Virtualization: this is a higher level virtualization, where applications can be run without being installed, since the application environment is offered as a virtual one. Application maintenance (updates) are performed at the virtual level and centrally. The main drawback is the loose of control concerning the application environment, which can have security impact. This is the solution adopted with Cloud SaaS.

- Resource Virtualization: is used to classify software modules that virtualize specific system components (like disks, or network interfaces). This is particularly relevant for the concept of Software-defined Network (Kreutz et al., 2015), which also play a very important role for Cloud Computing.

- Storage Virtualization: this is a particular case of the last type, used to classify systems that implement large data store systems using individual disks, either local or networked, servicing virtual volumes.

From the above description it is obvious that there is a one solution fits all virtualization systems, but only different solutions for different applications. However, it is important to note that implementing a virtual-based infrastructure using a particular technique will impose important limitations if we try to use it for a different purpose then the one first realized. It is also important to retain that virtualization opens new security threat vectors, like guest-to-guest attacks, external modification of the hypervisor and VM Escape (a malicious program running in a guest gets access to the host). Furthermore, when augmenting the density of VMs per physical unit the monitoring of all virtualized infrastructure becomes more complex and difficult to handle.

2.1.2 *Cloud Computing*

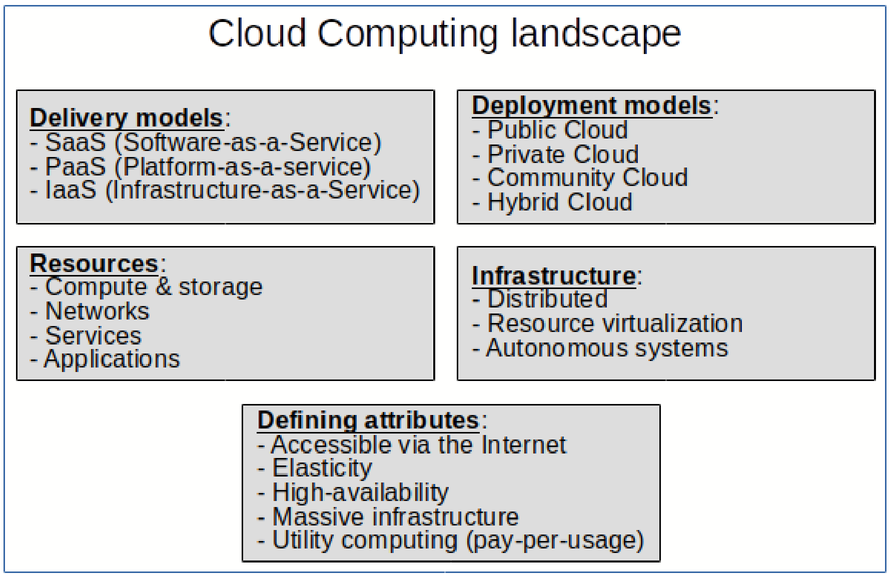
As stated before, the concept of Cluster, as *"a collection of interconnected stand-alone computers which can work together collectively and cooperatively as a single integrated computing resource pool"* (Hwang et al., 2013, p. 66), is central to the emergence of the Cloud Computing paradigm. Clusters have been developed mainly to achieve more powerful computers, providing high-availability and load-balancing the computational load among a given set of available nodes in an (as much as possible) homogeneous way.

Just housing and interconnecting a lot of computer nodes does not make a Cluster, being necessary to install in each one a dedicated software, typically some sort of Operating System extension, or middleware. There are several alternatives, promoting one or more of the properties identified above, and implementing different techniques concerning required functions, such as interface (both human and program), manageability, job scheduling, fault-tolerance and internode communication. From this brief description it is not difficult to deduce that Cluster Computing can be empowered by virtualization systems, making it possible to better manage the underlying hardware. Besides the homogeneity it promotes by an abstraction layer, it makes possible to turn on and off VMs and configure virtual resources (like virtual network components) as required and under demand. Altogether this allows to configure several infrastructure types, from a simple VM to a very complex Cluster or set of Clusters interconnected by proper virtual networks (Hwang et al., 2013; Yang and Guo, 2005).

Within this technology development context, Cloud Computing (CC) emerged as a paradigm to deploy Information Systems, considering all levels from the ITC infrastructure, to the software and service levels. Looking to all the alternatives and solutions provided it is very difficult to find a consensual definition, but one of the most well known and accepted, was provided by NIST (National Institute of Standards and Technology) (Mell et al., 2011, p. 2):

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Nowadays we can recognize a well establish taxonomy concerning the models to deploy and deliver CC, as well as the main resources involved and the more relevant attributes - Figure 2 presents a synthesis of what we can refer as the CC landscape (Marinescu, 2017).


Cloud Computing landscape

Delivery models:
- SaaS (Software-as-a-Service)
- PaaS (Platform-as-a-service)
- IaaS (Infrastructure-as-a-Service)

Deployment models:
- Public Cloud
- Private Cloud
- Community Cloud
- Hybrid Cloud

Resources:
- Compute & storage
- Networks
- Services
- Applications

Infrastructure:
- Distributed
- Resource virtualization
- Autonomous systems

Defining attributes:
- Accessible via the Internet
- Elasticity
- High-availability
- Massive infrastructure
- Utility computing (pay-per-usage)

Figure 2: The landscape of Cloud Computing.

In the last decade several ICT related companies invested largely to leverage this technology. Well known examples are Amazon, Google and Microsoft, which developed their own architecture and Data Centers creating an ICT market used by companies and regular users around the world. Amazon and its AWS (Amazon Web Services) provide mainly IaaS, allowing any user to start using an instance (computer node with limited storage) with no charge and proceed as required, allocating more resources, paying by use, a clear evidence of elasticity. Google presents a slightly different vision, providing storage (Google Drive) and applications (e.g., the very well known Gmail), following a SaaS approach, but also not charging for limited utilization which, by the way, fulfills the requirements of most regular users. Microsoft provide its Azure and Online Services, which follow PaaS and SaaS models respectively, approaching the market in a similar way to other providers. The huge amount of resources provided by those ICT giants (and others) are available mainly as public services, but organizations with more restricted requirements can negotiate the allocation of dedicated resources in private clouds. However, when that is necessary, organizations also have the possibility to deploy their own infrastructures using open-source platforms (of course, requiring a very high skilled technical team). Very well known examples of these platforms are Eucalyptus[1] (giving compatibility with AWS APIs), Openstack[2] and CloudStack[3] (Marinescu, 2017).

Performance Issues

Running a given job in a computer system with a massive number of resources (namely processors) does not imply to run it proportionally faster. In the early days of Computer Science, and in the context of parallelization research, Gene Amdahl argued that if a significant part of the job is sequential it seriously limits the speedup that can be obtained by running it in a parallel system, no matter the number of processors we allocate. Amdahl law (as it is known) - see Equation 1, where s represents the sequential fraction, p the parallel fraction (so $s + p = 1$) and N the number of processors - can be used to model that relation and, for instance, assuming 50% of the code is sequential, it shows the speed-up will always fall between 1.3 (if $N = 2$) and 2 (when $N \to \infty$), making it clear that allocating more then, say, 10 processors (with a speed-up of about 1.8) will not provide a significative gain (Gustafson, 1988).

$$Speedup = \frac{1}{s + \frac{p}{N}} \tag{1}$$

Although originaly used only to parallel computers performance analysis, Amdahl law can be generalized for several other applications where, in general, a fraction of a job can be subject of some sort of optimization through resource allocation control. In Díaz-del Río et al. (2016) the authors discuss such a generalization to analyse the impact of transferring a computer application from a single machine to the cloud, assuming some simplifications: 1) local processors and virtual processors provided by the cloud environment are similar; 2) the cloud environment can provide a huge number of processors (making negligible the time spent processing the parallel part); 3) data transfer in local computers is negligible but it is prevalent in cloud; and 4) the communication bandwidth with the cloud is constant. Under this assumptions they model the execution time in a local machine by Equation 2, where N_c represent the number of cores and E_{local} is the total *unicore* execution time, given

1 https://www.eucalyptus.cloud/index.html
2 Supported by OpenStack Foundation, a very active community (https://www.openstack.org/)
3 Supported by the Apache Software Foundation (https://cloudstack.apache.org/)

by $N_i \times CPI \times T$, where N_i is the number of machine instructions, CPI is the medium number of clock cycles per instruction and T is the clock period.

$$t_{local} = \left[(1-p) + \frac{p}{N_c}\right] \times E_{local} \tag{2}$$

Following a similar rational, the authors model the related execution time in the cloud by Equation 3, considering only the sequential execution time (see above simplification assumptions) and adding a new term to model the transfer time, where D represents the number of bits transferred and BW the bandwidth. This also assumes a worst case scenario where it is not possible to overlap the execution of instructions with data transfer - but it is not easy to assure any optimization at this level since data transfer time is normally out of program control.

$$t_{cloud} = \frac{D}{BW} + (1-p) \times E_{cloud} \tag{3}$$

The speedup obtained when moving to the cloud can now be deduced dividing t_{local} by t_{cloud}. Under the assumption that E_{cloud} is equal to E_{local} (justified by statement 1) above) and denoting that parameter by E_t, we get the expression shown in Equation 4. Replacing E_t by its equivalent expression and rearranging, we get Equation 5.

$$S_t = \frac{t_{local}}{t_{cloud}} = \frac{\frac{p}{N_c} + (1-p)}{\frac{D}{E_t \times BW} + (1-p)} \tag{4}$$

$$S_t = \frac{\frac{p}{N_c} + (1-p)}{\frac{D}{N_i} \times \frac{1}{CPI \times T \times BW} + (1-p)} \tag{5}$$

Calling $D_I = N_i/D$ the *Application Computing Density*, or APD (medium number of instructions per data unit), and $\mu = (CPI \times T \times BW)^{-1}$ the ratio between instruction execution capacity (instructions per second, typically expressed by Mips - Millions of instructions per second) and data transmission capacity (bits per second), we will get the final form given by Equation 6

$$S_t(p, N_c, \mu, D_I) = \frac{\frac{p}{N_c} + (1-p)}{\frac{\mu}{D_I} + (1-p)} \tag{6}$$

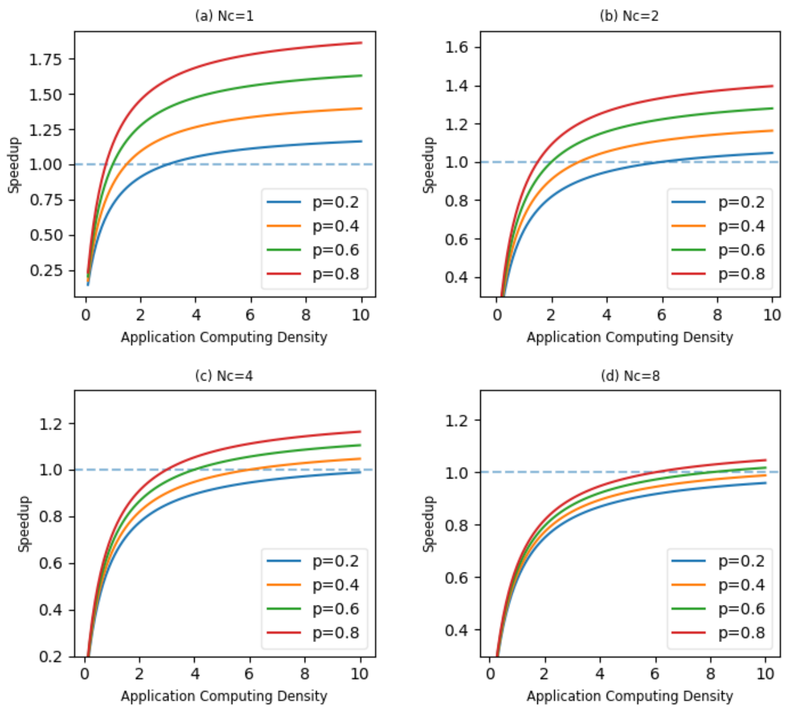
While APD is a metric reflecting the application nature (more data intensive, for values below 1, or more compute intensive, for values above 1), the μ metric is essentially linked to hardware characteristics, with high values indicating a device with high computation capacity, compared to its bandwidth capacity, and low values indicating the opposite (e.g., a device capable of executing 0.1 Mips - $CPI = 10$ and $T = 1 \times 10^{-6}$ - and with 1 Mbps of bandwidth - 1×10^6 - will have $\mu = 0.1$).

This model can be used to analyse how speedup behaves in scenarios that can be anticipated, given some of the parameters. The first scenario (device-centred) aims a low computation capacity node for which $\mu = 0.5$ (e.g., 0.5 Mips for instruction execution capacity and 1 Mbps for bandwidth). Figure 3a shows four graphics, each one considering some variations of parallezation factors (in different colors) and corresponding to the indicated number of cores (N_c). Despite not being expected to have such low powerful nodes with more than one core, it is important to see the effect of the number of cores. With just one core and unless for applications with a very low APD values (Figure 3a(a)), the massive computing resources available in the cloud allow to achieve speedup values greater than 1, specially for higher parallelization factor values (making it advantageous to move to the cloud). Naturally, when the number of cores is higher, the speedup decreases and if we have 16 cores (graphic 3a(d)) only for highly parallel programs it is predictable to achieve a speedup greater than 1. On the other hand, assuming a device with a lower bandwidth ($\mu = 2$, e.g, 1 Mips for instruction execution capacity and 0.5 Mbps of bandwidth), as depicted in Figure 3b, the achievable speedup is very modest, even for applications with higher APD values and the result is even worst when assuming 2 or more cores.

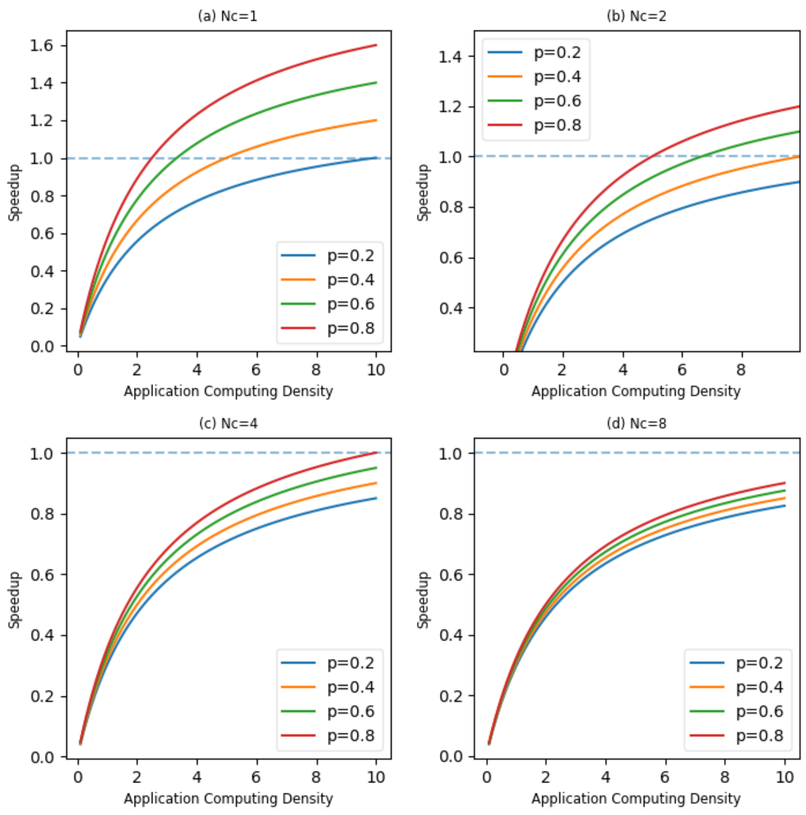
As expected, when considering to transfer an application processing to CC, the available bandwidth is a main issue and can easily become a bottleneck. This scenario is probably what we can expect in an typical IoT environment where millions of devices will share a common gateway channel (or set of). Consequently, this observation rises the question if it is practical to connect sensor devices directly to the cloud in an IoT environment, or if it should exist some middleware – a set of devices (by regions, for example) connects to the middleware, and these connect to the Cloud.

The second scenario aims to analyse how speedup according to the compute capacity bandwidth ratio (μ), for constant APD values (application centred analysis). First we consider an application with high APD value (e.g., $D_I = 2$), which denotes an application tending to be computationally intensive. Figure 4a shows the four graphics for different number of cores, and each one presenting variations of paralelization factor, as before. Again, speedup values grater then 1 are achieved only for low powerful devices but with high bandwidth.

To complete the scenario set we consider an application with a very low APD value (e.g., $D_I = 0.1$), characterizing a data intensive one. The graphics in Figure 4b shows, with no surprise, that to get speedup values greater than 1 we need to be dealing with very low powerful devices, preferably highly parallel solutions and with very large bandwidth values.

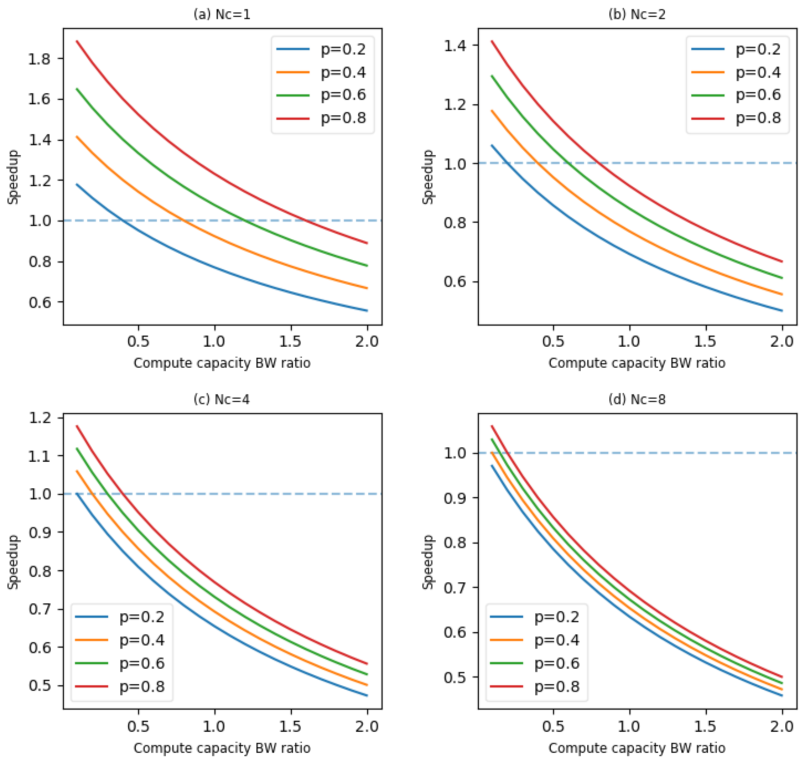


(a) Low computation capacity devices ($\mu = 0.5$).

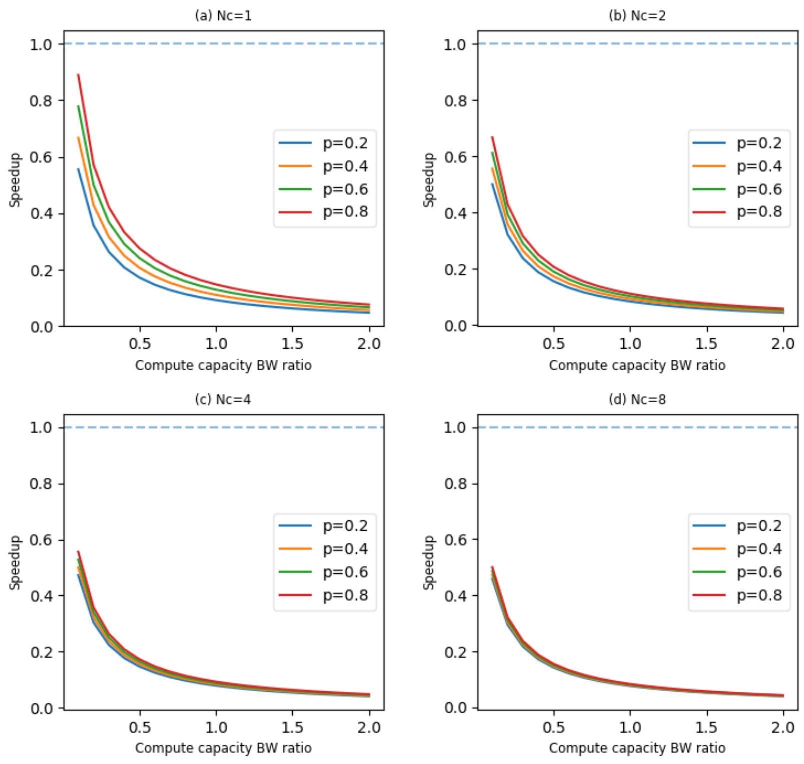


(b) Low bandwidth capacity devices ($\mu = 2$).

Figure 3: Speedup versus D_I.



(a) Low computing intensive applications ($D_I = 2$).



(b) Data intensive applications ($D_I = 0.1$).

Figure 4: Speedup versus μ.

In conclusion, when evaluating the viability of implementing IoT architectures entirely supported on CC for processing purposes, the available bandwidth should be considered as a critical requirement.

Security issues

The Cloud Computing paradigm presents a very important evolution in the way we use ICT nowadays, rising enormous opportunities, but bringing also very relevant security risks. Some of them arise from a highly complex technology stack (as describe in section 2.1.2), while others arise from a completely different environment where users, administrators and machines inter-relate in different ways, some times not fully understanded. Altogether those risks impose serious trust issues, harming the acceptance (Branco and Santos, 2016).

Depending on the delivery model it is possible to identify the key security issues directly linked with CC. With IaaS, VMs and Virtual Networks are in high risk since they are hosted remotely but are under control of the user. This way they are exposed, as any other computer, to malware and viruses, but in an unknown location and environment. The same is true for all other virtual resources, under user control. With PaaS the provider is responsible for infrastructure security, but all other service oriented threats are present, like DOS (Denial of Service) attacks, XML-related attacks, Replay attacks, Dictionary attacks and input validation related attacks (Buffer Overflow). In this case the security is shared among provider and user, being difficult to establish a clear border, frequently. With SaaS most of the responsibility for security is from the cloud and service providers – risk linked to data locality, segregation, access, confidentiality and backups, which are all under control of the provider, but affect mostly the data owner. In addition, there are also risks linked to management aspects, which are not so relevant for this work and will be ignored. Most of these risks can be mitigated by proper selection and deployment of Access Control mechanisms, Secure Software Development practices, key management and security management in general (embracing several well known controls to preserve mainly integrity and confidentiality) (Almorsy et al., 2016; Basu et al., 2018).

2.1.3 *IPV4 to IPv6 Transiction*

Within the context of this dissertation there is another important technological development, already consolidated, deserving special attention: the transition from IPv4 to IPv6. On the Internet all **publicly accessible** machines need an address that uniquely identifies each one. Since its inception in the 80's, IPv4 has been the established standard for defining these addresses, allowing around 4 billion addresses. However, given the rising rate of small computing devices, but whose storage and information processing capacity puts them at the level of computers of a decade ago - Smartphones in particular - the number of available addresses reveals as a limitation. Thus, at the end of the 1990s a new standard, called IPv6 (Deering and Hinden, 2017) was created and has gradually been integrated in the communications infrastructure. The range of IPv6 addresses allows to publicly address thousands of devices per cm2 of Earth surface. In February 2011, IANA (Internet Assigned Numbers Authority, the institute responsible for the assignment of Internet address) announced the exhaustion of the IPv4 addresses. The old Internet is thus giving way to the Internet of Things (IoT), creating a real Cyberspace, characterized by the enormous potential of applications that it allows (Levin and Schmidt, 2014).

This transition is essential to explore the new promises of IoT and all its possible applications, like Smart Cities. But it came also with some drawbacks, mainly related to security issues. In first place because we will have a large period where IPv4 and IPv6 need to be used together and the mechanism implemented to support this simultaneous operation exhibits some vulnerabilities. Besides, the IPv6 operation came with several new mechanisms, for instances, to get an address and to communicate with neighbours, which add new vulnerabilities too (naturally there is a lack of knowledge on how to handle those vulnerabilities) (Zagar and Grgic, 2006). Finally, with IPv4 and mainly within local places (like houses), to avoid the shortage of addresses a special mechanism called Private Network was created. This allows to have just one public IPv4 address, used by the router or external gateway, while all inside machines use private addresses - from a special range of IPv4 addresses, defined for that purpose and, naturally, shared by all machines in private networks. When an internal device sends an Internet request, the router i) exchanges the private address by its own public address, so it can receive an answer later, ii) saves the transaction state and iii) when it receives the answer it does the reverse exchange operation, keeping the network working smoothly and private - this technique is known by NAT (Network Address Translation) and is illustrated by Figure 5.

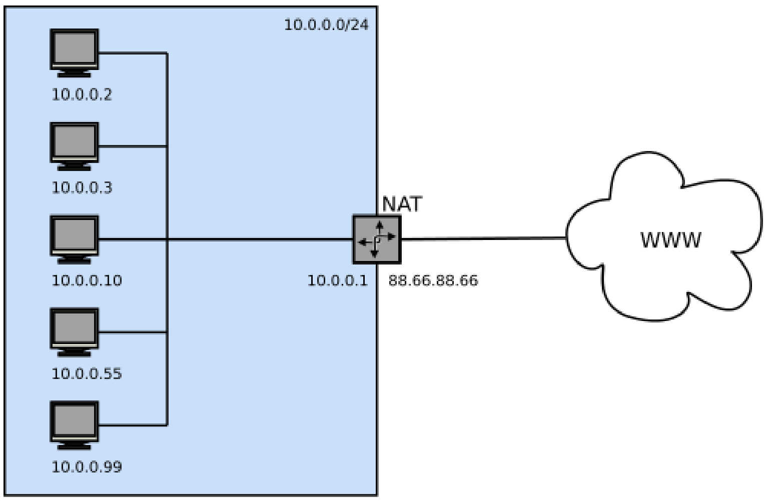


Figure 5: NAT mechanism functioning scheme (extracted from Scharm (2011)).

With IPv6 there is no need to implement such a mechanism and all internal devices may have a public address, making them exposed to the Cyberspace, which is the main purpose of IoT. However, this modification will occour without notice and most users will not be aware of it. In the previous (IPv4) private environment the vulnerabilities of the devices (like TVs, refrigerators, etc.) were not directly exposed, but they will be now (IPv6) and in an unnoticed way (Radhakrishnan et al., 2007).

2.2 THE RISE OF THE INTERNET OF THINGS

The Internet of Things is an emerging paradigm that is rapdly gaining importance in a highly networked world. Together with a pervasive existance of a plethora of *things* or *objects* - devices like RFID tags, sensors, actuators, mobile phones, etc. - linked to several applications, from homes (empowring domotics), to industries and manufacturing (allowing more efficient management and production processes), and not forgetting the enormous impact in several aspects of the way we live and work in cities, the IoT concept appears as a very challenging and promisor endeavor. The impact of this evolution trend is far from being understood, in first place because of the focus shift from an Internet *of people* (the one we are used to) to an Internet *of machines* - in Adat and Gupta (2018) the authors refer some studies indicating the exponential rising tendence of the number of devices in the Internet, which already surpassed the number of humans and is expected to reach the 50 thousand millions mark by 2020, as despicted in Figure 6. This is why we can find references to this technology in most strategic plans in several global and local political, economical and governamental agencies (Atzori et al., 2010).

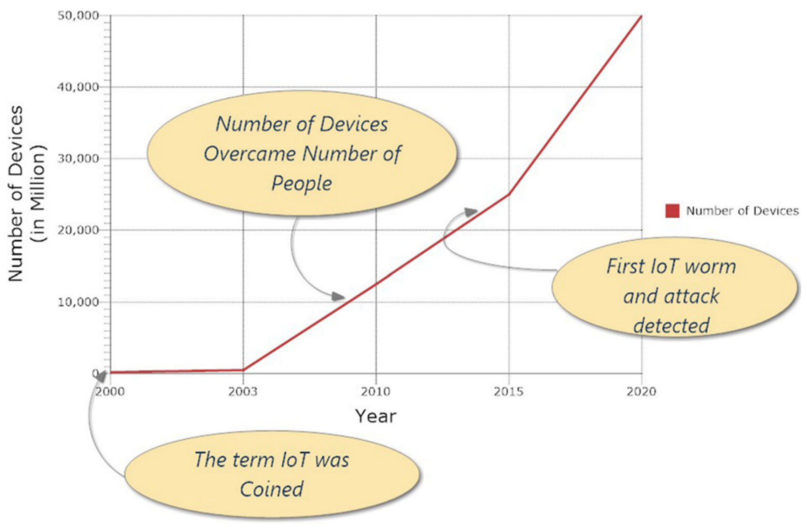


Figure 6: Number of devices linked to the Internet by year (extracted from Adat and Gupta (2018)).

It is common to refer to this paradigm from at least three different perspectives: i) from the elementary technology point of view, considering a set of supporting technologies (both hardware and software) including, besides the devices themselves, machine-to-machine communication devices; ii) from the Internet point of view, focusing on the resulting global network and the required Internet extensions necessary to interconnect all those (*smart*) devices, which includes the adequate protocols; and iii) from the application point of view, focused on the set of applications and services transforming all that technology in new business and market opportunities (Miorandi et al., 2012).

No matter the existence of different perspectives and approaches, IoT is deeply linked to two main characteristics: objects possess a unique identifier (and that is why RFID is almost always refered in this context); and there is frequently a link to some sort of physical phenomenon (e.g., temperature, light, sound, electromagnetic radiation level, etc.) which is supposed to be sensed, co-related and used with intelligent algorithms to allow smarter decisions concerning the applications under consideration. It is not difficult to identify research challenges at all levels and, by its own nature, a broad research area concerned with information security aspects, which are fundamental by trust reasons and because the technology used is typically resource constrained, besides suffering from lack of monitoring operations. Together, these characteristics limit the security controls that can be efficiently applied (Miorandi et al., 2012). Figure 7 illustrates this research vision.

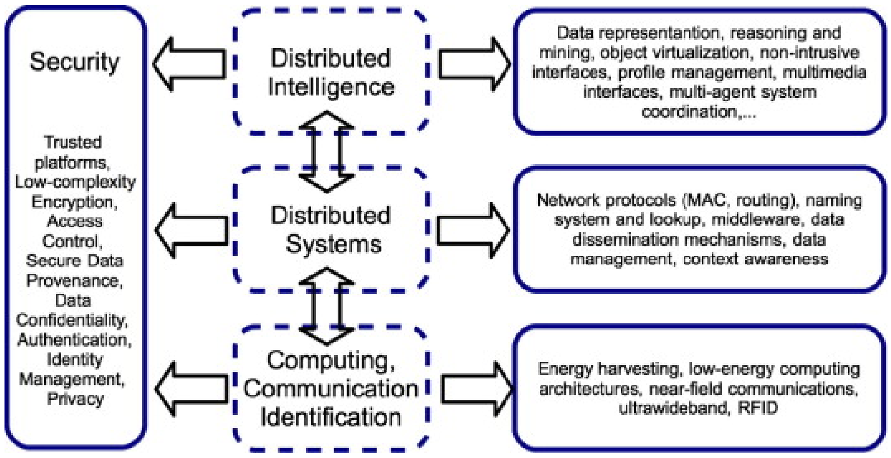

Security
Trusted platforms, Low-complexity Encryption, Access Control, Secure Data Provenance, Data Confidentiality, Authentication, Identity Management, Privacy
Distributed Intelligence
Distributed Systems
Computing, Communication Identification
Data representantion, reasoning and mining, object virtualization, non-intrusive interfaces, profile management, multimedia interfaces, multi-agent system coordination,...
Network protocols (MAC, routing), naming system and lookup, middleware, data dissemination mechanisms, data management, context awareness
Energy harvesting, low-energy computing architectures, near-field communications, ultrawideband, RFID

Figure 7: Research areas relevant for IoT (extracted from Miorandi et al. (2012)).

Besides the sensing function associated with IoT, it is also relevant to address another concept – Cyber-Physical Systems (CPSs). CPSs are mapped to a type of computational systems tightly linked with the physical world, which means that physical data and proper computational models to control and monitor physical processes play a central role in system design (Wan et al., 2010). IoT gracefully complements this function studying solutions to allow CPSs integration in a global networked way, to empower smarter systems. Smart Cities, industrial automation systems, wireless sensor networks, are good examples of applications that can take advantage of that type of systems. Normally the computational elements are embedded into larger artifacts, as they are resource constrained, in size, memory and computational and communication capacity. IoT is clearly a paradigm that aligns interestingly with CPSs, and even more concerning the common support of Cloud-like platforms and Big Data approaches to empower the achievable intelligence (Hwang et al., 2013; Al-Fuqaha et al., 2015).

The organization of such a layered architecture has been also addressed by the Internet of Things World Forum (IoTWF), being important to mention its reference architecture, shown in Figure 8. Besides the physical and connection layers, this architecture also includes important functions at the Edge (or Fog) computing level, namely data sanitization and transformation, making it ready for higher levels where it will be abstracted and integrated to provide proper support to applications and related business processes (Green, 2014). Even so some of these layers can be collapsed with other adjacent layers, the main functions and operations listed are present in most related works.

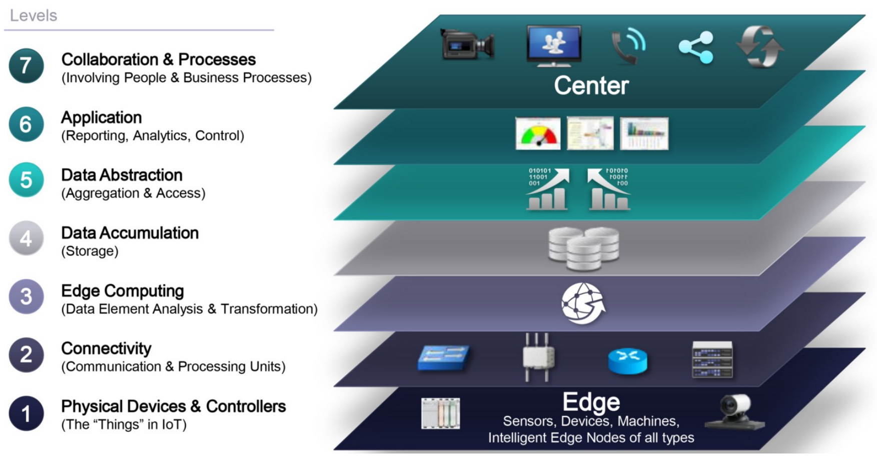

Levels
7 Collaboration & Processes
(Involving People & Business Processes)
6 Application
(Reporting, Analytics, Control)
5 Data Abstraction
(Aggregation & Access)
4 Data Accumulation
(Storage)
3 Edge Computing
(Data Element Analysis & Transformation)
2 Connectivity
(Communication & Processing Units)
1 Physical Devices & Controllers
(The "Things" in IoT)
Center
Edge
Sensors, Devices, Machines,
Intelligent Edge Nodes of all types

Figure 8: Reference IoT Architecture (extracted from Green (2014)).

It becomes more obvious now why Information Security is a fundamental aspect for the success of these smart applications. Considering the critical nature of data handled by CPSs, or the private nature of data provided by IoT devices linked to human activity (when it is the case), enforced by the resource limitation, the embedded operation model, as wel as the amount of computer technology between the physical layer and the business processes, the (potential) vulnerabilities and threats are enormous, both in number and type, with a lot of them resulting from having legacy components working in completely different and unforeseen environments (Kolias et al., 2016).

2.3 INFORMATION SECURITY IN IOT

Information Security (InfoSec, for short) is not a new topic, being possible to find references to it nearly from the era information started to be technologically processed and stored. But InfoSec relevance raised much more recently, with the aforementioned technological advances which allowed huge amounts of information available about almost everything and an enormous access flexibility. This way InfoSec became a fundamental strategic goal for several organizations (Dlamini et al., 2009).

Notwithstanding all the efforts already made in this area, it is appropriate to start by remembering that there is no absolute and definitive security. Instead, there is a perception of security in the face of a threat or a risk. This perception and the associated sense of risk depend heavily on the trust level and fear disposition, and how a human being (in a decisor role) deals with those feelings in a *light* way. Typically, when a new event comes we react and assume defensive behaviours, but we quickly adapt to the environment and, if there are no other stimuli, we relax those defensive behaviours, rising the trust level without plausable reasons. According to Howard and Prince (2011), the human mind is not *prepared* to live with fear permanently. An obvious consequence of this observation is that different agents, in different contexts, will characterize the same security state differently, hampering normalization efforts, but everyone, at some point, will perceive some insecurity.

Even so very influential organizations have already alerted for the security issues that actual Information Systems faces. A good example comes from OECD (2002), a guide published to rise awareness and aiming to promote a security culture through all organization members and targeting all stakeholders. Maybe not as a direct consequence, but aligned with that intention, several efforts from normalization related organizations produced important InfoSec frameworks, like ISO/IEC 27000, NIST SP 800, HIPPA (for health sector), PCI DSS (for financial sector), OCTAVE, MEHARI, CRAMM and CORAS, among others. Thanks to these developments and even though we cannot find a precise definition of InfoSec, it is generally accepted that **it is a management process** through which we try to keep under control **fundamental security properties**, like **Confidentiality**, **Integrity** and **Availability**. That management process is frequently referred as **Risk Management** and it approaches the problem by some possible variations of the following steps: i) identify the resources to be protected; ii) identify the main threats; iii) for each resource assess the risk, if possible determining its value, but at least ordering it in a rank - this includes finding vulnerabilities and determining a value for the resources, which sometimes is almost impossible, specialy for intangible assets; iv) applying a given set of security controls (including policies and technical solutions) to mitigate the security threats, when termined by risk assessment; v) deploy and measure the efficiency of the security controls; and vi) repeat the process periodically trying to improve efficiency, adjusting whatever is necessary (Pfleeger and Pfleeger, 2002; Santos, 2006).

Since the beginning of the IoT concept development, security has been researched actively, discussing threats, attacks, vulnerabilities and security controls, typically approaching the problems by fundamental levels identified in the IoT architecture (see also Figure 7). This means we can find research concerning low-level IoT architectures, addressing sensors/actuators structure and software, inter-communication protocols and Access Control implementation; Internet-like protocols that support routing and distributed networks; and high-level systems where we can find the intelligence associated with the information processing and data storing in large scale (this last level, depending on the focus, is often subdivided in a middleware level - typically implemented in an edge - and a Cloud-based system that implements the intelligent algorithms) (Zhao and Ge, 2013; Sathish Kumar and R. Patel, 2014; Andrea et al., 2015). Furthermore, and more recently, the privacy and trust issues are becoming more relevant, following a global tendency concerning Information Systems in general, appearing in an equal plan with more classical technical and human threats, which also evolve with technology, applications and use requirements, with M2M (Machine-to-Machine) interactions playing a challenge role (Sicari et al., 2015; Granjal et al., 2015; Li et al., 2016; Alaba et al., 2017; Adat and Gupta, 2018; Sha et al., 2018)

According to Adat and Gupta (2018) the first malware targeted to Iot devices was a worm, named Linux.Darlloz, discovered in October 2013 and by the end of the same year it was registered the first attack using IoT-enable devices (Spam campaign in which 29% of bots were recognized as that class). This incident showed that IoT devices could be used to organize Botnets, and the same strategy was used with subsequent attacks. Another class of malware that also caused high impact, was the exploitation of vulnerabilities in car control systems, resulting in the possible remote control, with all nefarious consequences. The same reasoning poses serious conditions to the utilization of Iot in health care systems.

Main IoT Security Requirements

Taking the different characteristics of each layer of the typical IoT architecture previously described (Figure 8), the main threats and attacks identified and the more demanding applications, most of the researchers converge on the identification of basic requirements. In Adat and Gupta (2018) the authors identify:

- at the lowest level, sometimes also referred as perception level (data source), the main concerns are data accuracy and authenticity, which calls for integrity requirements. The lack of resources pose additional difficulties being very difficult to implement the desirable encryption functions. Finally and assuming this level as the source of all data, the availability is sometimes also identified as a requirement, mainly when dealing with critical systems and privacy (for personal devices);

- at the network level and given the heterogeneous nature of the protocols and technologies used, the main concern is confidentiality and availability, in the last case mainly related to critical systems, like in health care related applications. Internet technologies (like Transport Layer Security - TLS, Internet Protocol Security - IPSec and Distributed Denial of Service - DDoS deterrent tools) are considered mature enough to resist to several attacks, but it is a fact that IoT relies also in other technologies not so mature, like IPv6, 6LoWPAN, IEEE 802.15.4, RPL routing protocol etc., created to address low-power requirements and not security requirements; and
- at the higher level, where heavy processing, storing and knowledge extraction is performed, confidentiality and privacy issues appear as main concerns, minimized only by large set of solutions already available, like strong authentication and cryptographic techniques developed for Cloud-based systems. However and at the top-most functional end, which is to provide information services to users, the very different contexts faced (varying from less critical environmental services to high critical health care related services) demand for a careful analysis and identification of the correct security requirements.

Not exactly a requirement, but deeply related with the information security aspects, most of the researchers also refer to trust as a main concern, which is enforced by Yan et al. (2014) defining the properties and objectives of a trust management process aiming to overcome the possible perceptions of uncertainty linked to risks and promoting the necessary acceptance of IoT based systems.

Main IoT Security Solutions

Concerning the mitigation of previousl identified requirements, in Kumar et al. (2016) the authors identify a set of security controls, also organized by the architecture levels, and taken from a survey of research works on the topic. In this same work they propose a security model to improve the applicability of these security controls. From the complete set and concerning the objectives of this work, the following deserve to be highlighted:

- identity manager and service manager for data authentication generated by devices;
- Public Key Infrastructure (PKI) to support certificate based authentication for devices;
- Access Control enforcement at all levels, from physical level, involving sensors, to application layer, involving all software modules;

- data encryption, both in storage and transit;
- intrusion detection through behaviour-based models (sensors and users) and signature-based, mainly at the network layer;
- enforcement of privacy preventing practices, including privacy by design; and
- continuous monitoring, empowered by analytics and predictive functions, based on adaptative security models - this is a very difficult endeavour, especially in highly distributed systems, like IoT and Smart Cities.

Access Control and Authentication are considered critical controls in this environment, by most authors. That is very clear in Sicari et al. (2015), where several published works on IoT security and privacy are surveyed, presenting mainly cryptographic solutions, but trying to overcome the performance limitations of current key-based solutions, not adapted to the scalability, dynamic and lack of resources in IoT scenarios. Some alternative models and protocols are discussed, but none seems to fulfil the requirements, or were never implemented. Trust is also addressed, but like with other security properties, the required metrics are difficult to define. Moreover, in most cases, trust is linked to Access Control and Authentication issues, focused on human-carried or human-related devices. Common reference parameters are honesty, cooperativeness, and community-interest, being evaluated through direct interaction between subjects, in some kind of Social IoT (SIoT), in P2P fashion. Reputation appears also as an important parameter, sometimes linked to more fine grained parameters like experience, knowledge and recommendation - these metrics are even used in a proposed Access Control model denoted by FTBAC - Fuzzy Trust Based Access Control). Some solutions also use location-ware, identity-ware and behaviour information to improve trust characterization. Even so, the lack of more effective metrics and its negotiation protocols, the subjective nature of most of them, and the absence of standards make it very hard to implement in so much heterogeneous environments, like IoT.

2.4 SMART CITIES

A Smart City can be defined as *"a city that monitors and integrates conditions of all of its critical infrastructures, where self-monitoring and self-response are core mechanisms"* (Hall et al., 2000, p. 1). According to IBM, a smart city implies three characteristics: is instrumented, interconnected and intelligent (Harrison et al., 2010). In fact, there are several definitions for this concept and it should be characterized as a massive term that congregates all of them (Nam and Pardo, 2011). The work to theoretically conceptualize it is still in progress, since the practical concept of a smart city is yet emerging, without anyone truly knowing its full potential (Chourabi et al., 2012).

2.4.1 *Architectures and Frameworks*

Even without a clear definition, the results of the research on this topic promotes an architecture based on a set of layers, which follow closely the reference architecture previous discussed in section 2.2 - see Figure 8. In Yin et al. (2015) the authors propose four layers: data acquisition, data vitalization, common data and services, and application domain. The first layer deals with all sort of sensors and the second layer deals with aggregation, filtering and some preprocessing operations. The problems and solutions at these levels are naturally very similar to what we find in IoT systems. The third level aims to organize and provide information services and the fourth level aims to define the requirements of applications relevant in the context of Smart Cities. It is also possible to envisage different domains of application, such as government, citizens, business and environment, and different perspectives: technical infrastructure, application domain, system integration, and data processing. In this work the focus is on the technical infrastructure, with some requirements from the application domain, mainly those related with Information Security, already discussed in section 2.3.

In Jin et al. (2014) the authors describe a similar 4-layer architecture, also based in an IoT infrastructure, as a key technological enabler, and detailing some of technologies normally used at each level. At the network level they discuss several critical aspects, like the sensing paradigm, the addressing scheme (highly affected by the ubiquitous nature of the application), the connectivity model (and the impact of IoT subnets such as RFID and WSN - Wireless Sensors Networks), and the global impact on QoS requirements. As expected, real-time applications (for billion of devices) do not fit well with the characteristics of these architectures because bandwidth limitations are clear, as evidenced. At the Cloud level we find the storage and analytical tools that, besides the obvious effective potential on data processing, some business issues may arise from the cost model. At the data level the authors identify as main issues the interpretation, through proper visualization, the data management, deeply linked to decision support systems, and the security and trust issues. They give a case study - noise mapping - and summarize several initiatives in course around the world.

There are also some open-source solutions to support IoT implementations, used frequently in Smart City applications. In Guth et al. (2016) the authors describe a comparison performed among such platforms, namely: OpenMTC[4], FIWARE[5] and SiteWhere[6]. They include in the discussion the proprietary solution of Amazon, refered as AWS IoT[7]. From the comparison they derive also a generic 4-layer architecture, enforcing the need for an IoT

4 http://www.open-mtc.org/
5 https://www.fiware.org/
6 http://www.sitewhere.org/
7 https://aws.amazon.com/iot/

Integration Middlware level, with a similar function that others include in this intermediate level. Furthermore, they refer other proposed reference architectures concluding that they all can be adjusted to the 4-layer one, assuming some logical aggregations at different levels. This study highlights the prevalence of some protocols/technologies, like MQTT, ZigBee, RESTful, JSON, or XML, as well as the need to research deeply the security dimension of such architectures/applications.

2.4.2 *Related Projects*

In last years and on more structured and organized research and development efforts, some Smart City like applications have been explored (at least around the topic). As an example we can point smart parking management systems. This application has been implemented in different ways and areas, but usually in private and small places. There are more than two dozen parking management strategies and it is proven that when appropriately applied, it can provide significant benefits in economic, social and environmental areas (Litman, 2006). Other example is smart traffic lights, where the vehicles communicate (wirelessly) with traffic lights, making an opening request and/or signaling special conditions. The traffic light network then manages the "best" way to control the traffic flow, as can be seen in Gradinescu et al. (2007) and Maslekar et al. (2011). Notwithstanding the relevance of such solutions, they do not fulfil the requirements of Smart City projects since they are not integrated in broader and powerful Information Systems, capable of augmenting the intelligent capacity embracing all the urban space - they are vertical solutions that can be part of a Smart City.

In such a complex environment it is desirable to test and verify different and alternative solutions before committing to project deployments. In this endeavor, some Smart City related simulators were described, but only a few have been implemented. An example described in Barba et al. (2012) shows a framework that simulates a smart traffic lights network. Another one, where a vision of a smart city for exploring "*structural criticalities, system vulnerabilities, restoration algorithms and overall control of the power system*" (Lugaric et al., 2010, p. 7) was implemented. Some more examples can be found, but none of them address a core requirement: the possibility to deal with virtual/external devices and events, where different security models can be tested. The simulator developed in this project addresses this limitation, which is believed to be an important idea to keep in a smart city simulator.

2.5 SUMMARY

In recent years there has been an unprecedented evolution in the ICT area, promoting the Internet access from everywhere, at every time, and using a panoply of increasingly powerful devices. From this development emerged the concept of IoT which, behind all functional potential, also brings new and challenging security risks, naturally depending on the application.

From another perspective, the actual urban growing phenomenon is imposing several challenges concerning mobility, quality of life (in all its dimensions) and sustainability. Putting it all together, IoT applied to urban environments is promoting a new paradigm to address those challenges, denoted by Smart City. Several architecture, models and frameworks have already been proposed and, some of them, at least partially, deployed. However, some important problems still require a lot of research effort, namely security issues related with Access Control, Identity Management and Trust Management. The work produced in this dissertation aims to address some of those gaps, namely:

- overcome bandwidth limitations;
- ensure "enough" security and privacy both for the devices and the architecture/information;
- grant real-time feedback from the system, even considering millions of devices communicating;
- and ensure that a smart city is not a lot of vertical solutions, but instead an horizontal and integrated one.

3 THE PROBLEM AND ITS CHALLENGES

The problems faced within the Smart City scope are several, being impossible to cover them all. This dissertation tries to approach the most evident ones, like IoT device definition/prototype, communication and applications protocols, information security, store and process device information and plausible architectures for the scenario. To understand these topics, the next sections present definitions and tries to show available options.

3.1 IOT DEVICE

IoT devices are nonstandard computer devices, equipped with sensors and/or actuators, that connect wirelessly to a network, being able to receive/transmit real information. Within the Smart City scope, as thousands or millions can be expected, they have to be cheap, small, georeferenced, have a low power consumption and communicate through lightweight protocols. Ideally, the simpler IoT devices could be seen as disposable.

Being disposable or not, these devices are "dropped" somewhere, and no one else will check their operation. So, one of the challenges for IoT devices is the maintenance, which is easily solved by tracking each one through the data they send. So, some of the features that a Smart City Control Station should implement regarding each device would be: (i) register the relevance/correctness of the generated information; (ii) change the periodicity of communication; (iii) remote configuration; and (iv) register information security and privacy level. Although the tracking of these devices is achievable for a small number of devises, for millions of them it becomes a real challenge.

3.2 COMMUNICATION PROTOCOLS

Although Communication Protocol is a very wide topic, this section refers only part of it. Considering the TPC/IP model composed by four layers (application, transport, Internet and link), this section discusses some wireless protocols under layer three (link and Internet). Although there are many options (Bluetooth, RFID, IEEE 802.11, LoRa, among others), not all of them fulfill the requirements of Smart Cities, due to security issues, scope, or flexibility, among other reasons.

Remembering the Smart City scenario, with lots of devices, scattered around the city, the communication protocol should have a wide range, low price, security and low power consumption. Considering these four requirements, this dissertation considers WiFi (IEEE 802.11), GSM (Global System for Mobile communications) and LoRa. Other known ones like bluetooth or ZigBee are not discussed because they are either insecure or expensive (respectively).

About WiFi protocol (contemplates layer one, link-layer), only IEEE 802.11 is considered because a wide network of known hotspots is necessary, which currently are only (economically) viable if using IEEE 802.11. This protocol has the advantage of being very well-known, allowing a good level of security, a high rate of data transmission and a power consumption admissible. Its main disadvantage is the low range. Even for micro-controllers, this technology is well implemented, allowing cheap devices (like ATmega processors) to work over WiFi.

Concerning GSM protocol (layer one and two, link-layer and internet-layer), it has a huge advantage, which is the accessibility. Its (almost) "absolute" range at no additional hardware cost is certainly its strongest point. Although usually it is subject to monthly fees, there are emerging solutions that in the purchase of a data card (at a residual price) offer limited data transfer per month (as an example, Hologram). However, there are two implementation disadvantages: (i) the available modules for micro-controllers are rudimentary (low flexibility), slow initialization and consume a lot of energy; and (ii) no control of the flow of information. With GSM communication, the packets are all forwarded to network operator gateways, which means they have access to every packet. Since security is believed to be a pillar of Smart Cities, this vulnerability potentiates several threats, jeopardizing the confidentiality and integrity of the information. If this solution is adopted, it is strongly recommended to implement encryption to protect those properties.

Finally, LoRa (an emerging protocol that operates at level one, link-layer) is known to have a long range, low power consumption and secure data transmission protocol. Each LoRa network uses its own gateways, which is good because only the owner controls the flow of information, however they are still expensive. The introduction of these protocol has happened quickly, having already available (expensive) modules and libraries for microcontrollers. The greatest advantages of this protocol are the range, about fifty kilometers depending on the radio-server, and the power consumption, which can be up to twenty years lifetime depending on the battery size (these values are only theoretical, divulged by Semtech, LoRa creators). However, this protocol has a huge limitation (still under study), that is the number of clients (messages) that a node (gateway) can handle. According to this theoretical values [1], at 1000 messages per minute (which is less then the expected in a Smart City), approximately 50% of the messages will be lost. At 100 messages per minute, 10% will be lost. Other studies [2] show that a LoRa node can only handle 120 devices sending 20 bytes every 16.7 minutes, and that at full capacity 82% of the messages are lost. Furthermore, they states that this protocol is not scalable. These values can be slightly increased adding more nodes, but will never (in a near future) be enough.

From this comparison, it is reasonable to justify that the final decision lays on GSM or WiFi. Although WiFi seems to be a better solution (better data rate transmission, more security and accessibility and lower power consumption), it can be very difficult to deploy a full network of hotspots, which requires complex logistic for the city management. So, the final solution may be GSM, even if it becomes more expensive.

To summarize the pros and cons of these three protocols, refer to table 1.

	WiFi	GSM	LoRa
Price for IoT (module)	Very low	Average	High
Price for installation	Very high	None	Very high
Range of signal	200m to 4Km	Unlimited	Up to 50Km
Power consumption	Low/average	High	Low
Security	High	Average	Average
Logistical impact	High	None	Average
Performance	High	Average	Low
Limitations	Hard to implement	Monthly fees per IoT	Low number of concurrent devices

Table 1: Brief comparison between plausible communication protocols.

3.3 APPLICATION PROTOCOLS

After analysing the communication protocols, the next step is to evaluate the application protocols (consecutively, the transport protocol). Recalling one of the main goals, to allow millions of low powered devices to communicate wirelessly, it is important to choose a light-weight, fast and secure protocol. From the literature, Hypertext Transfer Protocol - HTTP (HTTPS), Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP) and Advanced Message Queuing Protocol (AMQP) can be pointed out. To discuss these protocols, it is important to understand what is the scope of communication of a protocol. For example, a scenario where a device communicates with a server, the communication channel is dedicated to those two entities, being a device-to-device protocol. Another scope is the device-to-cloud, where the term "cloud" is used as a group of devices.

1 https://www.thethingsnetwork.org/forum/t/universal-lora-wan-gateway-limitations-because-physics/1749
2 https://www.linkedin.com/pulse/how-many-devices-does-one-lorawan-gateway-support-120-jay-wey

So, whenever a device communicates, all the devices in the "cloud" will listen. Finally, there is also the cloud-to-cloud scope, where everything becomes integrated, a whole shared environment. Due to security concerns, a device-to-device scope of communication is more advisable for a Smart City scenario, because it allows the server to "control" each device that communicates with it.

Starting with HTTP (Hypertext Transfer Protocol), it is a very well-known application protocol implemented over TCP/IP. Surely, the biggest advantage that it offers is its accessibility, being well documented and available almost for every device. Concerning security, it works on TLS, which is a recognized protocol for data encryption. As demanded, it is a client-server protocol (device-to-device). However, as it is oriented to "heavy" transactions (such as HTML documents), it fails on the light-weight requirement. On average, a request header size vary from 200 bytes to 2KB [3]. For a device to send a simple string containing about 22 bytes (such as a temperature value and a GPS location, "15º,-8.15472,41.25478"), its overhead can be considered excessive. As for the server, decoding it for each message received is too painful as well. This way, HTTP protocol for Smart City environment may not be a very wise decision.

MQTT (Message Queuing Telemetry Transport) is an application protocol which also works over TCP/IP. Although created in 1999, only recently started receiving attention. The operation of this protocol is based on a publish/subscribe system, where each user can publish a message in a topic, and every client that subscribes to that same topic, will receive the message. So, its scope is device-to-cloud, but it can be configured to work as device-to-device, without loosing its main virtues. Concerning availability, it can be easily implemented in several architectures, and can also work over TLS, so it is a plausible application protocol for Smart City. However, this protocol differs from HTTP on the header size. It only uses 2 bytes for the header, which results in a much lower power and data consumption for the device, and much better performance for the server (the parametrization and configuration capabilities provided by HTTP are lost, yet not important for the scenario). For example, in tests, Mosquitto broker (the most common MQTT server), showed as maximum number of active clients (each sending one message per second) sixty thousand at a 50% CPU usage [4]. For this reason, nowadays this protocol is frequently referred as the "IoT protocol".

Another alternative, CoAP (Constrained Application Protocol), is an application protocol designed to low-powered devices. Its operation method is request/reply (device-to-device), similar to HTTP, which is good. The major difference from this protocol to the other two mentioned, is that it is implemented over UDP/IP. In a glance, this appears to be better, because UDP is a lighter protocol (for example, there is no waiting for connection establishment). However, regardless the communication protocol (Wi-Fi, LoRa or GSM), in a

3 http://dev.chromium.org/spdy/spdy-whitepaper
4 http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf

Smart City the signal intensity will not be perfect, and the devices will be low-powered, which means there will be many communication failures. UDP is a protocol that does not implement any packet-level error recovery mechanism, while TCP does it (although it is not important to ensure that every single message is received, under a 2G GSM scenario, it is expected the loss of many messages). This way, even being available DTLS (TLS for UDP), being available for many microcontroller architectures and having an header of only 4 bytes (light-weight protocol), it should only be implemented in a "trustable" network, which is not the case of Smart City scenario.

Finally, the AMQP (Advanced Message Queuing Protocol) is a whole new specification/technique. Although there are not many tests executed/published, theoretically it performs relatively well for its complexity. The biggest advantage of this protocol is the flexibility, because it can work over TCP or UDP, can perform device-to-device, cloud-to-device and cloud-to-cloud communication, and it works on TLS or DTLS. This flexibility could bring a new vision to a Smart City, allowing each device to communicate in its own ways (although it would increase the complexity of the whole system). The inconvenience of this protocol is the accessibility. While the other three mentioned protocols, for clients and servers, are very well documented and available in various programming languages (like python, node.js and c/c++), this one is poorly documented and its availability is low (both in terms of languages and architectures). For this reason, this protocol is not yet considered suitable for Smart Cities.

As can be concluded from the above reflection about the available application protocols, only MQTT and HTTP (if optimized for smaller headers) are plausible for the scenario under consideration. However, as Smart City can bring new challenges for the current available protocols, it may justify the creation of a new one combining HTTP with MQTT, where the communication is natively device-to-device (as HTTP), the headers are minimized (as MQTT) and works over TCP (as HTTP and MQTT).

To summarize the evaluated application protocols, refer to table 2.

	HTTP	MQTT	CoAP	AMQP
Lightweight	No	Yes	Yes	Yes
Internet protocol	TCP	TCP	UDP	TCP / UDP
Availability	Very high	High	High	Very low
Scope of communication	D2D	D2C	D2D	D2D / D2C / C2C
Security	TLS	TLS	DTLS	TLS / DTLS

Table 2: Brief comparison between plausible application protocols. From scope of communication, the nomenclature used is the most common, where D2D is device-to-device, D2C is device-to-cloud and C2C is cloud-to-cloud.

3.4 INFORMATION SECURITY

With the introduction of the IoT concept, the security of the transacted information, as well as its privacy, are probably the most discussed issues lately. From ”what is the best authentication method” to ”should every device on Earth be authenticated?”, there are a lot of open questions.

Although the lab supporting this work is involved with its ”sponsor” (DigitalSign), it is crucial to understand the operation of the most common solutions. After a brief research, it came out as authentication solutions for IoT the usage of passwords, tokens, certificates and implementation of blockchains. To help understand them, a brief clarification of their advantages and disadvantages follows.

Passwords are probably the oldest method for authentication on the web. Nowadays, this security measure is considered weak, because it is subjected to several vulnerabilities. A password is a shared secret, typically 4 to 16 bytes, created by the end user (sometimes, with some rules from the server, like at least one number). However, as these secrets travel through the Internet (many times, not encrypted), it becomes possible for an hacker to be ”listening” that communication and ”steal” the password. Another vulnerability, is related to the end user that, being human, tends to create easy to memorize secrets (like 1234 or special dates). In result, there are available on the web giant lists of the most used passwords, for hackers to use in brute force attacks. This vulnerability gets worse, if the target user habits/life is known for the attacker. Although, it is important to remember that this mechanism is very easy to implement. So, for scenarios where a rigid authentication is not required, it is a plausible solution. Furthermore, to overcome some of the known vulnerabilities, there are alternatives like dual factor authentication, where the user enters his password and has to confirm the access through another way, or one time password, where the user asks for a password which it is valid for a time-limited single utilization. However, these adapted solutions are not available for IoT devices, since they need human interaction.

Stepping forward, tokens. This is a piece designed to ensure security and, mainly, trust in an application. As an example of token, JWT (JSON Web Token) is an object that, in a safe way, represent a set of information between two entities (as defined in RFC 7519)[5]. Usually, after user authentication, a token is composed by:

- user data (attributes);
- hashing algorithm;
- then these two pieces are hashed (with the chosen hashing algorithm) with the server secret, resulting in the signature.

5 https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec

Then, the token is the junction of these three pieces (hashing information, user data and signature), which is sent back to the user. In the future, when the user wants to communicate with the application, he/she sends his/her token. To check the authentication, the application hashes (with the server secret) the first and second piece of the token, and if the result match the third piece (the signature), then the user is who he/she claims to be. To help understand its behavior, see figure 9. However, it is important to remember that this method is not meant to obscure or hide information, being advantageous to communicate the tokens under a TLS channel. This technique of authentication, not being perfect, because the token is only valid for the server that issued it (only for device-to-device communication), is a very light-weigh mechanism to implement authentication, as tokens can be as small as 80 bytes and the server do not need to store any information. In summary, a token works as a complex password, generated by the server and, compared to passwords, it has two advantages: impossible to guess (brute force attacks will not work); and the server can check the integrity of the token (if it is altered during the communication, the signature will not match). From this brief analysis, token seems to be a acceptable authentication mechanism for a Smart City environment. One very important to mention vulnerability of this mechanism, is that if the server secret is discovered, all the system becomes useless. Another known vulnerability is related to the sending of the token - as the authentication implies it to be sent, there is the possibility of be stolen and latter used by the hacker.

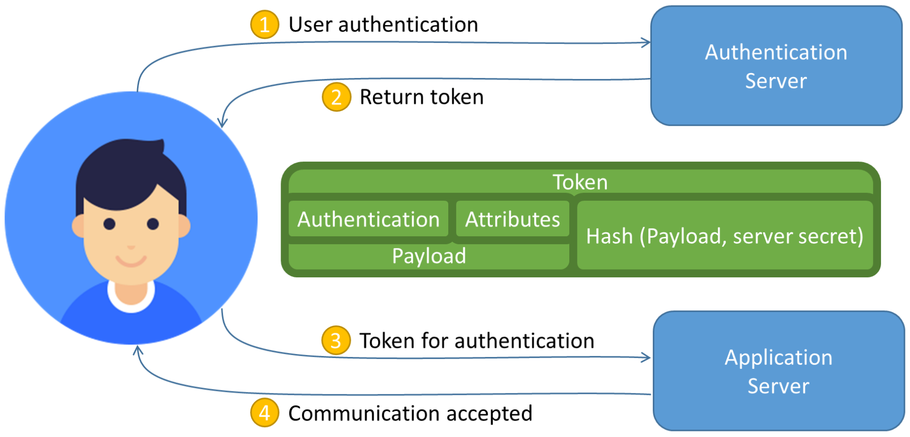


Figure 9: Creation, structure and validation of a token.

Concerning digital certificates, it is one of the strongest authentication mechanisms. It assumes that the end-user generates a key pair (public and private key), and creates a CSR (certificate signing request), which is the public key signed (hashed) with the private key. This CSR is sent to a CA (certificate authority) that will issue the certificate, which means that the CA ”recognize” that certificate. When the user wishes to be authenticated, he signs (hashes) the message with the private key and send the message, the signature and the public key. When the server receives the message, as the public key was sent too, he can decrypt it. It is important to mention that the public key is really public, and thus there is no problem in being stolen. Compared to the token, this method is ”better” because the private key is never sent through the web, it is generated by the user and never leaves it. As a result, this authentication method has very few vulnerabilities, being a common known one the physical theft of the private key. However, all this security comes at a cost, it is ”heavy” and expensive to implement a full PKI - keep track of CRLs (revocation lists), issue certificates and store them is computationally difficult. In summary, this authentication mechanisms delivers the desired security level to the system, however it may be hard to implement in small ARM devices, because they lack on encryption capabilities to sign messages by default. Another limitation of this mechanism is that it considerably increases the size of each message (more than 1kb).

Finally stepping into blockchains, the most recent integrity mechanism. The big innovation on this method is that it is decentralized, while the other methods are all centralized (only one identity can confirm the authentication of devices). The basic unit of operation of this mechanism is: the block, which is an object that has attributes, a unique incremental id, a time-stamp, a hash and a previous hash (always pointing to the previous block); a blockchain database, which stores all the blocks; and a node, which is a machine where the database is stored. For it to work properly, it requires many nodes (more nodes, more integrity). Every time a new block is created, every node updates its database. The security of this architecture relies exactly on the number of nodes, because if one node is hacked, the ”network” detects that it is different from all the others and kills it. This way, only hacking at least half of the nodes would result on an attack (which is why lots of nodes are suggested). Understanding this workflow, it is observable that blockchain is ideal to allow fast, transparent and secure transactions. Its main goal is to keep a record (a block) of every transaction, being impossible to delete it - delete a block will result in an error, because the previous hash of the next block would point to a non-existing block, leading to a corrupted blockchain.

Although this mechanism is still to be proven, for its purpose it already showed some good results, even considering the known disadvantages (irreversible transaction, for example). However, on the IoT field, register a device log in a blockchain is not very relevant.

As a curiosity, the real relevance of decentralized database as blockchain, is that no corruption could occur (as happens in the governments, for example). This corruption is only possible because all transactions are recorded in banks and in private entities, being possible to ”omit” them. If everyone had a node at home, this corruption would not be possible.

To summarize the evaluated authentication mechanisms, refer to table 3.

	Password	Token	Certificate	Blockchain
Lightweight	Highest	High	Low	Lowest
Security	Low	Avarage	High	High
Vulnerabilities	A lot	Some / 1 critical	Few	Depends on the number of nodes
Difficulty to implement	Very easy	Easy	Average	Very hard
Plausible for Smart City	Maybe	Yes	Yes	No

Table 3: Brief comparison between plausible authentication mechanisms.

3.5 STORE DEVICES INFORMATION

In a Smart City environment, where innumerable small devices send information in the form of logs (low-level information), the ability to store this information for latter processing is crucial. The typical answer for this problem is the usage of relational databases (like MySQL, Oracle and PostgresSQL). These databases are commonly used for a more rigid and structured way of storing data, which consists of two or more tables with columns (types of information) and rows (entries) and where tables must be related to others by at least one column. As a result of these complexity, relational databases may be unsuitable for low level information and millions of connections.

As an alternative, non-relational databases (like MongoDB, Oracle NoSQL and Redis) are much more flexible, being optimized for unknown/irregular information. Currently, there are four types of non-relational databases: key-value store, column store, document and graph database. This type of database is much more oriented to low-level information and, inevitably, to a Smart City environment.

An additional aspect that can be considered, is that non-relational databases are easily divisible, enhancing the scalability of the system. Furthermore, if an enormous number of calls and accesses are expected (Smart City environment), some of these databases can be built in RAM memory, improving drastically the performance of the system.

3.6 PROCESS DEVICES INFORMATION

As for processing, it is crucial a continuous non-blocking mechanism both to receive and process devices information. This is particularly important because the output (high-level information) is expected to be a real-time monitoring system. To achieve this level of performance, it is important to consider the available programming languages, mainly the support of synchronous and asynchronous paradigms. With synchronous programming, the program is executed line by line, and if it needs to get a client response, or get/store data, or process an image, it will be ”stuck” until the task is finished. If it is desirable to execute parallel functions while ”stuck”, they have to be launched in threads. However, in a computer, the threads are a limited resource and consume more power and, depending on the implementation, RAM. To overcome this limitation, asynchronous programming is a single core and thread method that allows the program execution to do other things when a function will take some time. For instance, if a web requests takes one second to be performed, the main event loop is informed that this function only needs ”attention” in one second, and the program execution can do other functions. With asynchronous programming, if the program needs to perform ten web requests (each one takes one second), they are all ”launched” sequentially, the program can execute other instructions meanwhile, and one second after they are all done and returned (similar to interrupts). Furthermore, this programming paradigm takes advantage of the whole CPU core, unlike synchronous programming. Figure 10 illustrates the operation of the two paradigms of programming.

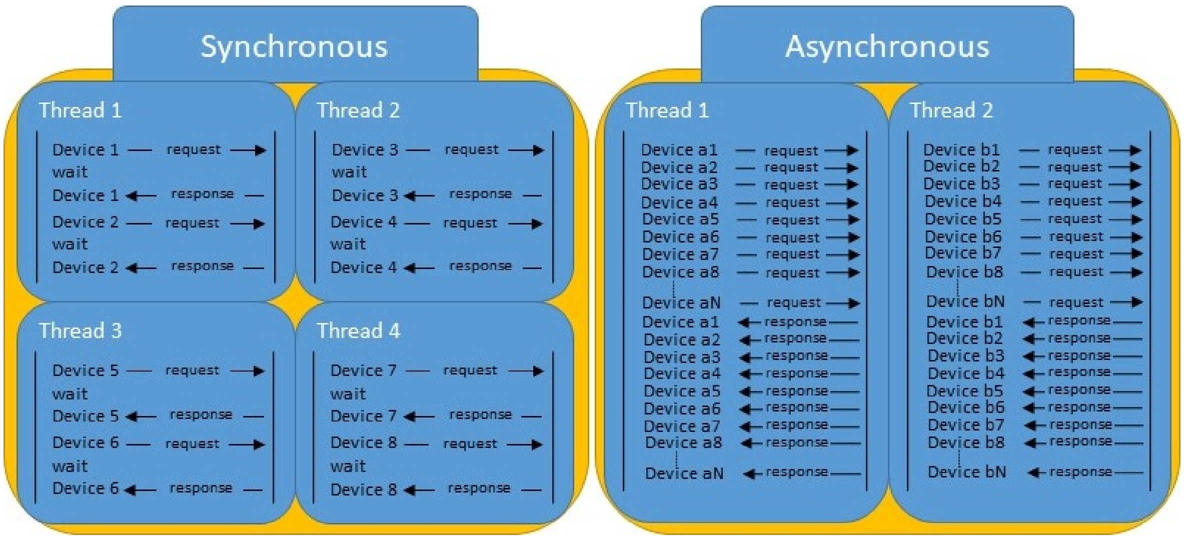


Figure 10: Synchronous and Asynchronous architecture.

Currently, there are three main languages able to work asynchronously:

- c/c++ oriented to micro controllers, through interrupts (not suitable for this scenario);
- Node (natively asynchronous);
- Python3 (needs to import event loop).

About these two plausible programming languages, Node stands out for its performance, while Python3 stands out for its simplicity and diversity of modules (useful for working the data).

3.7 PLAUSIBLE ARCHITECTURES FOR SMART CITY

Faced with the problem of dealing with many requests (from simple user requests to data generated by them), performing links between them and compiling new information or useful instructions for the end-user, it becomes essential to use an orchestration platform.

According to the latest years of development, the theoretical way to go is the Cloud Computing (in the olden days, refereed as centralized computing). This topology (already discussed in section 2.1.2) is based on a vast number of interconnected machines with no graphic capabilities, which the purpose was to receive requests, process them (sometimes ”smartly”), and return a useful response to the user. Nowadays, this so big concept of Cloud, consists of enormous data centers, where complex algorithms of big data and machine learning are applied to everyone data (sometimes, it may even be intrusive for personal information). With the introduction of the Internet of Things, an assumption was made that every IoT should be connected to the cloud. Although this implies a huge amount of processing power from the data centers (which is achieved by adding more machines), the limitation that cannot be overcome is the lack of bandwidth. Until a while ago, a regular citizen had one computer, one cellphone and one or two laptops. Last years, witnessed the emergence of small devices, disposable, that connect to the cloud. The ease with which these IoTs are introduced in society is so much, that will saturate the cloud, leading it to take minutes to process and answer to requests.

The solution to overcome this limitation, relies on the introduction of ”nodes”, between the devices and the cloud. Each node has a dedicated gateway and its role is to aggregate the low-level data from a group of devices into high-level information. This way, the bandwidth and the processing power is distributed between these nodes, and the cloud only receives information from the nodes. This new topology is followed by two paradigms (similar), the Fog Computing and the Edge Computing. The duality between these two, relies on the ”distance” from the devices.

Fog Computing claims that many nodes (simple machines) should exits, each one responsible for a limited and controlled number of devices. For example, a node can be a Raspberry Pi, responsible for parking sensors in a parking lot. This solution has the advantage of drastically decrease the computational power at the cloud, but the bandwidth is not very optimized (since it may justify too many nodes).

Alternatively, the Edge Computing claims that the nodes should be more powerful machines, responsible for a vast number of devices, with different functionalities, leading to a small number of nodes (consequently, less communication to the cloud). This solution places the nodes closer to the cloud than to the devices (optimizes the bandwidth at the cloud, but is not so effective with the computation power required). Furthermore, as each node deal with a higher amount of information, some minimal mechanisms of big data and/or machine learning can be applied.

To better understand the duality between these two paradigms, refer to figure 11.

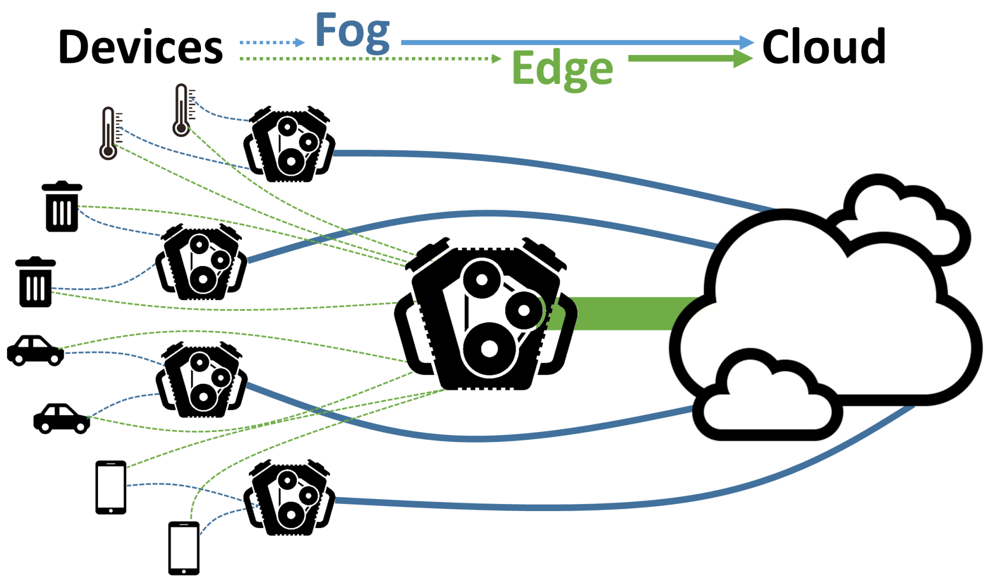


Figure 11: Duality between Edge and Fog computing. In this image, for the same devices and cloud, blue lines and arrows represent the data flow according to Fog Computing and the green ones represent the Edge Computing. Furthermore, dashed lines means low-level information and filled lines/arrows means high level information (where its width represent the amount of high-level information generated).

Although the difference is small, Edge Computing offers an interesting possibility, the introduction of security mechanisms. Nowadays, authentication and encryption is a hard, expensive and important concept to consider. At the cloud, security is hard to keep because of the huge amount of devices, following Fog Computing, security is expensive and ”heavy” to implement in ”small nodes”. Finally, with the Edge Computing, it is the most ”balancing” place to ensure it. Each node has enough resources to handle a PKI, and will deal with an admissible amount of devices.

4

DEVELOPMENT

This chapter intends to justify the decision made for the available technologies and explain in detail how to implement and integrate them to achieve the final proposed architecture. At the end, it also states some of the expected outcomes from the Edge.

4.1 DECISIONS

After the research performed in the above section, there are some important decisions to be made, mainly regarding to the application protocol, information security, devices information storage, process devices information and plausible architecture.

4.1.1 *Application Protocol*

Concerning the application protocol, the decision lies between MQTT and HTTP. Since both fulfill the requirements, the decision is based on performance. After some research, MQTT can be 93 times faster (in 3G network) than the commonly-used HTTPS. This protocol is optimized for IPv6 (which may be introduced anytime in a near future), and also allows the possibility to encrypt the communication with TLS protocol. Within the Smart City scope, as mentioned, the battery optimization is also critical, where MQTT stands out again. Table 4 shows a comparison between both protocols over the acceptable communication protocols. For these reasons, MQTT is the application protocol that the Edge under development implements.

To remember, MQTT is a protocol oriented to device-to-cloud communication, implemented with a publish/subscribe mechanism, which information flow can be seen in figure 12. However, as for Smart City scenarios a device-to-device scope is more appropriate (for security reasons, mentions in section 3.3), only the Edge is authorized to subscribe to topics. The devices will only be able to publish, and the subscription will be denied, resulting in a device-to-device architecture.

		3G		WiFi	
		HTTPS	MQTTS	HTTPS	MQTTS
Receive	msgs / hour	1.708	160.278	3.628	263.314
	% battery / msg	0.01709	0.0010	0.00095	0.00002
	msgs	240 / 1024	1024 / 1024	524 / 1024	1024 / 1024
Send	msgs / hour	1.926	21.685	5.229	23.184
	% battery / msg	0.00975	0.00082	0.00104	0.00016

Table 4: Performance and power consumption comparison between MQTTS and HTTPS.

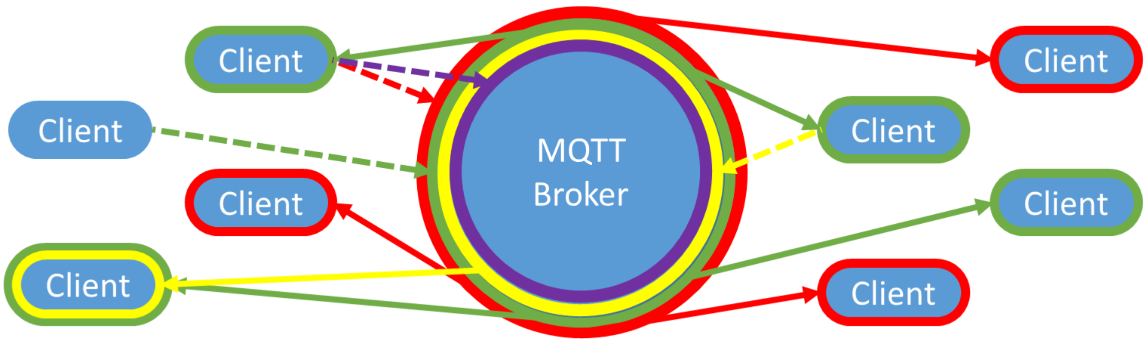


Figure 12: Information flow of MQTT protocol (publish/subscribe oriented). In this image, the published messages are pointing to the broker (dashed ones), while arrows pointing to devices (filled ones)are the messages subscribed. The contour of each device represents the topics it subscribed. Notice that every scenario is possible: (i) publish to a topic that no one subscribed; (ii) subscribe to a topic that no one published; (iii) a client can publish to multiple topics; (iv) a client can subscribe to multiple topics; and (v) a client can publish and subscribe simultaneously.

4.1.2 *Information Security*

About the information security, the decision to be made is based on authentication mechanisms. As described, the usage of certificates delivers an higher authentication level, but may be unsupported when talking about about microcontrollers level, and uses more data over the network. The token, although delivers a lower authentication level, is much more lightweight and requires no processing efforts. However, as this project is powered by a company that deals with digital certificates, it is a requisite to implement it, through a PKI.

4.1.3 *Store Devices Information*

To store devices information, the decision between relational and non-relational database needs to be done. Considering the type of information resulting from a Smart City (like temperature, humidity, parking lots, people concentration, and much more), there are not evident relations between the information. Furthermore, the scalability of the whole system is a requisite, which can only be achieved with non-relational databases. Another aspect that can be verified, is that for fifty thousand entries, MongoDB takes about nine seconds to create them. As this number of devices publishing low level information that only needs to be stored temporarily is expected, a new different database is required, as Redis. This tool is an in-memory (RAM) key-value data structure developed essentially aiming the high performance, that offers a much better performance (up to ten times faster). As it uses RAM memory to store the data, and it is only needed temporarily, Redis can work as a pool of low-level data, and MongoDB periodically process the information from Redis. As can be seen in figure 13, Redis takes only one second to create the same fifty thousands entries. If the data is received from the devices faster than it is processed, Redis can work as a very fast buffer, important to ensure that the system is not led to instability. Another aspect that is useful about Redis, is that it implements a mechanism to delete entries after a specified time, automatically. This feature is particularly important because low-level information does not need to be long-lived, and there is no need to perform queries to delete entries, nor to implement mechanisms to count the live-time of each entry.

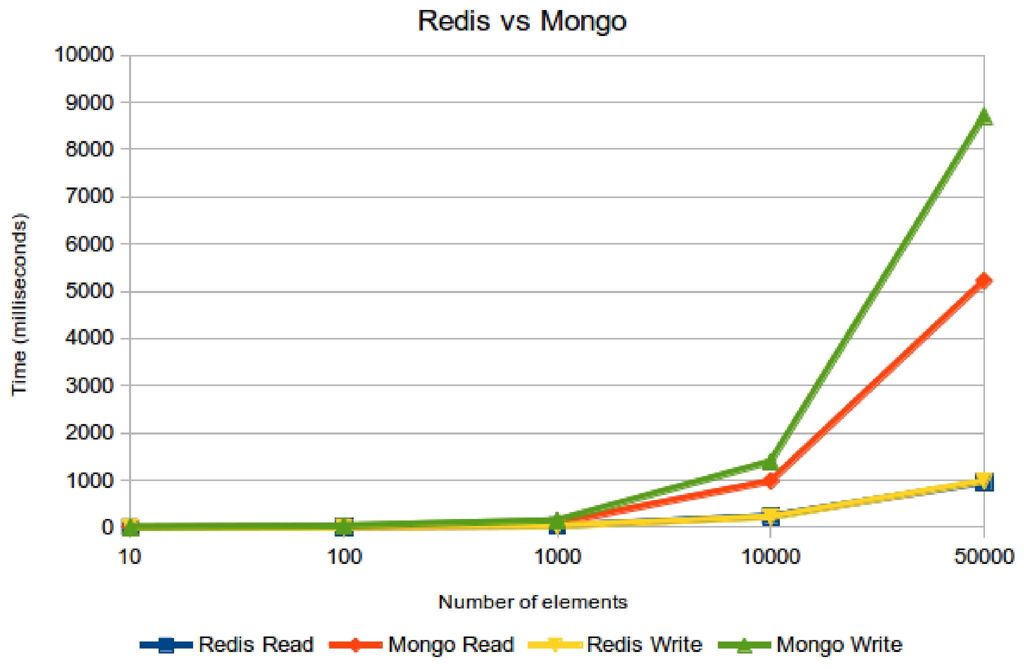


Figure 13: Performance comparison between Redis and MongoDB.

4.1.4 *Process Devices Information*

As to process devices information, understood the advantage of asynchronous programming to handle innumerable devices (requests), the decision is evident. As for the language to use, the decision was based on the programming experience. Since the author showed skills programming in Python, this was the language adopted for the whole project.

4.1.5 *Plausible Architectures*

Finally, about plausible architecture, the Edge Computing was the chosen one (although it would be important to test both Edge and Fog Computing, it was not possible due to time constraints). This architecture is believed to fit better the problem because, when dealing with such a huge scenario as the Smart City, it is more practical to deploy one Edge (or at least a few), than spread a lot of nodes throughout the city. Another advantage is that the Edge nodes should be placed all in the same facility (easier for the operator), while the Fog nodes must be placed near the groups of devices (which is logistically complex for the city). Furthermore, the possibility to integrate a PKI in the Edge is believed to add value to the solution, even from a business model point of view.

To synthesize the global behavior expected from the Edge applied to the Smart City, figure 14 illustrates the expected.

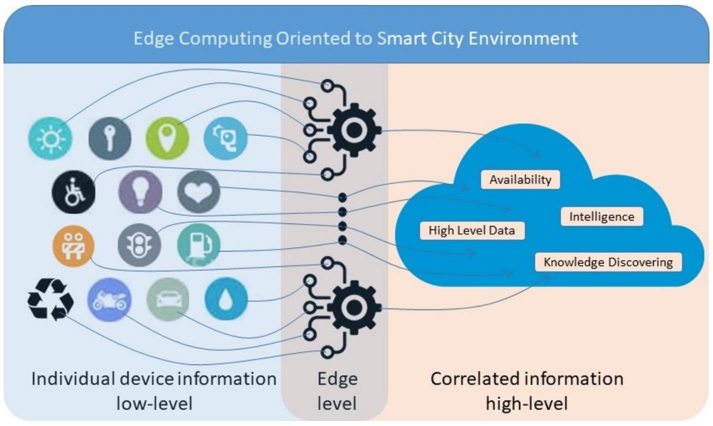

Edge Computing Oriented to Smart City Environment
Availability
Intelligence
High Level Data
Knowledge Discovering
Individual device information low-level
Edge level
Correlated information high-level

Figure 14: Edge Computing vision.

4.1.6 *Synthesis*

In summary, this project consists in the development of an Edge. This unit can be schematized by figure 15, and is built using:

- the MQTT protocol as Application protocol with TLS;
- ADSS as a PKI, to manage devices certificates;
- Redis as a temporary high-performance pool of low-level data;
- MongoDB to store the processed information, ready to be correlated;
- all these modules will talk between each other asynchronously, using Python.

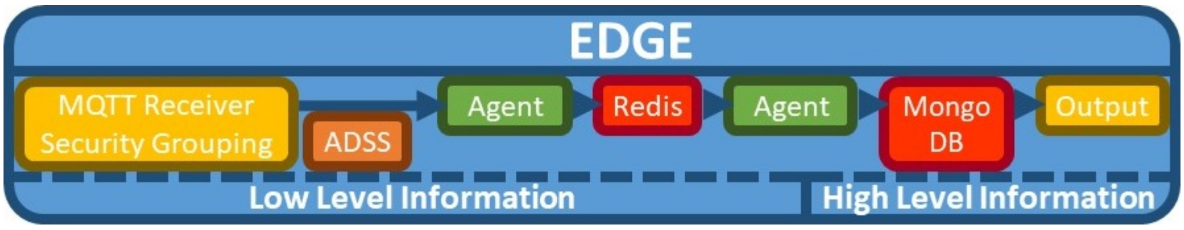


Figure 15: Synthesis of the proposed edge architecture.

As for the devices, ideally these should be ”disposable”, communicate over GSM and each one (ideally) has a private key and a certificate, to ensure authentication.

4.2 IMPLEMENTATION

Concerning a Smart City environment and assuming that the security of the information is primordial, there are five major pillars that need to be ensured:

- ability to receive as much information (low-level information) as possible;
- this information needs to be stored (temporarily) faster than received;
- the ”security level” assigned to each received information (based on encryption and authentication) must be registered;
- a solid interface able to filter, combine and relate all the low-level information into high level information;
- an attractive API responsible for delivering this information to users.

To achieve these pre-requirements, it is important to think about the global work-flow of the system and define it. To better understand the next detailed stages, see figure 16.

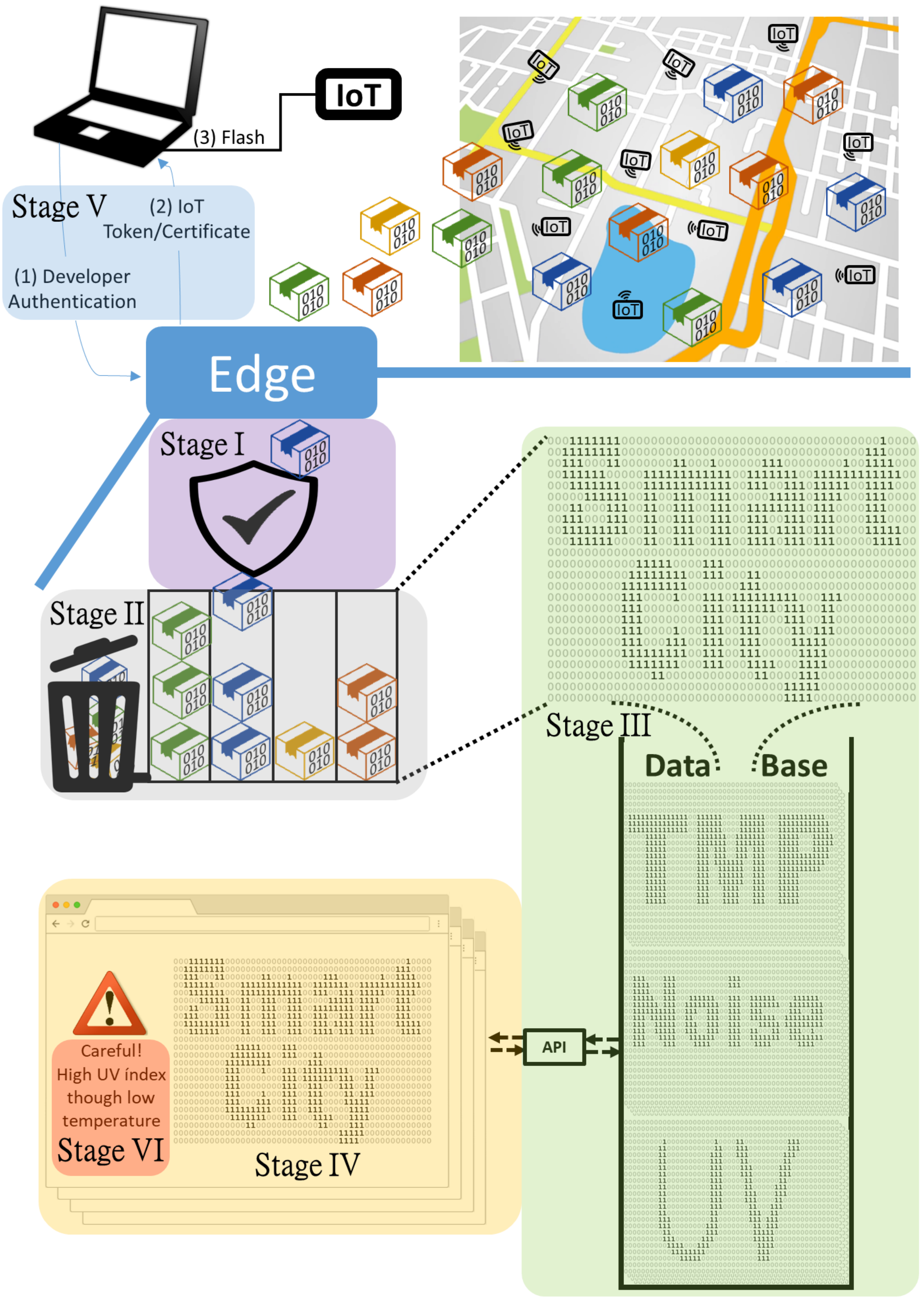

IoT
(3) Flash
Stage V
(2) IoT Token/Certificate
(1) Developer Authentication
Edge
Stage I
Stage II
Stage III
Data
Base
API
Careful! High UV index though low temperature
Stage VI
Stage IV

Figure 16: Implementation stages.

4.2.1 *Stage I - data classification*

In these days, the Information Security is complex to understand/implement, fundamental and pretty expensive. This last reason is particularly important, because it is impossible to imagine a Smart City where every IoT has its own certificate. This is why this Edge, instead of filtering the information, will classify it, allowing the coexistence of different kind of devices. This way, there may be critical devices with certificates, and others without. So this Edge, as the messages arrives (via MQTT), they will be tagged with the security level from the source. This security level can be:

- 0 - public, oriented to personal devices that do not care about security issues (no costs associated);
- 1 - encrypted via TLS, which ensures the confidentiality of the information and avoids one of the most common attacks, the Man in the Middle (this security level is accessible to all, and also does not entail any costs);
- 2 - encrypted via TLS and authenticated with a server generated secret (token), where the user authenticates itself in a system and it returns the token for the IoT device. This security level, in addition to ensuring confidentiality, also ensures integrity as well as authenticity, although at a lower level (this solution may already be associated with some costs, since it is necessary to save and manage all the keys);
- 4 - encrypted via TLS and authentic with a x509 certificate, which ensures both the confidentiality, integrity and authenticity of the information, almost at a perfect level. The perfection of this technique only depends on the level of certainty that a certificate is not physically stolen/cloned from the device (this solution is expensive, since the costs to maintain a PKI are high).

The available security levels are synthesized in table 5.

Security Level	Description	Confidentiality	Integrity	Authenticity	Final User Costs
0	Public	None	None	None	None
1	TLS	High	None	None	None
2	TLS + PWD	High	Low	Low	Average
3	TLS + CRT	High	High	High	High

Table 5: Comparison between the available security levels.

Furthermore, as for privacy, by default no personal/device information is stored. If the user wants the Edge to track the device, it should be explicit in the message. The user may wish to track the device in three ways:

- by ID, where the Edge only feedback if the IoT is online or not;
- by ID and GPS, where the position of the device last time it was online is stored;
- by ID, GPS and data, where the full historic of the device is stored.

Everytime a message is received from a device, this classification occurs. For example, if a device sends a public message like this (in JSON format):

{'id': 'ID123', 'gps': [41.4507, -8.2933], 'temperature': 19.5, 'track': []}

it will be tagged with the security level, becoming:

{'gps': [41.4507, -8.2933], 'temperature': 19.5, 'security': 0}

As no tracking was requested, the ID of the device is ignored. If the user want Edge to track the device, it should be explicit. For example, for a device communicating over TLS and with certificate that wish to be tracked with GPS, the message:

{'id': 'ID123', 'gps': [41.4507, -8.2933], 'temperature': 19.5, 'track': ['gps']}

would be stored as:

{'id': 'ID123', 'gps': [41.4507, -8.2933], 'temperature': 19.5, 'track': ['gps'], 'security': 3}[1]

The message resulting from this stage is then passed to the next stage, to be temporary stored.

4.2.2 *Stage II - temporary data storage*

This stage is concerned with the meaning of each type of information. Whenever a log is registered in Redis (the temporary pool of low-level information), the lifetime of that message is defined. The idea behind this topology, is that the pool of data will only contain logs that are significantly valid. To clarify this idea, lets assume the next example, where a device sends a message like this:

{'id': 'ID123', 'gps': [41.4507, -8.2933], 'temperature': 19.5, 'noise': 50}

Looking into the meaning of each value, it is obvious that the temperature at that position will remain the same for thirty minutes, whereas in the next second the noise could be zero. This interval, where a value is considered to be physically true, is the time that a message

1 in the next stage, will be shown that this is not exactly true. This information is stored, but not in this format

will live in Redis. So, for every kind of value that the Edge receives, there must be a lifetime defined for that type of information.

It is important to notice that the lifetime of a message has nothing to do with the periodicity of the device messaging. It is possible to exist a device, a mobile one, that every five seconds sends a temperature value. Each message will still remain on Redis for the specified thirty minutes. Obviously, the number of logs in the pool will grow, but only until the thirty minutes. After that, everytime a message arrives, there is one that disappears too. For the given example, after the thirty minutes, Redis will keep constantly 360 logs from that device.

Focusing on this technique of dealing with information, it seems at first glance that a drop of temperature momentarily will only be tracked thirty minutes after. However this is not true, and for explaining why, imagine that the temperature is 15 degrees at 15:00:00, and at 15:15:00, it instantaneously drops to 10 degrees, remaining in this value. Every 5 seconds the device sends the temperature to the Edge. Table 6 shows the evolution of the resulting temperature on the Edge.

Time	Real Temperature	Number of 15 degrees logs	Number of 10 degrees logs	Temperature at the Edge	Deviation
15:00	15ºC	1	0	15ºC	0
15:02	15ºC	24	0	15ºC	0
15:04	15ºC	48	0	15ºC	0
15:06	15ºC	72	0	15ºC	0
15:08	15ºC	96	0	15ºC	0
15:10	15ºC	120	0	15ºC	0
15:12	15ºC	144	0	15ºC	0
15:14	15ºC	168	0	15ºC	0
15:16	10ºC	180	12	14.7ºC	4.7
15:18	10ºC	180	36	14.2ºC	4.2
15:20	10ºC	180	60	13.8ºC	3.8
15:22	10ºC	180	84	13.4ºC	3.4
15:24	10ºC	180	108	13.1ºC	3.1
15:26	10ºC	180	132	12.9ºC	2.9
15:28	10ºC	180	156	12.7ºC	2.7
15:30	10ºC	180	180	12.5ºC	2.5

Table 6: Behavioral example of Redis variables.

As can be observed, at the moment of the drop down, the deviation is maximum, but it will gradually adapt to the real value, describing an exponential negative function. 15 minutes after, the error is 2.5 degrees, which for a temperature value, is not very drastic. Furthermore, it can be improved with a mechanism that weighs more ”fresher” values than the ”old” ones. This solution would make the deviation curve to have a more negative exponent, converging faster to zero.

So, according to this stage, the plausibility of the resulting information will depend on the lifetime of each type of variable. This is a very critical decision that will affect the response time of the Edge, being important to be chosen wisely. This decision should essentially reflect two parameters: the desired response time; and the maximum "acceptable" error allowed.

One more feature that this Edge offers, the tracking of the device, starts in this stage too. As already mentioned, it is desirable to be able to track the status of a device (online or offline), if the user so desires. To achieve this, an identical solution was adopted. Whenever a device with tracking desire sends a message, he is obliged to send the estimation for the next message to be sent. For example, lets assume an IoT sends this message encrypted:

{'id': 'ID123', 'gps': [41.450, -8.293], 'temperature': 19, 'noise': 50, 'track': ['id'], 'sleep': 17}

From this message, Redis would store the next three messages:

{'gps': [41.450, -8.293], 'temperature': 19.5, 'security': 1}

{'gps': [41.450, -8.293], 'noise': 50, 'security': 1}

{'id': 'ID123'}

where the first one remains in Redis for 30 minutes (for example) and the second for 1 second. These two messages "lost" the device ID because it is not relevant to build a temperature or noise map (enhannces the device privacy). The only important information that needs to be stored is that the device which generated it (whichever it was), communicated over TLS. About the third message, it will be stored only for 17 seconds, being automatically deleted after that time. If the device communicates again in the next 17 seconds, the "auto-delete" timer will be renewed. This way, while the device keeps communicating every 17 seconds, the third message will always be on Redis. If for some reason the device takes more than that time to communicate, than the log on Redis will be deleted. So, to check weather a device is online or not, it is simply needed to check if its ID is on Redis.

4.2.3 *Stage III - low to high level converter*

Until this stage, the Edge has a huge, dynamic and disorganized pool of data, where every little message has associated the security level of its source. So, this stage has the function of structuring the low-level data into high-level information.

The algorithm to perform this task was built in python, asynchronously and modular. For each variable (like temperature), the program implements an autonomous subroutine that connects to the Redis and, periodically, grabs all the data with respect to its variable (temperature, in this example). This subroutine, now contains an array of objects like this:

```
[0]{'gps': [41.473, -8.302], 'temperature': 18, 'security': 1}
[1]{'gps': [41.462, -8.385], 'temperature': 23, 'security': 0}
[2]{'gps': [41.419, -8.332], 'temperature': 22, 'security': 3}
[3]{'gps': [41.492, -8.726], 'temperature': 19, 'security': 1}
: : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : : :
[N]{'gps': [41.425, -8.951], 'temperature': 16, 'security': 2}
```

With an array like this, periodically updated and with many redundant information, the possibilities are too many. As this stage is the only one where the data is evaluated in real-time, this is probably the most important one, because it will synthesize all the logs (purpose of the Edge), determining the value of the information. The theory behind the aggregation of these logs should be performed by someone expert on the use of each particular variable. As an example, lets assume two variables, the temperature and the level of waste in rubbish bins. While for the temperature it makes more sense a colored map with the averages from all the low-level logs, for the rubbish bins it is more informative a map with a level bar in each bin location. However, the evaluation for each variable is very different. Although this example only approaches two easy to understand variables, more complex ones may exist, like water quality of rivers, air quality, noise, among others. For these variables, as its behavior is not easy to understand, more complex algorithms may be required. So, for this stage, it is critical to know precisely the behavior of the relevant variables. The more realistic the models are, the higher the value of the Edge output will be.

In addition to the evaluation on each variable, it is still necessary to consider the registered level of security of the logs. From this "tag", it is possible to improve the final information in two ways:

- if there is abundance of information on a specific region, the algorithm can choose only the ten (for example) most reliable ones, or order the logs by security level and use only the top ten;
- to the final information, the Edge can couple the average security level of all logs used in its construction. This would allow the final user to take a more care decision about some event. For example, if in a specific area 100 public devices indicate temperatures of 150 degrees, the final map will indicate a low level of trust, being probable that it is some type of hack. If the 100 devices are certificated, then the final information will have a maximum level of trust, and the final user knows for sure that a fire is starting.

As could be observed, each variable has its own profile (periodicity to be evaluated, different algorithms and different output forms). This is why this stage was designed to work in blocks, where each variable is treated in its own block. This topology allows for different blocks to be developed by different specialists. As each block is autonomous (neither affects nor is affected by others), each one has its own periodicity. This is particularly important because for some variables, it is important to evaluate the data more frequently (like noise), but other can be evaluated hourly (like ultraviolet radiation).

Finally, each block has a private connection to the MongoDB, through a client that is only allowed to write to its variable table (ensuring integrity). MongoDB is the database that will store all the high-level information, "forever". This database will be responsible for keeping an historic of each variable. Although not yet implemented, there will be a block only responsible for the historic. This block will define, for each variable:

- transitory regime duration - period of time that a variable should be stored according to the variable periodicity;
- permanent regime periodicity - infinite period of time, to where the transitory regime information will migrate, with a lower frequency.

To clarify this idea, lets assume again the temperature, and that its block generates a map every minute. This means that, every minute, a new entry in the table "Temperature" (in MongoDB) is created. Lets assume that the transitory regime is one day, and the permanent regime periodicity is one hour. This would mean that for the last 24 hours, there would be entries every minute (60 x 24 = 1440 temperature maps), and after that, every 60 entries will be joined forming hourly entries. This measure intends to relieve the need for disk space, since for historic if does not matter very high temporal resolutions. Again, every variable must have its own transitory regime duration and permanent regime periodicity. However, contrary to the algorithm, these two values are not critical for the systems. Furthermore, it makes more sense that these two values can be chosen by the end user, since he is who knows what the information is for.

As an output, for global system security, it is not desirable that external entities interact directly with the MongoDB to grab data, even if they only have read privilege. To achieve this, an API was developed to mediate the queries to Mongo. This API implements global functions like:

- getMap(type), to get the last map of type "type" (like temperature, noise, etc.);
- getMaps(type, number), to get the last "number" maps of type "type";
- getMapsFrom(type, from), to get every map of type "type", from "from" date;
- getMapsFromTo(type, from, to), to get every map of type "type", from "from" date to "to" date;
- getMapFromToGPS(type, fromlat, fromlon, tolat, tolon), to get a map of type "type", from the square specified;

In addition to these functions, more were implemented. However, it is not necessary to specify all of them in this dissertation.

4.2.4 *Stage IV - information distributor (dashboards)*

With the implementation of the above three stages, the theory behind the Edge Computing is achieved - a system that receive low-level data and transforms it into high-level information, relieving the Cloud of many connections and ”basic” work. However, this project intends not only to develop an Edge, but also to give usefulness to the information. To achieve this, a set of dashboards were built, for the final user to operate.

About this stage, there is one topic that must be clarified. The output of an Edge and these dashboards are not intended to manipulate actuators in the city (the Edge purpose is mentioned in section 1.2, offer an integrated, solid, secure, trustable and user-friendly decision-making help desk). So, the critical topic on this stage is exactly the ”creativity” behind each dashboard.

To demystify this idea, lets assume the temperature again. As a front-end, it can be created a platform that only shows the temperature map over the city map (this would be the simplest use-case). However, with the structure behind the Edge, it can be created a platform that, in addition to show the temperature map, can implement:

- an alert panel that shows drastics variations in specific locals, anticipating fires (or risk of it), for example;
- a side view panel with recommendations, for example usage of sunscreen due to high temperatures;
- possibility of pan and zoom over the map, always showing the security level of the information in view;

These three items are just simple examples of the information that can be extrapolated from the temperature variable stored in the Edge. However, the creativity is the limit. The more information that can be extracted and displayed, the more usefulness the information will have and, consecutively, more valuable the final solution will have for the end-user.

As for the previous stages, here is also important to evaluate each variable independently, and think about the most appealing way of displaying the information. In a first stage, this Edge is expected to have dashboards to:

- see temperature information;
- see general air quality (based on the common particles);
- track public transports trajectories, and offer ETA (estimated time of arrival);

- track waste level on containers;
- and, eventually, track some temporary parking lots for cars (like in front of pharmacies).

Recalling what was written in section 3.1 about IoT devices, these are small things ”dropped” in the City, probably thousands of them. So, the interaction with the developer is null (or almost null). For this reason, it is extremely important a platform where the owners of IoTs can follow its behavior (where it is and if if is online or not). However, this platform, as mentioned, raises some privacy issues, because a user does not have the right to know the status of other users IoTs. To overcome this privacy invasion, one limitation has to be imposed: only authentic devices (with certificate or token) can be tracked. In order to achieve so, the device certificate or token must have an attribute with the owner ID. This allows the user to use that same ID to perform authentication the Edge, allowing to query the database only for her/his devices.

If public and encrypted devices become very important to be tracked, it is essential that the owner, when deciding to enable the tracking by the Edge, is made aware that this tracking will be visible by everyone.

4.2.5 *Stage V - security distributor*

As already mentioned, about security it is desirable that the Edge implements encrypted and authentic communications. About the encryption, the solution is almost obvious, implement TLS. This protocol has already been studied, is safe, open-source and is currently in a well-consolidated version. As the application protocol in use already implements it, the only thing required to add was the certificates. As requirements, the Edge must have a key pair (private and public) and ask a CA (Certification Authority) for a certificate. Then, to implement the TLS protocol with MQTT, it is only required to give the broker the CA certificate, the Edge private key and the Edge certificate. With this information, the devices can verify the identity of the Edge with the CA certificate, and with the Edge certificate they encrypt the message to send, being only the Edge able to decrypt it, with its own private key. One of the advantages of using the TLS protocol, is that this process is all transparent to the developer.

For the authentication, it was implemented two mechanisms: by token[2] and by certificate.

For tokens, MQTT offers the possibility to store a ”password” file. The authentication mechanism behind this is that a device (or the owner) should create a password, and ”register” it on MQTT. For the registration, the MQTT hashes the password with a owned secret, generating a signature. The resulting signed password (a string) is stored in the authenti-

2 in chapter 3 is presented a specific token (JWT) which was integrated with the MQTT technology.

cation file and it is irrecoverable. When a device initiates the connection with the broker, it sends the token and the MQTT does the same process, and then checks if the resulting string is already on the authentication file. If it exists, the communication is accepted and has a limited life-time, otherwise the communication is rejected.

As for the certificates, a more complex solution is required, because the idea of a ″certificates file″ does not exist. To achieve this, it is required a Public Key Infrastructure (PKI). In this project, the PKI is the ADSS, integrated with a database (Postgres) to store all the required information. The ADSS works as the authority that receive certification requests, sign and deliver the certificates, revokes certificates and publishes CRL (Certificate Revogation List). This list contains every revoked certificate that was ever revoked so, a certificate that is not in any CRL is an active one. To integrate this with the MQTT, the broker has a specific filepath where CRL files can be placed. When a message arrives, it checks if the presented certificate is in the CRL or not. Again, if it is, the connection is rejected and if not, the connection can continue. However, this only checks if a certificate is active or not before the CRL CA. To clarify this, lets assume the CA A and the CA B. The Edge has the CRL of A. If a device has a certificate issued by B, when it is revoked it will only be visible in the CRL of B. So, when the device with the revoked certificate (from CA B) tries to communicate with the Edge, as the certificate will not be in the CRL A, the communication would proceed (which is, obviously, incorrect). To overcome this, when a device sends a certificate to the Edge, the MQTT will check whether it has the CRL from the CA that emitted the certificate in question. With the above example, when the device tries to communicate with the certificate from B (active or revoked), it will always be rejected because the Edge does not have the CRL from B. This method forces that only devices certified by ″known″ (for the Edge) CAs can communicate. The mechanism behind the authentication process (to ″prove″ that a message is sent by a specific device) happens as opposed to encryption. For authentication, the device signs the message with its own private key, and the receiver ″checks″ the message origin with the device public key.

For this project, although not crucial, a Certification Path was created. Usually, this is only created in public and visible infrastructures. However, as this project intends to implement the whole Smart City Systems, it was implemented too. To clarify this idea, a Certification Path is composed by a Root CA (which is the entity that has a wide recognition and value), sub CA (which are all ″children″ of the Root, and aims to deal with different areas) and End CA (where each one is a ″child″ of a sub CA, and this entity is the one who issues the certificates for end-users). On this project, the next Certification Path was created:

- Minho University
    - LabSecIoT
        - Smart City

Theoretically, the Root CA (Minho University) should be a officially certified CA (which has very restrict rules and is very expensive to achieve), the LabSecIoT is responsible for different areas (like Smart City, Smart Building, Smart Health), and the Smart City is the final entity that signs Smart City certificates for the IoTs. Theoretically, it could still exist under the Smart City CA different CA for different cities. This would be particularly important if it were desirable that a device from city A could not communicate to city B. However, towards the Root CA, both devices would be authentic.

Finally, the complex mechanism behind the authentication, it is necessary to clarify how to deliver these tokens and certificates to the end-user. Usually, the signing of the CSR (certificate signing request) is performed by a team of operators. However, this solution is not viable because it is expected a huge amount (millions) of IoTs. So, it becomes necessary to implement a mechanism to deliver the certificates automatically, without loosing trust/security on the process. To implement this platform, it was created a web page where a end-user can request a certificate or token for each IoT. To ensure security on the process, it is required that the end-user presents a developer certificate (which allows to request tokens or certificates, with a limit associated). Lets assume an example: an IoT company wants to certify 10 IoTs with certificates. The owner of the company buys to the city a developer authentication certificate with one attribute saying he can only ask certificates and another attribute with the number of certificates he can ask for (ten, in this example). Then, the owner authenticates itself with the bought certificate in the platform, and asks for the ten certificates. If he asks for more, it is rejected, and if he asks for tokens, it is rejected too.

To implement this platform, it is required to look for the two authentication methods differently, because one returns a certificate and other a token. Starting with tokens, he authenticates himself towards the platform, and the Edge generates the token (string), registers it on the MQTT password file, and gives it back to the user. Once returned to the user, the password is no longer recoverable. Furthermore, when the user registers the token, it must be defined a life time (between 1 and 365 days), after which the token will be expired and removed.

As for the certificates, a much more exhaustive research had to be performed. As the ADSS was designed to be handled by an operator, a deep study of the client SDK (in java) was required, to interact with the specific modules to sign and emit certificates.

4.2.6 *Stage VI - information correlator*

This final stage was not implemented, however it is extremely important. As already mentioned, collect information and make it available is not enough, it is important to find usefulness in the information and show it wisely. Stage IV already explained how this project achieved it. But the usefulness of the information can be much more explored.

At the introduction of the present dissertation, it was referred a duality between vertical and horizontal solutions. This stage intends to clarify this duality. To do so, lets assume two different scenarios:

- scenario 1 - a company implements a system to measure the water and trash level in the sewage lids to avoid floods. Another company implements a system to predict the rain probability. In a certain day, a citizen needs to go from A to B, and there are available 2 routes, 1 and 2, but route 1 is shorter and faster. Assuming it is a rainy day, the company will generate the alert for the citizens, and assume also that route 1 has one sewage lid full and is flooding. As these are two distinct companies, the end-user will never be advised to follow route 2;
- scenario 2 - the city implements both the above systems, and the low-level data of each sensor goes through the Edge. As an output, because the same system knows that it is expected rain and that a specific sewage lid is flooding, it can advise the end-user to follow route 2.

With this example, it becomes clear the advantage of an horizontal platform. Metaphorically, it is much more useful a Smart City of Things than a City of Smart Things.

However, the integration of different information is complex to achieve, mostly because it is not obvious. Although this stage is not implemented yet, the reason why it is here specified in the "Implementation" section, is because it is one of the outputs expected from an Edge that aims to achieve the full horizontality.

Furthermore, there is one more advantage on existing an architecture that evaluates every variable. From a business model point of view, and remembering that a Smart City is for the citizens through the City Council, it is very simplifier for the City Council if they only have to deal with one entity, instead of many companies with vertical solutions.

Concerning now to the implementation of this stage, this integration can be performed by two different ways. "Manually", where the developer has to think which variables should be integrated, and how to integrate them. Or via algorithms of Big Data and Machine Learning, from where even unexpected results can come out.

4.3 SYSTEM ARCHITECTURE

The above stages demystified how the Edge handles an huge amount of low-level data, some redundant, and periodically builds a single structured piece of useful information (per variable). Furthermore, it shows how this information is made available for the Cloud. This way, if (for example) a Google Service at the Cloud wants to implement some machine learning algorithms about temperature in a city, it has only to perform one query to the Edge, and it is answered with a matrix (or map) with the values from a specific time.

Understood the detailed implementation, this section intends to show how all of those stages can be executed in a machine, in the Edge. For that, it will be explained separately how to achieve a ″maximun″ performance with high scalability.

4.3.1 *Performance Oriented Edge*

Although performance was already discussed on above topics (performance of individual modules), this subsection addresses the performance of the integration of all modules.

Understood the complexity and dynamics behind a Smart City, it is reasonable to define, a priori, that each stage should be implemented in an individual machine, to optimize the performance of each one. Although for each stage it would be true, for the global system work flow, it would not, because the communication between them would be limited to the LAN connection speed (typically Gigabit, ideally 128Mb/s). Furthermore, assuming that following this implementation the Edge system could handle 100 thousand devices (which is a reasonable high number), this solution would not be scalable.

As an alternative, if all the modules are implemented in a single machine, the performance of each one individually will be worse (relatively), but the communication between stages is made by memory access, which is much faster (up to 2Gb/s, with M.2 Pci Express SSD disks). However, this implementation is less expensive (only one machine) and is expected to deal with about 10 thousand devices (because the machine can execute all the modules in parallel). Nevertheless, this implementation brings an extremely valuable consequence - scalability. This solution allows to deploy more machines and define, for example, that each one is responsible for a quadrant of the city. At the end, this ecosystem of Edges can handle with a ″theoretical″ infinite number of devices, at the price of deploying more Edges.

4.3.2 *Scalability Oriented Edge*

About this topic, it was already explained why scalability is a requisite. However, the solution to achieve it is not obvious. For a system to be considered scalable, it is necessary three distinct cares:

- every technology implemented individually must be scalable;
- the deployment of a new entity should be easy;
- a mechanism to distribute the computational load across the various machines.

About the first item, every selected technology was chosen ensuring it is scalable. The only topic where it is not evident, is security. For the tokens, it is scalable for sure, since it does not need to store the token. For the certificates, it is not evident, because every certificate needs to be stored by ADSS (however, the company claims it is).

As for the second item, ease of lifting a new machine is crucial. As the project evolves, more modules are implemented (with more technologies) and more complex the solution gets. So, to install a new machine, the task can get confuse and time-consuming, which is not desirable. To overcome this, three solutions were considered:

- solution 1 - operating system image with all the setup performed;
- solution 2 - user virtual machines for every module;
- solution 3 - use containers for every module;

About solution one, effectively it solves the problem, but it has many disadvantages as: tends to get deprecated (newer OS versions will come, and the image will not follow); it does not offer OS flexibility (oblige a specific OS); and does not allow a pre-setup (for example, different machines can use different ports for the same service). So, solution one works but is not ideal.

Solutions two and three, are discussed together, because they offer similar advantages, as: independency from the OS; private network for modules (improves the security of the system); and minimal pre-setup. Based on these points, the decision seems to be indifferent. However, the way these two technologies are implemented is very different. Figure 17 shows the main difference between them.

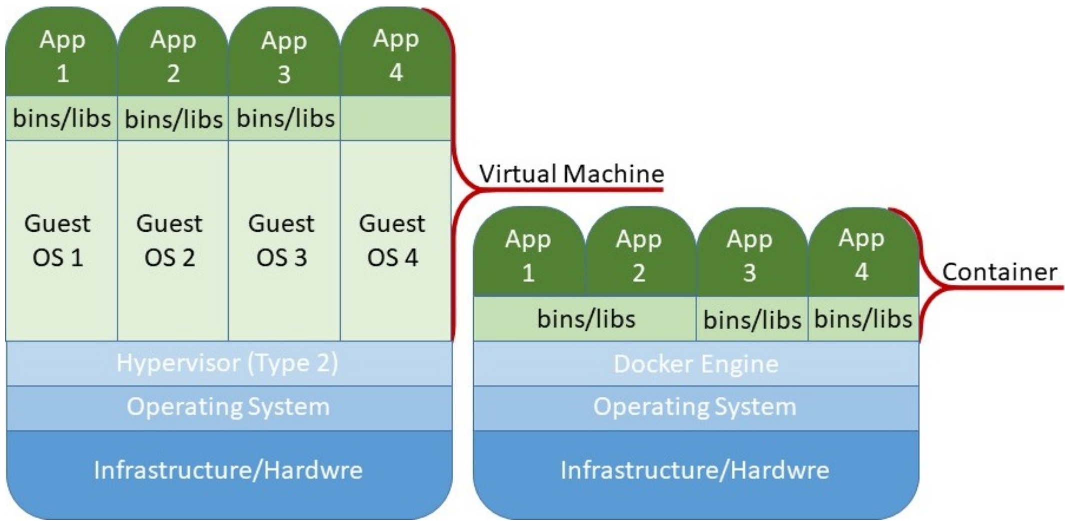


Figure 17: Virtual machine and container architecture.

While the virtual machine has a kernal and the whole OS for each VM, containers share the kernel with the host, eliminating the need to implement the OS. This explains why VMs are much more computationally heavy than containers. If a regular computer launches 20

VMs doing nothing it will probably freeze, but launching 20 containers doing nothing, the computer runs fluently. As in the Edge implementation, each module does not need any particular or distinct OS, containers are a preferred solution. Furthermore, the Docker Engine (a container implementation) allows the creation of a file (file.yaml) that specifies more settings that can be done regarding each container, for example:

- exposed ports;
- auto-restarting;
- define access restrictions;
- define host names for each container inside the private network;
- share data volumes between containers;
- launching order;
- define fixed versions of technologies;

These are just a few options that can be set in the yaml file, that have proved useful in designing the architecture. Furthermore, the possibility of creating a private network inside the host machine, is a very relevant feature, because the modules do not get "opened" directly to the Internet. It improves the systems security (which is different from the communication security mentioned in the dissertation).

With the creation of the docker file, putting up a new Edge only implies to install Docker in the computer (available for many OS), edit some pre-setup on the yaml file if desired, and execute one single command line instruction:

```
sudo docker-compuse up -d
```

To synthesize the above comparison between the three solutions, see table 7

Given that the global architecture should be implemented over containers, it was necessary to verify that each technology in use was available as a container image (all are, with the exception of ADSS). The creation of the container image with the ADSS ready-to-go was by itself a hard work, implying a detailed study of how the system works and how to build container images.

Finally, the third item, the mechanism responsible to load the balance across Edges. This third item was not implemented due to time constraints, but all the developed work aimed it. But in synthesis, the development of such system is based on the implementation of a load balancing concept as HAProxy, working as Round Robin. This technology offers three main advantages for the Edge systems:

	Disk Image	Virtual Machines	Containers
Pre-Setup	No	Yes	Yes
Detailed-Setup	No	No	Yes
OS independent	No	Yes	Yes
Implement Private Network	No	Yes	Yes
Performance	Highest	Lowest	Middle
Deprecatable	Yes	Moderate	No
Ease of Implementation	Lowest	Moderate	Highest
Security	Lower	Average	Average

Table 7: Comparison between Operating System disk image, Virtual Machines and Containers.

- This system can ”track” the load of each Edge, delivering new incoming messages to the less loaded one;
- As all the traffic passes through the HAProxy, it allows to have only one port opened to the Internet, decreasing the possibility of attacks (as a single port is easy to track);
- As all the traffic passes through the HAProxy, it also allows to perform a better security classification. Until now, checking the security of devices, happens restrictively through the technologies that the MQTT offers (CRL and tokens). This way, it can be implemented manually different checks (for example OCSP, which for many users can deliver much more performance than CRL).

4.3.3 *Final Architecture*

As the global architecture is vast and complex, this subsection simply intends to schematize the final architecture in figure 18. In it, each box is a container that works for one of the stages detailed in section 4.2.

4.4 EXPECTED OUTCOMES

As outcomes, this dissertation tries to prove some distinct points:

- 1 - prove that security is primordial when conceiving a Smart City scenario (this point is proven in the next chapter). Furthermore, that the apology of classifying instead of filtering is a must;
- 2 - develop a stable, scalable and high-performance Edge, oriented to Smart City and Cloud-friendly;

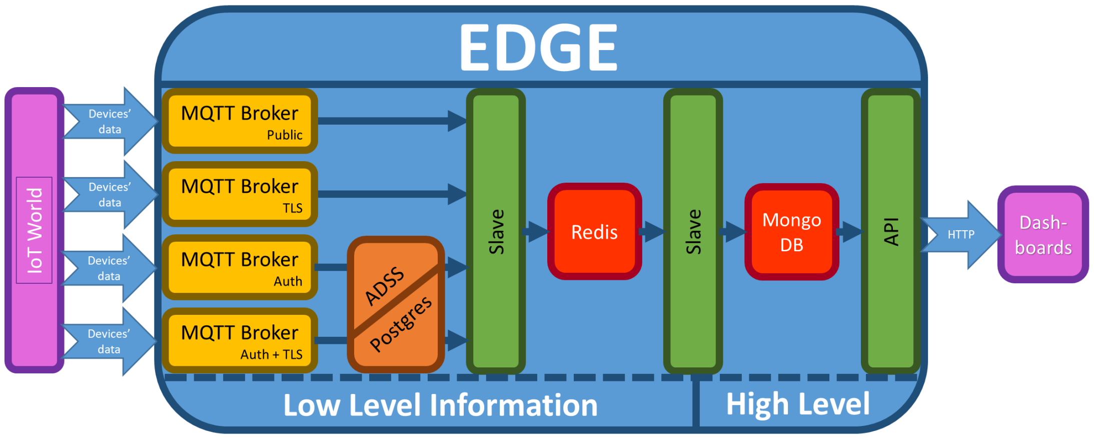


Figure 18: Conceptual Edge Design.

- 3 - prove that the way the information is showed is very valuable for the end-user. In addition, demonstrate that the capability to correlate different information brings usefulness;
- 4 - study the performance of a PKI (ADSS) within the field of Smart City, and show that it delivers the best "security vs effort" efficiency;
- 5 - test that this Edge architecture represents a good "cost-number of devices" relation, and that it can handle, for a few years, the Smart City devices;
- 6 - prove that the ability to allow different experienced people in the areas to implement the algorithm to deal with each variable is a must.

5 CASE STUDIES / EXPERIMENTS

This chapter intends to explain the detailed steps performed and some metrics to evaluate the project output, the Edge. Furthermore, a reflection about the results is presented, as well as a brief discussion about global topics of the Smart City.

5.1 EXPERIMENTAL SETUP

As mentioned in the section 4.4 of this dissertation, many of the expected results are related to the Edge performance. However, to "measure" it the reality is not suitable, as an IoT is not yet so disposable as desired. But, regarding reality, it is preferable to implement first the support infrastructure for Smart City and secondly the IoTs, than otherwise. Thus, in order to test the developed Edge, a simulator was used.

As the simulators available in the Internet all implement closed systems (meaning no information could go out, to the Edge), and these were mostly dedicated to a specific service (for example, road traffic), it justified the creation of a new one, from scratch. The new simulator is intended to simulate the Smart City scenario, where thousands of devices of many kind (like static or mobile) can co-exist, many variables represent the reality (like temperature, air quality, etc) and a city model gives reality to the whole system.

As this simulator was also a complex project whose development is not directly linked to the Edge, its detailed implementation is not explained here. Thus, a brief overview of what it intends to achieve is detailed bellow and in the next subsections:

- more than simulate, emulate devices. It is expected that each device opens its own MQTT connection, has its own authentication credentials and implements its own virtual sensors;
- over a city model, it is expected to see the devices moving around, some static devices, and different shaped devices, that can represent several kind of entities (wearables, cars, phones, ambulances, etc.);
- visually distinguish devices by its security level (through colors);

- capability to simulate over one thousand devices, each one sending messages regularly (every 10 seconds, for example);
- each device must be capable of georeferencing relative to a common axis system;
- as near devices must read similar values of a determined variable, it is needed to create common-services, where each device requests a value of variable (simulate devices sensors);
- as many services must be implemented, there must be a "reality controller", to generate external events (like fires, accidents, etc.);
- as a new generation web-based simulator, it is expected the development of a template to allow anyone anywhere, to implement a new service, and all the devices will "read" values to that new service and feed the Edge with it (this topic is particularly important because it allows the study of new and more precise models and behaviors to "tech" the Edge);
- finally, although this simulator is developed to test Edge performance, it must be easily re-configurable to feed any other architecture, allowing to test another solutions for the Smart City.

To explain how this simulator was achieved, its explanation is separated into three modules: devices emulation; city simulation; and reality-services simulation.

5.1.1 *Devices Emulation*

The main goal of this module is to test the Edge performance, by optimizing the number of emulated devices. About this topic, it is important to emphasize that each device implements a real thing of the IoT, with a dedicated connection and attributes. Thus, from the point of view of the Edge, each device behaves like a real-world IoT thing (one authentication, one ID, own sensors, own position and own tracking option).

In summary, a device is an asynchronous class (implemented in python), autonomous, that can be instantiated anywhere in the world. The developed template allow the user to decide:

- the shape and color of the device;
- the type of device, between mobile and static;
- if mobile, the speed at which it moves;
- security level, among public, TLS, TLS + token and TLS + certificate;

- a url, to where it sends its information (Edge, in this context, or any other infrastructure);
- a list of urls from where it reads values (simulating sensors);
- the periodicity with which it reads and sends data;
- and the desire to be tracked by ID or by ID + GPS.

The greatest advantage of use URLs to specify the source and destination of data, is that this same template can be implemented in any other programming language (or even architecture). This allows that even someone who wants to try its own algorithms, can program its own devices and they will be surely possible to integrate in the simulator. It is available a template script in Python, that implements all the background tasks required for the device to exist, and offers a "write your code here" section, where the user just defines the loop time, which variables it sends, authentication mechanisms, among other settings.

As an example (and taking advantage of the freedom of URLs), aiming the interaction with the real end-user, a web-client was developed. This way the users, with a browser in the smartphone or computer or tablet (Google Chrome is advisable), can be part of the simulator through LabSecIoT site[1]. Users can choose and change the color, the shape, the speed, authentication method (only public, TLS and TLS + token we implemented in the HTML client) and even decide to send variable readings to the Edge. A particular attractive interaction is that when the Noise service is enabled, the noise the device will send to the Edge is the noise perceived by the devices microphone. This web page is designed in HTML, and all the algorithms and connections are programmed in JavaScript. The interface offered to the user can be seen in figure 19.

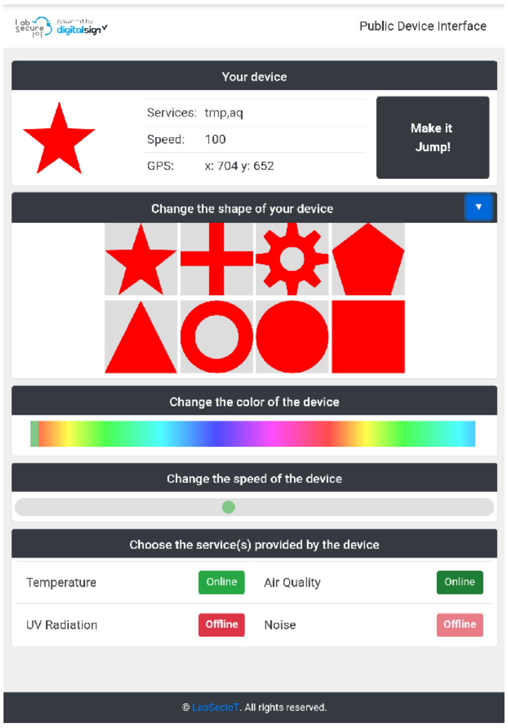
5.1.2 *City Simulation*

This module had a whole different purpose. While the previous module had a functional purpose (test Edge performance), this one intends mostly to be attractive and give a general overview of what a Smart City populated with IoTs can look like.

The greatest feature of this module is the integration. As it runs entirely online, anyone who accesses it can see all the devices that are instantiated anywhere in the world, regardless of the programming language or the architecture. Furthermore, it illustrates in an attractive and effective way the different level of security. Figure 20 shows how different types of devices enhance their security level.

The city background was drawn in svg (scalable vector graphics), to fit any screen resolution. To promote the beauty and attractiveness of the simulator, a JavaScript library (GSAP)

1 https://labseciot.dsi.uminho.pt:4568/static/new_device.html


Public Device Interface
Your device
Make it Jump!
Change the shape of your device
Change the color of the device
Change the speed of the device
Choose the service(s) provided by the device
© LabSecIoT. All rights reserved.

Services: tmp,aq
Speed: 100
GPS: x: 704 y: 652

Temperature	Online	Air Quality	Online
UV Radiation	Offline	Noise	Offline

Figure 19: User interface for adding devices.

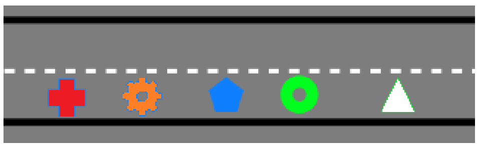


Figure 20: Example of different types of devices with the available security levels, where: red - public; orange - encrypted over TLS; blue - encrypted over TLS and authentic with token; green - encrypted over TLS and authentic with x509 certificate; and white - devices which are not communicating with the Edge or desired to keep their information in private.

was used. This engine can animate any CSS (Cascade Style Sheets) property (shape, color, border-color, position, animations, etc.) from HTML in a relatively easy way, allowing the devices to have a smooth movement.

Again, this simulator runs online, and anyone through LabSecIoT site[2] can see all of this information flowing in real time. The aspect of the simulator can be seen in figure 21.

2 https://labseciot.dsi.uminho.pt:4568/static/map.html

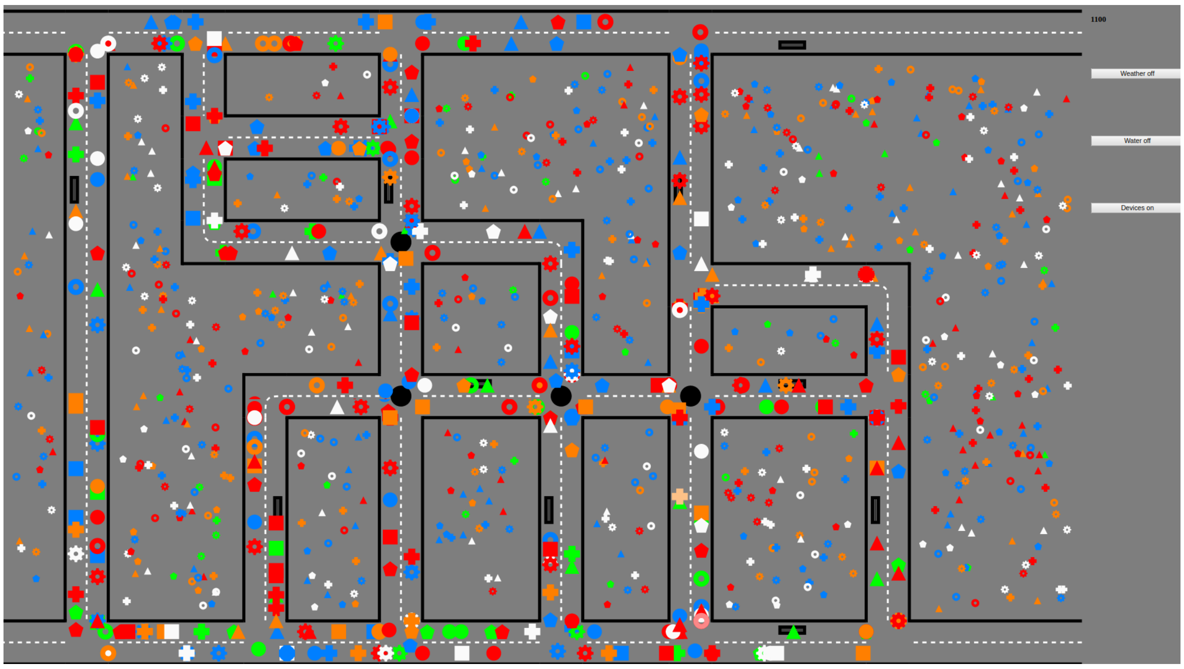


Figure 21: Smart City simulator.

5.1.3 *Reality-Services Simulation*

Finally, the third module has the purpose of simulating real variables (services) and allow the generation of unusual events. In this context, by service it is understood a program that generates information (from any source, and may even be from devices on the simulator), builds a map/matrix (or other representation of a phenomenon), and makes it available by implementing an HTTP server that receives coordinates and responds with a value associated with those coordinates. As a start point, the next 5 distinct services were implemented:

- temperature - service that every second within one day, generate a map with an average temperature between 10 ºC and 30 ºC (described by a sine). Also, there are always two bobbles, one with more 5 ºC and the other with less 5 ºC that moves clockwise. Finally, the whole map is smoothed and granulated to simulate minimal temperature differences. As an external event, a mouse click triggers a fire around the click which is gradually erased for half a day;

- noise - service that generates a map with the noise of every device in the city. Each mobile device generates a noise value proportional to its traveling speed and each static device generates a random static value of noise. As a trigger, the mouse click generates an accident, which noise takes 2 seconds to disappear;

- air quality - similar to noise, but only mobile devices generate pollution and it takes long to disappear. This service has no triggers;

- ultraviolet radiation - service that between the 11 hours and 15 hours has its max value and between 20 hours and 8 hours is minimum. This service also has no triggers;
- sewerage - this service implements ten sewage lids and the paths connecting them. The idea is to control the pollution level on each one and control the rain quantity. If a sewage path is very polluted and it is raining, then there will be a flood on the correspondent lid. The paths are designed so that its caudal is such that can handle with the input rain plus the max caudal from the source path, recursively. This allows to assign different rigor level to each path. For example, the final path (that collects water from every other paths) should be very critical and its pollution value should never exceed 50% (for example).

To add some reality to the system, a minimal interaction was implemented. So, when it is raining the temperature drops, fires are erased faster and the UV level also decreases. This integration intends to allow the Edge to correlate variables, and see more useful information and warnings from it. Furthermore, it was implemented the possibility of choosing the duration of a day in the simulator between a day, an hour or a minute in the reality (this allows to speed up the tests). Some images of the services running can be seen in figure 22.

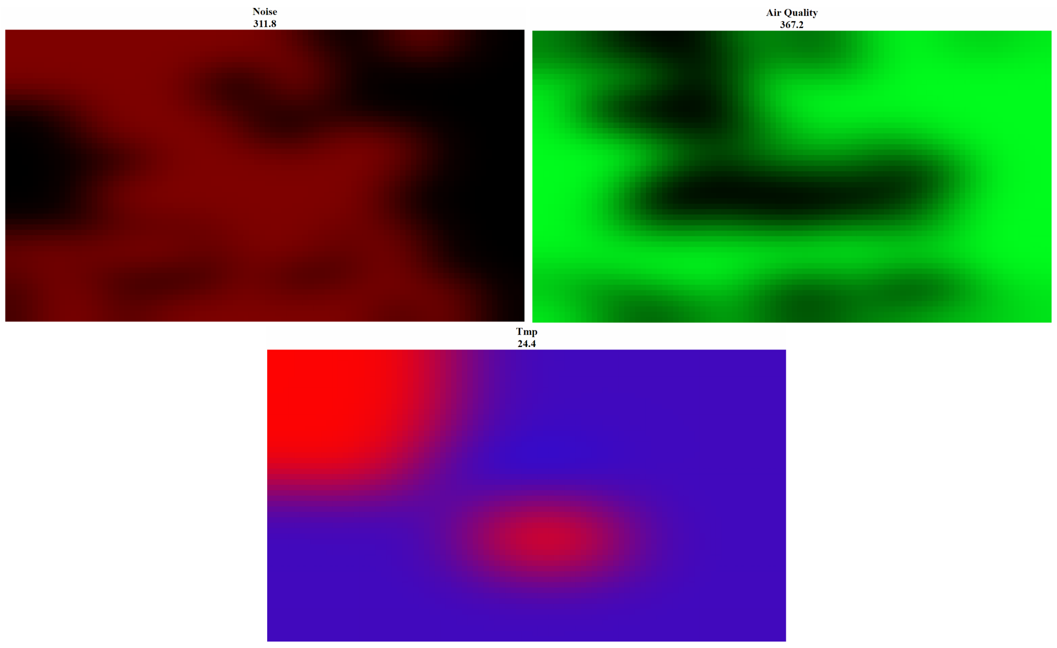


Figure 22: These are three example of the services running. The city map is populated with about 500 devices, and the noise map shows the existence of these, as well as the air quality. The noise map is more "polluted" than the air quality because both static and mobile devices generate noise, while air pollution only mobile devices produce it. The bottom service is the temperature one, where a fire was generated by clicking in the map.

Another outcome expected from this module of the simulator, is again the availability. As it works online and is modular, it is expected that anyone with the template can implement a new service. This template is basically a Python script that implements the HTTP server, and with a "write your code here" section, where the user writes the algorithm that the service executes. The rigorousness of the module that generates the maps is very important, because it will allow to perform much better tests with the Edge dashboards. For example, the variable temperature (the sine with two circulating bubbles) is obviously unreal. This module allows others to work on more accurate modules, while helping to test the Edge.

This minimal description of the service was provided because it will be useful to understand the dashboards, discussed in the next section, "Results".

5.2 RESULTS

To evaluate the results from this project, three aspects will be analyzed:

- Edge performance - number of devices it can keep connected and capability to output high level information in real time;
- Information usefulness - compare the traditional ways of displaying information with the ones outputted by the dashboard. Furthermore, prove that integrate different variables can result in useful unexpected information;
- Real case - results from real IoTs.

5.2.1 *Edge Performance*

About first topic it is important to specify the relevant characteristics of the Edge machine used:

- Intel Core i5 8600k, 6 cores at 3.8 GHz;
- 16 GB of RAM DDR4 at 2100 GHz;
- 256 GB SSD.

Also important to know, is the rate of update and dataset of each variable detailed above and synthesized in table 8:

- temperature - values on Redis last about 25 seconds, and every 5 seconds a new matrix of 32x18 is created and stored on MongoDB;

- noise - values on Redis last about 2 seconds, and every second a new matrix of 32x18 is created and stored on MongoDB;
- air quality - values on Redis last about 10 seconds, and every 5 seconds a new matrix of 32x18 is created and stored on MongoDB;
- ultraviolet radiation - values on Redis last about 30 seconds, and every 10 seconds a new matrix of 32x18 is created and stored on MongoDB;
- sewerage - values on Redis last about 5 seconds, and every 2 seconds the information of the 10 sewage lids is updated on MongoDB. This information is not stored in form of matrix, but in JSON form (one JSON by lid);
- device tracking - for those devices who decide to do it, values on Redis last the specified time by each device, and every 3 seconds the information of the devices is updated to MongoDB. This information is not stored in form of matrix, but in JSON form (one JSON by IoT).

Variable	Duration in Redis (s)	Data Set	Updated to MongoDB (s)
Temperature	25	matrix 32x18	5
Noise	2	matrix 32x18	1
Air Quality	10	matrix 32x18	5
UV Radiation	30	matrix 32x18	10
Sewerage	5	JSON	2
Device Tracking	variable	JSON	3

Table 8: Preset used to test Edge performance.

To "measure" the Edge performance, the CPU and RAM usage for a specific number of devices is not a relevant metric. When dealing with 500 devices, if machine A is at 10% and machine B is at 20%, it can not be concluded that A is better than B. If the output expected from the Edge is the highest number of devices, than the best metric to measure its performance is indeed the maximum number of devices and the capacity of output the respective high level information. This can be measured when the Edge starts processing the devices with delay, which means that it is buffering connections.

At a first stage, the whole Edge was set as a regular server, working synchronously over HTTP. The number of devices at which it started delaying was about 400 devices. The first improvement was the application protocol, which became MQTT. Still synchronously, the maximum number of devices was about 800. With the huge update from synchronous to asynchronous on every python script on the Edge, the maximum number of devices increased drastically to 6000 devices (this proves that for a massive usage system, like Smart

City, asynchronous paradigm is much more suitable). In an attempt to explain this limit, the usage of the computer and the usage of the network were analyzed, and it was verified that the limiting factor was the network. So, the simulator visualization was disabled (each device makes one less publish) and it was tested again. The limit was about 7000 devices. Although the network usage was again at its maximum, the Edge usage was about 80%. This means that, limited or not by the network, the Edge is near its maximum capacity. However, it is important to highlight that these results come from an over-rated preset (it makes no sense to evaluate every variable with a rate of 1 to 10 seconds). So, as a qualitative result, one Edge is expected to deal with all the devices from a city for the next few years, since it is not expected that such number of IoTs will be deployed. To synthesize the quantitative results, see table 9.

	Programming Paradigm	Application Protocol	Simulator View	Max Devices	Reason
1	Synchronous	HTTP	Enabled	400	time wasted on ″waitings″
2	Synchronous	MQTT	Enabled	800	time wasted on ″waitings″
3	Asynchronous	MQTT	Enabled	6000	network usage and possibly Edge usage
4	Asynchronous	MQTT	Disabled	7000	network usage and possibly Edge usage

Table 9: Tested scenarios for Edge performance.

One more unexpected result, now concerning devices security, it was unnoticeable the difference between 6000 public devices and 6000 authenticated by certificate devices. This means that the challenge faced by the Smart City is not tracking the security level, but how to deliver and monitor it. In detail, the problem will be insertion of the certificate in the production phase of the IoT, and the renewal of the certificate when it revokes (it must be a practical solution).

Finally, one more result (or curiosity) about performance, with the simulator running for about 15 days, Mongo stored about 50 GB of data, an average of 432000 matrices by variable. Doing the math, this is the expected, which means that no delays occurred.

5.2.2 *Information Usefulness*

Although there are no metrics to evaluate the usefulness of the information, it can be compared to other solutions. So, lets assume the next examples:

- through IPMA (Portuguese Institute for Sea and Atmosphere) site[3], a service implemented in Portugal that measures and predicts the weather and sea forecasts. To do so, it provides maps, with colored regions and tables with more specific information;
- through ViaMichelin site[4], it can be seen, statically, the transit conditions in the most common routes, where the color of the route indicates the traffic density;
- not available online, but private parks have internally a map of the available and occupied places;
- TUB (Urban Transports of Braga) is currently working on a platform to allow users to see in real time the bus they are waiting over the city map.

As these examples, there are many more. The expected result of this project (about this topic) is to prove that having all this information on the same platform, more dynamic and integrated, it is more advantageous. To promote this, the developed dashboard, fed with the Edge output from the simulator, has the next features:

- integrates the five services from the simulator (figure 24);
- divides the city map in small blocks (figure 25);
- to visualize a variable, the same view has in half of the screen the high level information backgrounded with the city map, in a quadrant the graphic with the average of the variable for the whole city (for example, the average of the temperature for the whole city), and in the last quadrant a similar graphic but only with information from the block or set of blocks selected by the user (figure 25);
- mousing over a block, it pops-up the instant variable value for that block, and the security level from which it was calculated by the Edge (figure 25);
- instant notification for drastic variations and dangerous values;
- an advisement area, where general information/guidelines are displayed for the common citizen, like ”The UV index is high for children, consider use sunscreen or avoid direct sun contact.” (figure 23);
- a dedicated view where all the online and offline devices can be tracked (figure 26);
- auto-generate periodic reports for storing and meeting purposes;

3 https://www.ipma.pt/pt/index.html
4 https://www.viamichelin.pt/web/Trafego

- finally, a global map view where the city blocks are represented, and mousing over them shows all the available information about every available topic in the Edge for that block, in real time.

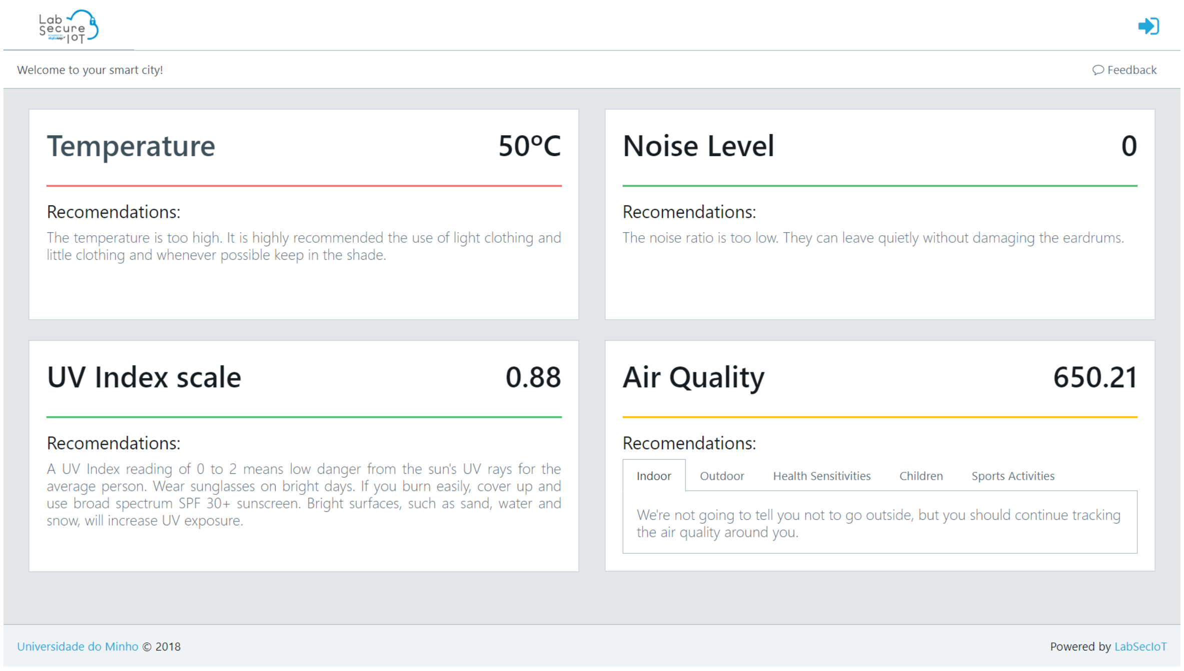
The whole interaction with the dashboard is very intuitive and it was designed with beauty, attractive and recent frameworks. Some of its output can be seen in figures 23, 24, 25 and 26. It is important to emphasize that a dashboard is just a portal to look "through" the Edge content. Different dashboards can show the same information, in different ways, for different purposes, and conclude different things.

To compare the usefulness of this integrated dashboard with the common-known ones, it was presented to an audience and some feedbacks were collected. In general, the audience was amazed by the idea. For example, the Director of the Intelligent Systems Department of Guimarães County stated "I need this for my city.". Although after the development of the dashboards this result was evident, when the project started it was not, being extremely important to prove it. However, for a solid proof, a bigger sample (audience) is required.

5.2.3 *Real Case*

Regardless of the results, the most important one is the real case, because the Edge performance is irrelevant if real IoTs can not be used. With all the architecture implemented and running, some devices were developed. Within this project, the developed IoTs were equipped with:

- Arduino, atmega328p @16MHz or esp8266, nodemcuv2;
- GSM module, sim800l;
- GPS module, neo-6m;
- Adjustable power converter;
- Temperature and humidity sensor, dht22;
- General air quality sensor, MQ135;
- Carbon monoxide sensor, MQ7;
- Noise sensor, KY-038;
- Waterproof power-in jack;
- Circuit board;
- Waterproof case.



Welcome to your smart city! Feedback

Temperature 50°C

Recomendations:

The temperature is too high. It is highly recommended the use of light clothing and little clothing and whenever possible keep in the shade.

Noise Level 0

Recomendations:

The noise ratio is too low. They can leave quietly without damaging the eardrums.

UV Index scale 0.88

Recomendations:

A UV Index reading of 0 to 2 means low danger from the sun's UV rays for the average person. Wear sunglasses on bright days. If you burn easily, cover up and use broad spectrum SPF 30+ sunscreen. Bright surfaces, such as sand, water and snow, will increase UV exposure.

Air Quality 650.21

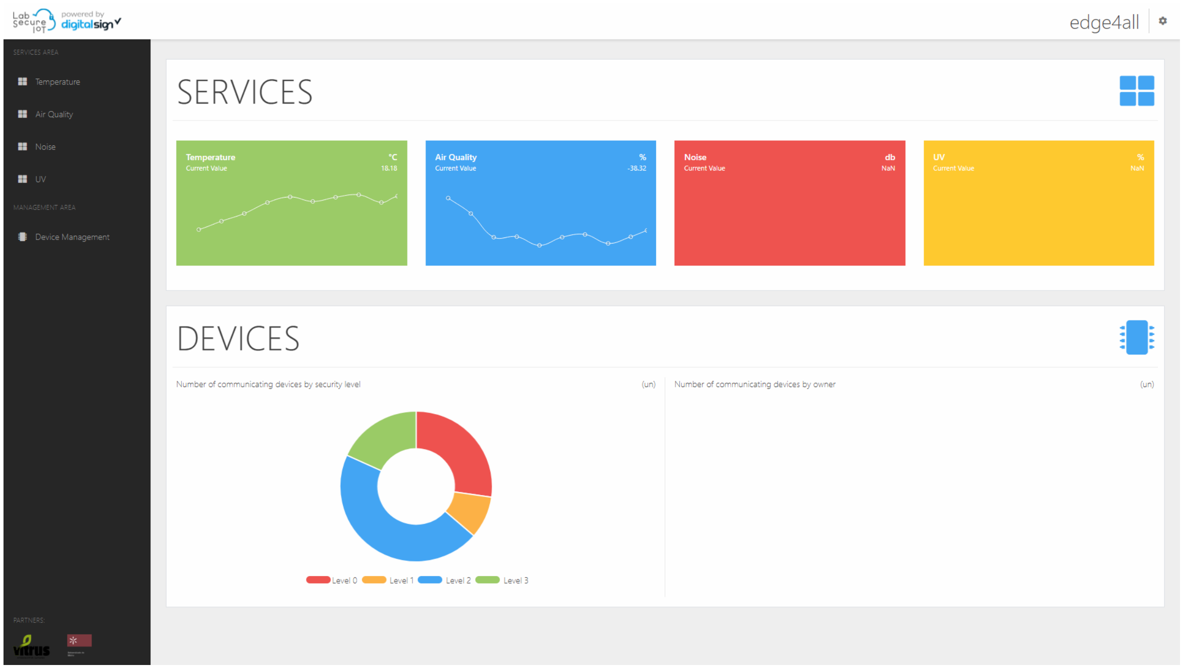
Recomendations:

Indoor | Outdoor | Health Sensitivities | Children | Sports Activities

We're not going to tell you not to go outside, but you should continue tracking the air quality around you.

Universidade do Minho © 2018 Powered by LabSecIoT

Figure 23: Dashboard – Home page, where some global information is displayed, as well as some warnings and recommendations.



edge4all

SERVICES AREA: Temperature, Air Quality, Noise, UV

MANAGEMENT AREA: Device Management

PARTNERS

SERVICES

Temperature	Current Value	°C 18.18
Air Quality	Current Value	% -38.32
Noise	Current Value	db NaN
UV	Current Value	% NaN

DEVICES

Number of communicating devices by security level (un)

Number of communicating devices by owner (un)

Figure 24: Dashboard – services page, where the user can see a global history of each subscribed service, some warnings, some status about devices, log in or navigate to services or devices.

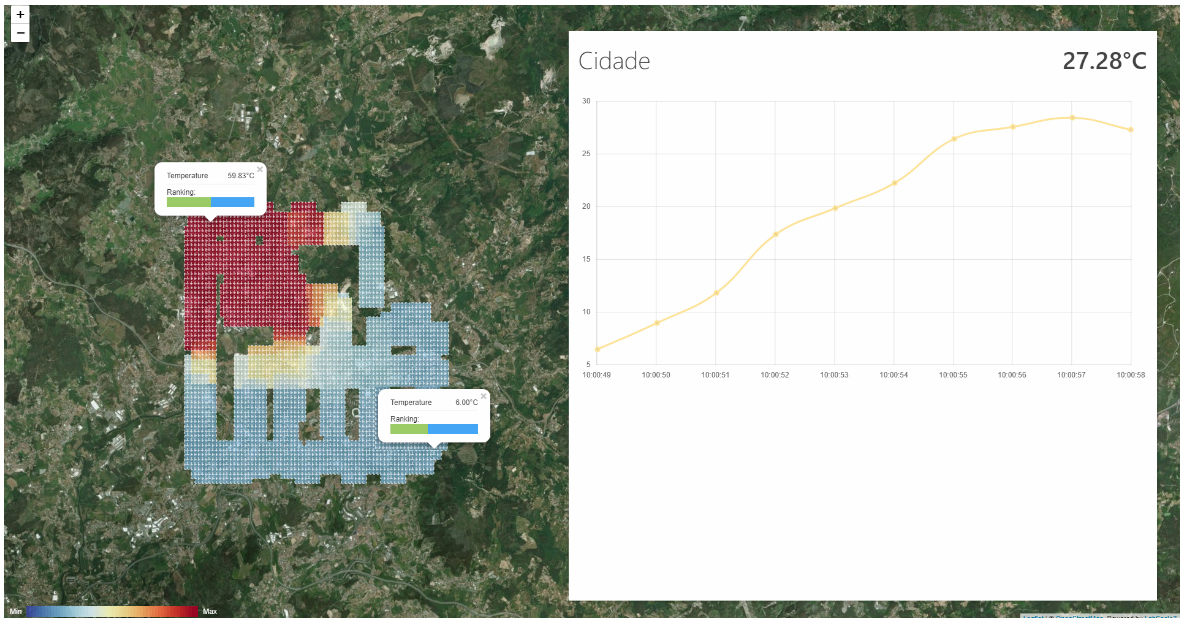
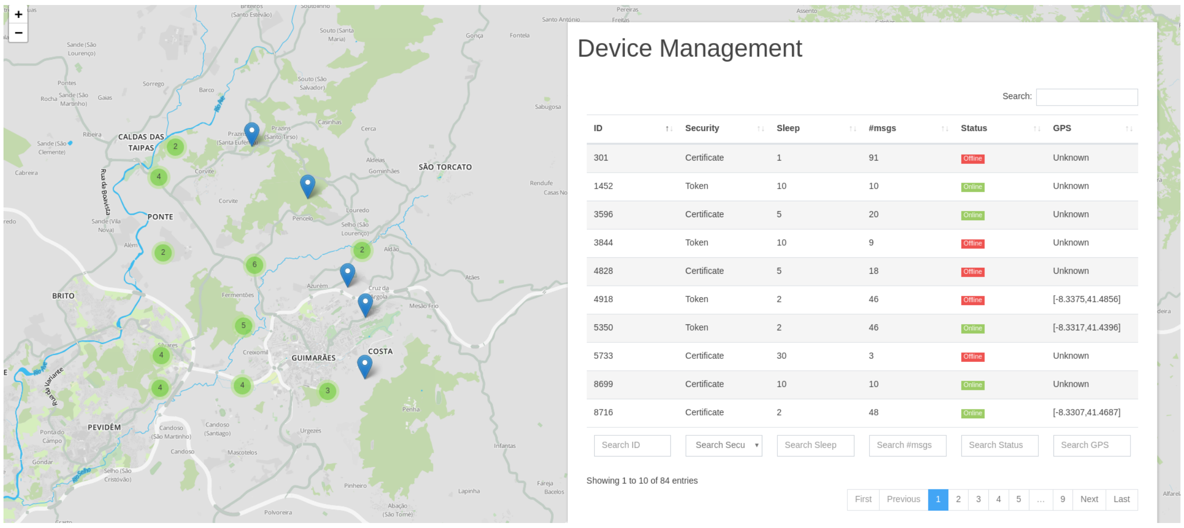


Figure 25: Dashboard – specific service page (temperature in this case), where on the right can be seen the history of the temperature and the current value; on the left it can be seen the temperature map with a specified resolution from settings (each square con have 100 meters, 1 kilometre or 10 kilometres). Mousing over each square, can be seen the specific temperature of it as well as the "confidence" of that specific value (based on the devices security level). In this specific image, it can be seen that a fire is happening at the upper left corner.



Device Management

Search:

ID	Security	Sleep	#msgs	Status	GPS
301	Certificate	1	91	Offline	Unknown
1452	Token	10	10	Online	Unknown
3596	Certificate	5	20	Online	Unknown
3844	Token	10	9	Offline	Unknown
4828	Certificate	5	18	Offline	Unknown
4918	Token	2	46	Offline	[-8.3375,41.4856]
5350	Token	2	46	Online	[-8.3317,41.4396]
5733	Certificate	30	3	Offline	Unknown
8699	Certificate	10	10	Online	Unknown
8716	Certificate	2	48	Online	[-8.3307,41.4687]
Search ID	Search Secu	Search Sleep	Search #msgs	Search Status	Search GPS

Showing 1 to 10 of 84 entries

First Previous 1 2 3 4 5 … 9 Next Last

Figure 26: Dashboard – device management page, where the devices that wished to be tracked are shown, as well as some information about it. If can be seen whether the device is online or offline, the number of messages that it has already sent to the Edge, its last position, and some more information. This table has some advanced filters that allows to find a specific device if desired.

About the last two topics, the circuit board was drawn with proper program and printed in the Minho University facilities. For the IoT case/box, a 3d printer was acquired and the design were all performed in a CAD online framework[5]. Figure 27 shows 2 of the developed IoTs. The baseline of the IoT (micrcontroller, GPS module, gsm modules, box and board) costed about 15 to 25 euros each (depending on the type and number of sensors used). Although the price per unit is not very high (mentioned by the council members), it is still far from being disposable. Furthermore, the negotiations with the telecommunication operators did not run as expected. Monthly costs offered by two major telecommunication operators ranged from 5 to 8 euros for an amount of data usage between 30 and 500MB. Obviously, these are not acceptable costs per IoT. Moreover, these costs already come from partnerships, not being the public-price. For this reason, only 8 IoTs were developed (figure 27).

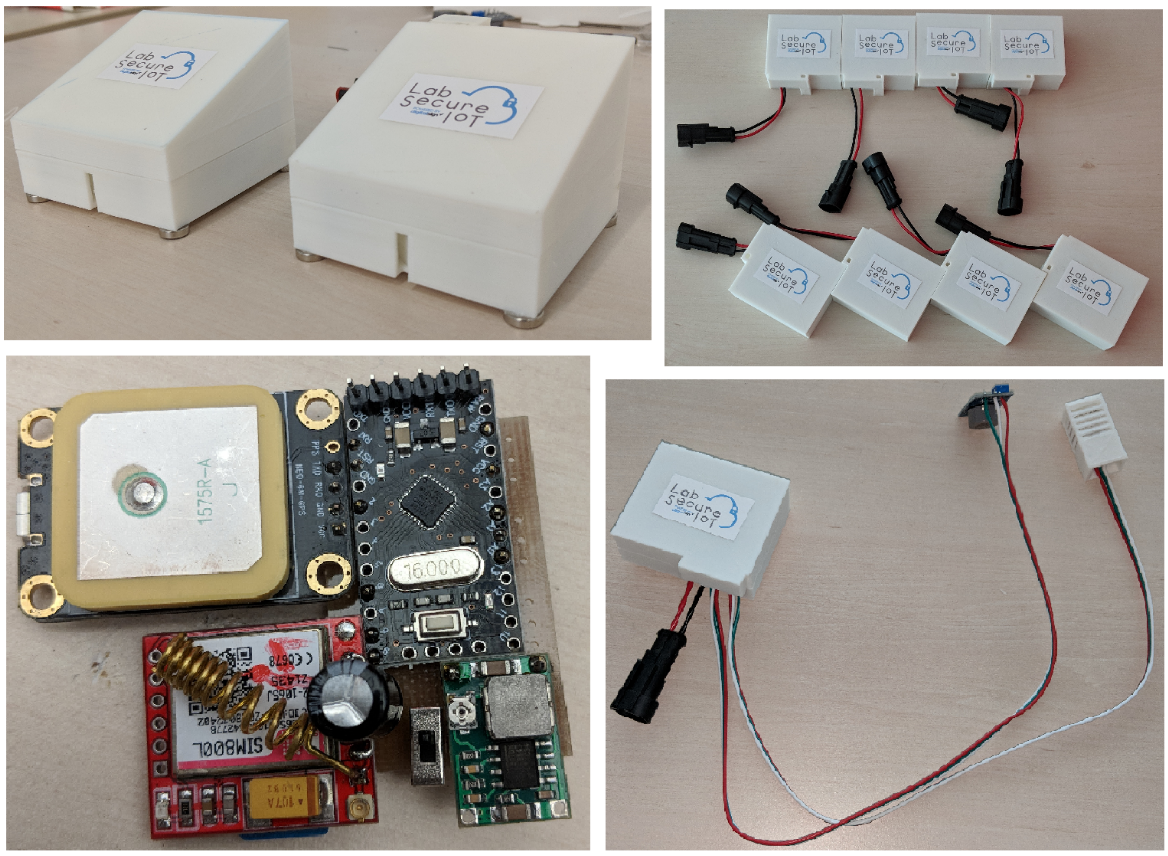


Figure 27: Developed Devices. The upper left image contains the first version of the box, and the upper right shows the second version (size and performance optimized). The bottom left image show the circuit (designed and soldered by the author also), and the bottom right image show the "final product", where two sensors are connected.

5 https://www.tinkercad.com/

As the idea behind this project is to build a common and integrated platform (an horizontal solution), the production of IoTs is not the main goal. The next step was to convince other companies, with vertical solutions, to join the system. This does not imply that they can not have their own platforms and ideas. It only means that their IoTs should send the collected data to the Edge, and then they have their information stored, in a trustable system. To the project, it means that it would be possible to correlate that information with data from devices of other companies. To move forward with this idea, NipsTechnologie was the first partnership. They have currently deployed 50 sensors in waste containers to track the waste level. They use this information to optimize the truck routes. When their sensors started "talking" with the Edge, they won an IoT tracking platform (to know whether their devices are online or not), a safer communication (authentic and reliable), and they could still ask the Edge for sensors output to optimize their routes. On the other hand, this project can offer to citizens and council members dashboards to see and correlate data, with some general warnings. Like this company, other two were invited, but unfortunately they refused.

The goal of the 8 IoTs is to collect the greatest amount of data from various areas. The idea is to deploy them in buses, as they travel through the city the whole day. For this purpose, TUB (in Braga) and Arriva (in guimarães) made available some of their vehicles as prototypes, in exchange of a real time tracking of their buses. Furthermore, as there are some vehicles with WiFi (and some spots in the city), some WiFi IoTs will be deployed too, as there are no monthly fees associated.

Finally, about all these real information, the dashboards are still under development, as the focus was the dashboards for the simulator. Although the differences are not many, some are required. For example, the coordinate system (with the simulator, the coordinates were refereed to a theoretical origin, in a map with 1920 x 1080 pixels, while in the reality the real latitude and longitude must be used) and the variables living time (in the simulator about 1 to 30 seconds, but in the reality longer intervals make more sense).

One last result (less important, but relevant), is that the company that powers the laboratory (DigitalSign), offered their data-center to host the final Edge. This way, there is one Edge hosted at Minho University, for the purpose of development and tests with the simulator, and there is a production Edge hosted at DigitalSign[6]. The good result that came from this was the implementation of the new Edge. All the architecture was designed to be portable and on-the-go (easy to implement), and this opportunity allowed to prove it, as the new Edge took about 30 minutes to be ready (from scratch, starting by instantiating the virtual machine that hosts the Edge), in a OS (in the data-center only CentOS was allowed) different from the one it was developed (Ubuntu).

6 edge4all-dev.digitalsign.pt:8000

5.3 SURVEY

One topic that is still unclear is whether citizens are predisposed/willing to live/contribute for a Smart City ecosystem. Although the importance of it is indisputable, whether citizens are willing to pay for it is not so evident. To clarify this doubt, a questionnaire was constructed and released in social networks (it was not asked to close friends to respond it to avoid biased answers). It was built with Google services, and is still online[7]. In summary, with 204 answers until 26/06/2018, approximately the same number of men and women, and almost 90% of the sample between 18 and 45 years old, it showed that:

- about 80% of the population wants the Smart City services;
- 93% of people want the dashboards in a mobile application, 33% in a website and 64% in public panels (this answer had multiple choice);
- 46% of people accept to pay a symbolic monthly fee to contribute for Smart Cities and from the 54% who do not, 30% would prefer to pay for the mobile application. This means that 37% of the population does not want to pay for the Smart City. However, only 20% of the sample does not want the Smart City services (from the first item), which means that 17% to 37% of the citizens consider that they have the right to live in a Smart City for free;
- finally, if system offering smart-city-points to citizens who practice good Smart City actions (like report damaged IoTs, recycle, travel by bus, etc.) and these points could be exchanged by promotions in the daily habits, 93% of the citizens would like it.

7 https://goo.gl/forms/ep81KpDrz1N7JnrA3

6 CONCLUSIONS

This chapter summarizes the encouraging results obtained, the achieved goals and some personal thoughts about the outcome. Finally, a brief reflection on examples of tasks that still need attention/development is presented.

6.1 SUMMARY AND DISCUSSION

Concluded the project, the results were quite satisfactory. From a scientific point of view, it is considered that all the research hypothesis were answered, and the proposed goals were achieved (and exceeded):

- the Edge architecture is solid and easily replicable;
- its performance could be tested;
- there are some real IoT devices deployed in the city;
- there is a production Edge up and running;
- the usefulness of the information was proven;
- and there are new ideas and students to keep/help the project moving forward.

After this vast study on Smart City, it is safe to state that the biggest obstacle it faces are the communications costs. Assuming that in the future there will be new business models for IoT communications, its costs will no longer be a problem. Over GSM, the only plausible scenario is if the acquisition of the SIM cards is a one-time purchase, instead of a monthly fee. Although this solution is useful, it is not the most advantageous, since the GSM module consumes a lot of energy. A futuristic (yet possible) solution would be the deployment of a Mesh network over the city. Even if the implementation of this solution is expensive, the decrease of the production price per IoT would justify the investment in a short/medium term. Furthermore, the deployment of a private network only for IoTs, would allow for a much better intrusion detection, improving drastically the security of the whole system. With this investment, an IoT device would not need the GSM module, no SIM card or data plan, and maybe the GPS module could be dispensed as well. This would drastically decrease the power consumption, and the price of an IoT would be less than half of the current one.

Although not everyone yet sees the advantages of a Smart City (from section 5.3), a reasonable amount of people already do, which strengthens the need to start developing and implementing the entire ecosystem, as this project intends too.

About the global proposed (Edge) architecture, as proved above, a single Edge (developed in this project) can deal with about 7000 devices, under exaggerated conditions forced by the simulator. Aside from all the Smart City controversies (overrated data plans, expensive IoTs, existence of vertical solutions and not everyone convinced/willing to pay for it), maybe a single Edge is enough for a city, or even a region, while Smart Cities are in the early ages.

About the simulator, although it is not the focus of the project, a personal/local opinion was collected. From observation, the simulator is clearly attractive, but its true goal is to show other entities what a populated Smart City looks like, and convince them that it is necessary. Throughout the project, some students and professors passed through the laboratory and saw the simulator running projected on the floor. As desired, it aroused their curiosity to know what it was. With the explanation, everyone looked amazed with the project and showed interest in seeing the concept implemented in a city. Besides these individual curiosities, the simulator was also used in the inauguration of the LabSecIoT. At this event, where "important" entities like board directors of the surrounding councils and companies were present, the simulator stood out once more. It helped not only to see the global picture, but also to show the importance of the security, as some attacks were simulated/performed. At the end of the meeting, a brief brainstorm about desired Smart City services took place, where ideas like smart parking and smart traffic lights (and more) were mentioned. This inauguration proved that developing a simulator is a must. As a result from the inauguration, LabSecIoT was mentioned on some local magazines, like "Laboratório de segurança em cidades inteligentes desafia câmaras municipais" (2017, December 12th), Correio do Minho[1] and "UMinho inaugura primeiro laboratório de segurança em smart cities para estudantes" (2017, December 8th), i9Magazine[2]. It was also invited to speak on TV (Porto Canal), as the first security and Smart City lab in the country.

Finally, not a technical question but as well important, to continue the hard work of making the companies aware of the importance of an horizontal solution, is required. Without this core requisite, all this project becomes objectively irrelevant (yet, scientifically and technologically it is). If this specific task continues, and if the project keeps having the visibility

1 https://www.correiodominho.pt/noticias/laboratorio-de-seguranca-em-cidades-inteligentes-desafia-camaras-municipais/106384
2 https://portal.i9magazine.pt/uminho-inaugura-primeiro-laboratorio-seguranca-smart-cities-estudantes/

and results it has had so far, a new business model may even emerge, possibly justifying the creation of a new company.

6.2 PROSPECT FOR FUTURE WORK

After the conclusion of this project, it is easily observable that it will never be concluded. As this project relies on the creativity and capability to generate more and more useful information, the limits are unreachable. However, in a near future, some tasks are sill required.

For example, deliver certificates to the end-user. Currently, the user authenticates itself towards a platform, and he can request/download certificates. However, there is no guarantee that he will actually build a trustable device, or that he will not use the same certificate in several IoTs. To overcome this, if the platform directly flashes the IoT device with the certificate, the user would never have it in his possession. Furthermore, some automatic checks could be performed to the device code, like: only talks to the Edge, does not communicate at a too high frequency, among others. These two updates would certainly improve the security and confidence of the ecosystem.

Also about the security topic, the tokens must be improved. Currently, the tokens in use are an adapted version to work with the MQTT (however, this is not a true token). The implementation of an authentication server (as OAuth) to generate and check true tokens (JWT tokens) is a great improvement and a big step towards authentication (compared to the current solution). This solution may become very critical indeed. If the telecommunication operators keep the same position (very expensive data plans), the usage of a certificate in every communication is not viable, as a certificate occupies about 1.9KB plus the data to send (1MB would only allow to send less then 500 messages, which is about one message per two hours, under ideal circumstances). Although for some services it is enough, for many other it is not. However, with tokens, the IoT device could use a certificate to authenticate itself and request a token, and it would be valid for 24 hours, lets assume. This way, the certificate would only be used once a day, and the token in every communication. As a token can be as small as 80 bytes, plus some data bytes, lets assume it would be 100 bytes per message. Then, it is about 20 times smaller than in the previous case, meaning it would allow for 20 times more messages (about a message every 6 minutes). Furthermore, the implementation of a token system like OAuth does not imply storing more information from devices, as explained in section 3.4 , which means it is salable and it is impossible to compromise the performance of the system.

Still about optimizing data communication, and even improving power consumption and reducing the complexity of the system, the application protocol can be improved as well. As mentioned, currently the application protocol to communicate is the MQTT, which is a device-to-cloud protocol, and it was chosen for being lightweight (which it is indeed). However, watching more closely, the devices only publish to one single topic (SmartCity), and the Edge only subscribe to that same single topic. This means that a lot of functionalities that the MQTT implements are being discarded. This means that, even light, it could be lighter if all those features are removed. As the Smart City is a whole new topic, maybe it is time to emerge a new application protocol, that implements security at the application level, while MQTT and others implement it at communicating level, limiting the devices that can use it. For example, the IoTs developed in this project that communicate over GSM, due to module restrictions, can not use certificates, because the module is designed to only offer to the final user the application protocol. However, the WiFi IoTs developed can already use certificates. If the security would be implemented at the application level, everyone who can store a certificate could use it. Furthermore, as all those features would not be implemented, the communication would be even lighter, and the performance on the Edge would improve as well (less operations, because less features would be implemented).

It is absolutely required to update the current dashboards to deal with the real world. About this topic, some evident requisites must be implemented, like "study" the life-time of each measured variable, define limits and alerts, among others. However, a great introduction and more difficult to implement update would be tracking IoT devices by certificates. To clarify this idea, it is intended to promote privacy for the IoT owner. Currently, anyone can see the online devices of everyone. However, a company that has their own devices, may agree to share the data with the Edge, but may not desire that everyone can track their devices. To overcome this, if in the communication the device states an owner, then the user could provide a certificate to log-in in the dashboard, and one attribute of the certificate would be the "owner" name mentioned by the IoT. This way, it could be guaranteed (by filtering) that only the developer of the IoT could track their own IoTs, promoting the privacy of the company. Any device that does not state an owner, could be seen by everyone, with the "guest" log-in.

BIBLIOGRAPHY

Vipindev Adat and B. B. Gupta. Security in Internet of Things: issues, challenges, taxonomy, and architecture. *Telecommunication Systems*, 67(3):423–441, mar 2018.

Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 17(4):2347–2376, 2015.

Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, jun 2017.

Mohamed Almorsy, John C. Grundy, and Ingo Müller. An analysis of the cloud computing security problem. *CoRR*, abs/1609.01107, 2016.

Ioannis Andrea, Chrysostomos Chrysostomou, and George Hadjichristofi. Internet of Things: Security vulnerabilities and challenges. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, volume 2016-Febru, pages 180–187. IEEE, jul 2015.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

Carolina Tripp Barba, Miguel Angel Mateos, Pablo Reganas Soto, Ahmad Mohamad Mezher, and Monica Aguilar Igartua. Smart city for VANETs using warning messages, traffic statistics and intelligent traffic lights. In *2012 IEEE Intelligent Vehicles Symposium*, pages 902–907. IEEE, jun 2012. ISBN 978-1-4673-2118-1. doi: 10.1109/IVS.2012.6232229.

S. Basu, A. Bardhan, K. Gupta, P. Saha, M. Pal, M. Bose, K. Basu, S. Chaudhury, and P. Sarkar. Cloud computing security challenges amp;amp; solutions-a survey. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 347–356, Jan 2018.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog Computing and Its Role in the Internet of Things Characterization of Fog Computing. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–15, 2012. ISBN 9781450315197. doi: 10.1145/2342509.2342513.

Flavio Bonomi, Stefan Poledna, and Wilfried Steiner. The Role of Fog Computing in the Future of the Automobile. In *Fog for 5G and IoT*, pages 189–210. John Wiley & Sons, Inc., Hoboken, NJ, USA, mar 2017. ISBN 9781119187202. doi: 10.1002/9781119187202.ch8.

G. Brambilla, M. Picone, S. Cirani, M. Amoretti, and F. Zanichelli. A Simulation Platform for Large-Scale Internet of Things Scenarios in Urban Environments. *Proceedings of the The First International Conference on IoT in Urban Space. ICST.*, 2014. doi: 10.4108/icst.urb-iot. 2014.257268.

Teofilo Branco and Henrique Santos. What is Missing for Trust in the Cloud Computing? In *Proceedings of the 2016 ACM SIGMIS Conference on Computers and People Research - SIGMIS-CPR '16*, pages 27–28, New York, New York, USA, 2016. ACM Press.

S. Cherry. Edholm's law of bandwidth. *IEEE Spectrum*, 41(7):58–60, 2004. ISSN 0018-9235. doi: 10.1109/MSPEC.2004.1309810.

Hafedh Chourabi, Taewoo Nam, Shawn Walker, J. Ramon Gil-Garcia, Sehl Mellouli, Karine Nahon, Theresa A. Pardo, and Hans Jochen Scholl. Understanding smart cities: An integrative framework. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 2289–2297. IEEE, jan 2012. ISBN 9780769545257. doi: 10.1109/HICSS.2012.615.

Steve Deering and Robert Hinden. Internet protocol, version 6 (ipv6) specification. Technical report, Internet Engineering Task Force (IETF), 2017.

Prashant Ramchandra Desai. A survey of performance comparison between virtual machines and containers. *ijcseonline. org*, 2016.

F Díaz-del Río, J Salmerón-García, and J L Sevillano. Extending Amdahl's Law for the Cloud Computing Era. *Computer*, 49(2):14–22, 2016. doi: 10.1109/MC.2016.49.

M.T. Dlamini, J.H.P. Eloff, and M.M. Eloff. Information security: The moving target. *Computers & Security*, 28(3-4):189–198, may 2009. ISSN 01674048. doi: 10.1016/j.cose.2008.11.007.

SB Furber. *VLSI RISC architecture and organization*. Taylor & Francis, 2017.

Victor Gradinescu, Cristian Gorgorin, Raluca Diaconescu, Valentin Cristea, and Liviu Iftode. Adaptive traffic lights using car-to-car communication. In *IEEE Vehicular Technology Conference*, pages 21–25, 2007. ISBN 1424402662. doi: 10.1109/VETECS.2007.17.

Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Communications Surveys & Tutorials*, 17(3):1294–1312, 2015. doi: 10.1109/COMST.2015.2388550.

J Green. The internet of things reference model. In *Internet of Things World Forum (IoTWF) White Paper*, 2014.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. ISSN 0167739X. doi: 10.1016/j.future.2013.01.010.

John L. Gustafson. Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533, May 1988. ISSN 0001-0782. doi: 10.1145/42411.42415.

Jasmin Guth, Uwe Breitenbucher, Michael Falkenthal, Frank Leymann, and Lukas Reinfurt. Comparison of IoT platform architectures: A field study based on a reference architecture. In *2016 Cloudification of the Internet of Things (CIoT)*, pages 1–6. IEEE, nov 2016.

Robert E Hall, B Bowerman, J Braverman, J Taylor, H Todosow, and U Von Wimmersperg. The vision of a smart city. Technical report, Brookhaven National Lab., Upton, NY (US), 2000.

C Harrison, B Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, and P. Williams. Foundations for Smarter Cities. *IBM Journal of Research and Development*, 54(4):1–16, 2010. ISSN 0018-8646. doi: 10.1147/JRD.2010.2048257.

JL Hennessy and DA Patterson. *Computer architecture: a quantitative approach*. Morgan Kaufmann, 2011.

Doug Howard and Kevin Prince. *Security 2020: Reduce Security Risks this Decade*. Wiley Publishing, Inc., Indianapolis, Indiana, 2011.

Kai Hwang, Jack Dongarra, and Geoffrey C Fox. *Distributed and cloud computing: from parallel processing to the internet of things*. Morgan Kaufmann, 2013.

Yong Ho Hwang. IoT Security & Privacy. In *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security - IoTPTS '15*, pages 1–1, 2015. ISBN 9781450334495. doi: 10.1145/2732209.2732216.

Sidra Ijaz, Munam Ali, Abid Khan, and Mansoor Ahmed. Smart Cities: A Survey on Security Concerns. *International Journal of Advanced Computer Science and Applications*, 7 (2):612–625, 2016. ISSN 21565570. doi: 10.14569/IJACSA.2016.070277.

Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An Information Framework for Creating a Smart City Through Internet of Things. *IEEE Internet of Things Journal*, 1(2):112–121, apr 2014.

Constantinos Kolias, Angelos Stavrou, Jeffrey Voas, Irena Bojanova, and Richard Kuhn. Learning Internet-of-Things Security "Hands-On". *IEEE Security & Privacy*, 14(1):37–46, jan 2016. ISSN 1540-7993. doi: 10.1109/MSP.2016.4.

Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

Sathish Alampalayam Kumar, Tyler Vealey, and Harshit Srivastava. Security in Internet of Things: Challenges, Solutions and Future Directions. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 5772–5781. IEEE, jan 2016.

Stanford L. Levin and Stephen Schmidt. IPv4 to IPv6: Challenges, solutions, and lessons. *Telecommunications Policy*, 38(11):1059–1068, dec 2014. ISSN 03085961. doi: 10.1016/j.telpol.2014.06.008.

Shancang Li, Theo Tryfonas, and Honglei Li. The Internet of Things: a security point of view. *Internet Research*, 26(2):337–359, apr 2016.

Todd A Litman. *Parking Management Best Practices*. Routledge, 2006. ISBN 1-932364-05-6.

Luka Lugaric, Slavko Krajcar, and Zdenko Simic. Smart city - Platform for emergent phenomena power system testbed simulator. In *IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT Europe*, pages 1–7. IEEE, oct 2010.

Dan C Marinescu. *Cloud computing: theory and practice*. Morgan Kaufmann, 2017.

N Maslekar, M Boussedjra, J. Mouzna, and H. Labiod. VANET Based Adaptive Traffic Signal Control. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2011. ISBN 978-1-4244-8332-7. doi: 10.1109/VETECS.2011.5956305.

Marcel Medwed. IoT Security Challenges and Ways Forward. In *Proceedings of the 6th International Workshop on Trustworthy Embedded Devices - TrustED '16*, pages 55–55, 2016. ISBN 9781450345675. doi: 10.1145/2995289.2995298.

Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, sep 2012.

Taewoo Nam and Theresa A Pardo. Conceptualizing smart city with dimensions of technology, people, and institutions. In *Proceedings of the 12th Annual International Digital Government Research Conference on Digital Government Innovation in Challenging Times - dg.o '11*, page 282, 2011. ISBN 9781450307628. doi: 10.1145/2037556.2037602.

Jakob Nielsen. Nielsen's law of internet bandwidth. *Online at http://www.useit.com/alertbox/980405.html*, 1998. Accessed 16-9-2018.

OECD. OECD Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security. Technical report, OECD, 2002.

Charles P Pfleeger and Shari Lawrence Pfleeger. *Security in computing*. Prentice Hall Professional Technical Reference, 2002.

R Radhakrishnan, Majid Jamil, Shabana Mehfuz, and Moinuddin Moinuddin. Security issues in IPv6. In *International Conference on Networking and Services (ICNS '07)*, pages 110–110. IEEE, jun 2007. doi: 10.1109/ICNS.2007.106.

M. Mazhar Rathore, Awais Ahmad, Anand Paul, and Seungmin Rho. Urban planning and building smart cities based on the Internet of Things using Big Data analytics. *Computer Networks*, 101:63–80, jun 2016. ISSN 13891286. doi: 10.1016/j.comnet.2015.12.023.

Jyotiprakash Sahoo, Subasish Mohapatra, and Radha Lath. Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues. In *2010 Second International Conference on Computer and Network Technology*, pages 222–226. IEEE, 2010. ISBN 978-1-4244-6961-1. doi: 10.1109/ICCNT.2010.49.

Felipe Díaz Sánchez, Sawsan Al Zahr, Maurice Gagnaire, Jean Pierre Laisné, and Iain James Marshall. Compatibleone: Bringing cloud as a commodity. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 397–402. IEEE, 2014.

Henrique D Santos. ISO/IEC 27001 A norma das normas em Segurança da Informaçao. *Publicaçao da Associaçao Portuguesa para a Qualidade*, pages 11–19, 2006.

J. Sathish Kumar and Dhiren R. Patel. A Survey on Internet of Things: Security and Privacy Issues. *International Journal of Computer Applications*, 90(11):20–26, mar 2014.

R.R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, jun 1997. ISSN 0018-9235. doi: 10.1109/6.591665.

Martin Scharm. Connecting through a NAT - the non-trivial direction, 2011. Accessed: 2018-07-01.

Kewei Sha, Wei Wei, T. Andrew Yang, Zhiwei Wang, and Weisong Shi. On security challenges and open issues in Internet of Things. *Future Generation Computer Systems*, 83: 326–337, jun 2018. doi: 10.1016/j.future.2018.01.059.

S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini. Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164, jan 2015.

Kaiyu Wan, Danny Hughes, Ka Lok Man, and Tomas Krilavicius. Composition challenges and approaches for cyber physical systems. In *2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA'10)*, pages 1–7. IEEE, 2010.

Rolf H. Weber. Internet of Things – New security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010. ISSN 02673649. doi: 10.1016/j.clsr.2009.11.008.

Rolf H. Weber. Internet of things: Privacy issues revisited. *Computer Law and Security Review*, 31(5):618–627, 2015. ISSN 02673649. doi: 10.1016/j.clsr.2015.07.002.

Zheng Yan, Peng Zhang, and Athanasios V. Vasilakos. A survey on trust management for Internet of Things. *Journal of Network and Computer Applications*, 42:120–134, jun 2014. doi: 10.1016/j.jnca.2014.01.014.

Laurence T Yang and Minyi Guo. *High-performance computing: paradigm and infrastructure*, volume 44. John Wiley & Sons, 2005.

ChuanTao Yin, Zhang Xiong, Hui Chen, JingYuan Wang, Daven Cooper, and Bertrand David. A literature survey on smart cities. *Science China Information Sciences*, 58(10): 1–18, Oct 2015.

Drago Zagar and Kresimir Grgic. IPv6 Security Threats and Possible Solutions. In *2006 World Automation Congress*, pages 1–7. IEEE, jul 2006. ISBN 1-889335-33-9. doi: 10.1109/WAC.2006.375753.

Kai Zhao and Lina Ge. A Survey on the Internet of Things Security. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 663–667. IEEE, dec 2013.

Jun Zhou, Zhenfu Cao, Xiaolei Dong, and Athanasios V. Vasilakos. Security and Privacy for Cloud-Based IoT: Challenges. *IEEE Communications Magazine*, 55(1):26–33, 2017. ISSN 01636804. doi: 10.1109/MCOM.2017.1600363CM.

Project supported by DigitalSign Certificadora Digital LDA.