UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENTO DE COMUNICACIONES

INTERCONNECTION ARCHITECTURE OF PROXIMITY SMART IoE-NETWORKS WITH CENTRALISED MANAGEMENT

Ph. D. Thesis

By:

Pedro Luis González Ramírez

Advisors:

Prof. Dr. Jaime Lloret Mauri
Prof. Dr. Jesús Tomás Gironés

Valencia, España

Octubre 2021

Abstract

Interoperability between communicating objects is the main goal of the Internet of Things (IoT). Efforts to achieve this have generated several architectures' proposals; however, no consensus has yet been reached. These architectures differ in structure, degree of centralisation, routing algorithm, routing metrics, discovery techniques, search algorithms, segmentation, quality of service, and security. Some are better than others depending on the environment in which they perform, and the type of parameter used. The most popular are those oriented to events or actions based on rules, which has allowed them to enter the market and achieve rapid massification. However, their interoperability is based on alliances between manufacturers to achieve compatibility. This solution is achieved in the cloud with a dashboard that unifies the different allied brands, allowing the introduction of these technologies into users' everyday lives but does not solve problems of autonomy or interoperability. Moreover, it does not include the new generation of smart grids based on smart things.

The architecture proposed in this thesis takes the most relevant aspects of the four most accepted IoT-Architectures and integrates them into one, separating the IoT layer (commonly present in these architectures) into three layers. It is also intended to cover proximity networks (integrating different IoT interconnection technologies) and base its operation on artificial intelligence (AI). Therefore, this proposal increases the possibility of achieving the expected interoperability and increases the functionality of each object in the network focused on providing a service to the user.

Although the proposed system includes artificial intelligence processing, it follows the same technical aspects as its predecessors since its operation and communication is still based on the application and transport layer of the TCP/IP protocol stack. However, in order to take advantage of IoT-Protocols without modifying their operation, an additional protocol is created that encapsulates and adapts to its payload. This protocol discovers the features of an object (DFSP) divided into functions, services, capabilities, and resources, and extracts them to be centralised in the network manager (IoT-Gateway). With this information, the IoT-Gateway can make decisions such as creating autonomous workgroups that provide a service to the user and routing the objects in this group that provide the service. It also measures the quality of experience (QoE) of the service. Moreover, manages internet access and integrates with other IoT-Networks, using artificial intelligence in the cloud.

This proposal is based on a new hierarchical system for interconnecting objects of different types controlled by AI with centralised management, reducing the fault tolerance and security, and improving data processing. Data is preprocessed on three levels depending on the type of service and sent through an interface. However, if it is data about its features, it does not require much processing, so each object preprocesses it independently, structures it and sends it to the central administration.

The IoT-Network based on this architecture can classify a new object arriving on the network in a workgroup without user intervention. It also can provide a service that requires high processing (e.g., multimedia), and user tracking in other IoT-Networks through the cloud.

A major advantage of this architecture is that the data extracted from the objects and reflecting the preferences and habits of their users would not be controlled exclusively by the manufacturers but can be managed directly by the user, and it is the user who decides what to do with their data. Another great advantage is to leave the dependence on already manufactured objects without the possibility of being modified. Any everyday object can be converted and reinvented into a smart object by incorporating commercially available IoT technologies and devices. Thus, giving users the ability to control and transform their environments using AI. Similar to the options that users currently have to create their apps on their smartphones.

The proposed architecture is scalable since the number of Io-Networks with objects with the ability to process and connect to the internet is continuously growing. Moreover, it means that its potential use would provide an alternative for the compatibility of objects from different manufacturers. It could generate special interest in developing new integrated systems within reach of all users.

Resumen

La interoperabilidad entre los objetos comunicados es el objetivo principal del internet de las cosas (IoT). Algunos esfuerzos para lograrlo han generado diversas propuestas de arquitecturas, sin embargo, aún no se ha llegado a un conceso. Estas arquitecturas difieren en el tipo de estructura, grado de centralización, algoritmo de enrutamiento, métricas de enrutamiento, técnicas de descubrimiento, algoritmos de búsqueda, segmentación, calidad de servicio y seguridad, entre otros. Algunas son mejores que otras, dependiendo del entorno en el que se desempeñan y del tipo de parámetro que se use. Las más populares son las orientadas a eventos o acciones basadas en reglas, las cuales han permitido que IoT ingrese en el mercado y logre una rápida masificación. Sin embargo, su interoperabilidad se basa en alianzas entre fabricantes para lograr su compatibilidad. Esta solución se logra en la nube con una plataforma que unifica a las diferentes marcas aliadas. Esto permite la introducción de estas tecnologías a la vida común de los usuarios pero no resuelve problemas de autonomía ni de interoperabilidad. Además, no incluye a la nueva generación de redes inteligentes basadas en cosas inteligentes.

La arquitectura propuesta en esta tesis toma los aspectos más relevantes de las cuatro arquitecturas IoT más aceptadas y las integra en una, separando la capa IoT (comúnmente presente en estas arquitecturas), en tres capas. Además, está pensada para abarcar redes de proximidad (integrando diferentes tecnologías de interconexión IoT) y basar su funcionamiento en inteligencia artificial (AI). Por lo tanto, esta propuesta aumenta la posibilidad de lograr la interoperabilidad esperada y aumenta la funcionalidad de cada objeto en la red enfocada en prestar un servicio al usuario.

Aunque el sistema que se propone incluye el procesamiento de una inteligencia artificial, sigue los mismos aspectos técnicos que sus antecesoras, ya que su operación y comunicación continúan basándose en la capa de aplicación y trasporte de la pila de protocolo TCP/IP. Sin embargo, con el fin de aprovechar los protocolos IoT sin modificar su funcionamiento, se crea un protocolo adicional que se encapsula y adapta a su carga útil. Se trata de un protocolo que se encarga de descubrir las características de un objeto (DFSP) divididas en funciones, servicios, capacidades y recursos, y las extrae para centralizarla en el administrador de la red (IoT-Gateway). Con esta información el IoT-Gateway puede tomar decisiones como crear grupos de trabajo autónomos que presten un servicio al usuario y enrutar a los objetos de este grupo que prestan el servicio, además de medir la calidad de la experiencia (QoE) del servicio; también administra el acceso a internet e integra a otras redes IoT, utilizando inteligencia artificial en la nube.

Al basarse esta propuesta en un nuevo sistema jerárquico para interconectar objetos de diferente tipo controlados por AI con una gestión centralizada, se reduce la tolerancia a fallos y seguridad, y se mejora el procesamiento de los datos. Los datos son

preprocesados en tres niveles dependiendo del tipo de servicio y enviados a través de una interfaz. Sin embargo, si se trata de datos sobre sus características estos no requieren mucho procesamiento, por lo que cada objeto los preprocesa de forma independiente, los estructura y los envía a la administración central.

La red IoT basada en esta arquitectura tiene la capacidad de clasificar un objeto nuevo que llegue a la red en un grupo de trabajo sin la intervención del usuario. Además de tener la capacidad de prestar un servicio que requiera un alto procesamiento (por ejemplo, multimedia), y un seguimiento del usuario en otras redes IoT a través de la nube.

Una gran ventaja de esta arquitectura es que los datos extraídos de los objetos y que reflejan las preferencias y hábitos de sus usuarios no serían controlados exclusivamente por los fabricantes, sino que pueden ser administrados directamente por el usuario, y es el quien decide qué hacer con sus datos. Otra gran ventaja es la de dejar la dependencia con los objetos ya fabricados sin la posibilidad de ser modificados. Cualquier objeto cotidiano puede ser convertido y reinventado en un objeto inteligente incorporando tecnologías y dispositivos IoT disponibles en el mercado. Dándole la posibilidad a los usuarios de controlar y transformar sus propios entornos usando la AI. Similar a la opción que tienen actualmente los usuarios de crear sus propias aplicaciones en sus teléfonos inteligentes.

La arquitectura propuesta es escalable ya que el número de redes IoT con objetos con la capacidad de procesar y conectarse a internet crece continuamente. Esto significa que su potencial uso supondría una alternativa para la compatibilidad de más objetos de diferentes fabricantes, lo que podría generar un especial interés en el desarrollo de nuevos sistemas integrados al alcance de todos los usuarios.

Resum

La interoperabilitat entre els objectes comunicats és l'objectiu principal de la internet de les coses (IoT). Alguns esforços per aconseguir-ho han generat diverses propostes d'arquitectures, però, encara no s'arriba a un concens. Aquestes arquitectures difereixen en el tipus d'estructura, grau de centralització, algoritme d'encaminament, mètriques d'enrutament, tècniques de descobriment, algoritmes de cerca, segmentació, qualitat de servei i seguretat entre d'altres. Algunes són millors que altres depenent de l'entorn en què es desenvolupen i de el tipus de paràmetre que es faci servir. Les més populars són les orientades a esdeveniments o accions basades en regles. Les quals li han permès entrar al mercat i aconseguir una ràpida massificació. No obstant això, la seva interoperabilitat es basa en aliances entre fabricants per aconseguir la seva compatibilitat. Aquesta solució s'aconsegueix en el núvol amb una plataforma que unifica les diferents marques aliades. Això permet la introducció d'aquestes tecnologies a la vida comuna dels usuaris però no resol problemes d'autonomia ni d'interoperabilitat. A més, no inclou a la nova generació de xarxes intel·ligents basades en coses intel·ligents.

L'arquitectura proposada en aquesta tesi, pren els aspectes més rellevants de les quatre arquitectures IoT mes acceptades i les integra en una, separant la capa IoT (comunament present en aquestes arquitectures), en tres capes. A més aquesta pensada en abastar xarxes de proximitat (integrant diferents tecnologies d'interconnexió IoT) i basar el seu funcionament en intel·ligència artificial. Per tant, aquesta proposta augmenta la possibilitat d'aconseguir la interoperabilitat esperada i augmenta la funcionalitat de cada objecte a la xarxa enfocada a prestar un servei a l'usuari.

Tot i que el sistema que es proposa inclou el processament d'una intel·ligència artificial, segueix els mateixos aspectes tècnics que les seves antecessores, ja que, la seva operació i comunicació se segueix basant en la capa d'aplicació i transport de la pila de protocol TCP / IP. No obstant això, per tal d'aprofitar els protocols IoT sense modificar el seu funcionament es crea un protocol addicional que s'encapsula i s'adapta a la seva càrrega útil. Es tracta d'un protocol que s'encarrega de descobrir les característiques d'un objecte (DFSP) dividides en funcions, serveis, capacitats i recursos, i les extreu per centralitzar-la en l'administrador de la xarxa (IoT-Gateway). Amb aquesta informació l'IoT-Gateway pot prendre decisions com crear grups de treball autònoms que prestin un servei a l'usuari i encaminar als objectes d'aquest grup que presten el servei. A més de mesurar la qualitat de l'experiència (QoE) de el servei. També administra l'accés a internet i integra a altres xarxes Iot, utilitzant intel·ligència artificial en el núvol.

A l'basar-se aquesta proposta en un nou sistema jeràrquic per interconnectar objectes de diferent tipus controlats per AI amb una gestió centralitzada, es redueix la tolerància a fallades i seguretat, i es millora el processament de les dades. Les dades són processats en tres nivells depenent de el tipus de servei i enviats a través d'una interfície. No obstant això, si es tracta de dades sobre les seves característiques aquests no requereixen molt

processament, de manera que cada objecte els processa de forma independent, els estructura i els envia a l'administració central.

La xarxa IoT basada en aquesta arquitectura té la capacitat de classificar un objecte nou que arribi a la xarxa en un grup de treball sense la intervenció de l'usuari. A més de tenir la capacitat de prestar un servei que requereixi un alt processament (per exemple multimèdia), i un seguiment de l'usuari en altres xarxes IoT a través del núvol.

Un gran avantatge d'aquesta arquitectura és que les dades extretes dels objectes i que reflecteixen les preferències i hàbits dels seus usuaris no serien controlats exclusivament pels fabricants, sinó que poden ser administrats directament per l'usuari, i és el qui decideix que fer amb les seves dades. Una altra gran avantatge és la de deixar la dependència amb els objectes ja fabricats sense la possibilitat de ser modificats. Qualsevol objecte quotidià pot ser convertit i reinventat en un objecte intel·ligent incorporant tecnologies i dispositius IoT disponibles al mercat. Donant-li la possibilitat als usuaris de controlar i transformar els seus propis entorns usant l'AI. Similar a l'opció que tenen actualment els usuaris de crear les seves pròpies aplicacions en els seus telèfons intel·ligents.

L'arquitectura proposta és escalable ja que el nombre de xarxes IoT amb objectes amb la capacitat de processar i connectar-se a internet creix contínuament. Això vol dir que el seu potencial ús suposaria una alternativa per a la compatibilitat de mes objectes de diferents fabricants. El que podria generar un especial interès en el desenvolupament de nous sistemes integrats a l'abast de tots els usuaris.

Table of Contents

List of figures

List of tables

List of acronyms

6LowPAN	*IPv6 over Low power Wireless Personal Area Networks*
AP	*Access Point*
API	*Application Programming Interface*
BW	*Bandwith*
CoAP	*Constrained Application Protocoll*
CSV	*Comma-Separated Values*
IoT	*Internet of Things*
IoT-A	*Internet of Things Architecture*
IoT-ARM	*IoT Architectural Reference Model*
JSON	*JavaScript Object Notation*
LLN	*Low-Power and Lossy Networks*
LR-WPAN	*Low-Rate Wireless Personal Area Networks*
M2M	*Machine-to-Machine*
ML	*Machine Learning*
MQTT	*Message Queue Telemetry Transport*
NFC	*Near Field Communication*
PAN	*Personal Area Network*
RFID	*Radio Frequency Identification*
RPL	*Routing Protocol for LLN*
SaaS	*Software as a Service*
SBC	*Single-Board Computer*
SoC	*System on Chip*
SVM	*Support Vector Machine*
WLAN	*Wireless local Area Network*
WPAN	*Wireless Personal Area Networks*
WS	*Wearable-Based Systems*
WSN	*Wireless Sensor Networks*
WWAN	*Wireless Wide Area Network*
XML	*Extensible Markup Language*
ZDO	*ZigBee Device Object*

Chapter 1. Introduction

1.1. Introduction

The Internet of Things (IoT) and the Internet of Everything (IoE) are two closely related but different concepts. The IoT seeks to interconnect Things directly to the internet, while the IoE is a vision that seeks to connect users to Things and Things to each other. It will create new possibilities in understanding the current concept of “Network” and revolutionise how we design it. This new challenge also introduces other new concepts and different views of what is expected to be the IoE from those who create and develop technologies [1].

These new IoE-Networks will include different types of Things from different manufacturers, and we will have to rethink the way we design the new generations of networks taking into account how they connect and how they communicate through different protocols and interconnection technologies [1].

Nevertheless, we should first know that there are already some advances in IoT-Networks that will allow then, through the appropriation of some already accepted concepts, to reach the first IoE-Networks. These advances in this topic can already be seen through some everyday things, for example, in the house (appliances) in the office (registration and access control) and the car (geographic location) are connected to the internet. However, there is still a high percentage of things that are not yet connected or, are in the process of doing, and each represents an IoE-Network. E.g., in the home (light bulbs, doors, windows, bicycle), in the office (desk, chairs), and the car (automotive parts, doors, stereo and systems in general), including those devices that a user is wearing (wearables such as, clothing, watch, glasses, bracelets). In this way, Things near each other will be able to form proximity IoE-Networks since the vision of these IoE-Networks enables interconnection between multiple proximity IoE-Networks [2].

If additionally many of these conventional Things are Smart Things, able to deduce typical behaviours of their users through usage statistics and autonomously publish, discover and connect to Machine-to-Machine (M2M) services through the network [3], to the cloud or other users, giving rise to the creation of Smart Networks connected [4], then would be obtained proximity smart IoE-Networks.

This work proposes an intelligent centralised management architecture that brings together all these connectivity and interoperability needs to network Things. Things are classified, according to their features in capacities, functions, services, and resources and are organises in layers. Following this organisation, it is possible to create multiple network configurations around a central entity. It is also a flexible and reprogrammable architecture, which helps it to readapt and evolve, enabling high scalability.

The idea is to show with different applications and through different scenarios its scalability and easy adaptability. These scenarios were mostly recreated in domestic environments, easy to understand and experiment. In order to the user's activities been the as a central element for providing a service. In other words, one of the central focuses in this work is to ensure that the architecture allows the generation of network designs aimed at providing a service to the user. Service that will require working together through collaboratives workgroups to achieve it whereby it is necessary to manage its M2M relations, routing, grouping and Internet access. In this way, the machines involved in providing a service can be controlled by Artificial Intelligence (AI) through the features that define their capacities, functions, services, and resources shared in the network.

A clear IoT or IoE concept is essential to design, create, and modify the relations between IoT-Interconnection technologies, IoT-Protocols, IoT-Gateway, and IoT-Platforms necessaries to propose a new concept. The proposed architecture is born from this concept and based on studying current concepts issued by organizations and alliances in IoT. Taking these approaches and establishing a network model that incorporates Machine Learning (ML) techniques or other AI technologies, the architecture is expected to offer a design option for smart IoE-Networks.

An IoT network design that an organisation follows under this architecture's interaction policies allows conventional and intelligent things to be integrated with ease. Whereby smart things that are currently being marketed, they will have an architecture that takes advantage of the functions and services they offer, auto-integrating it into the network without user intervention and focusing on offering intelligent services.

1.2. Objectives and motivation

The growing rise of IoT-Networks and the problems in their multiple forms of implementation, promote the generation of different architectures that allow the interoperability of the Things like objects that need to connect to the Internet. This thesis's main objective is to present an architecture that facilitates the interconnection of Smart Things, manages them and organises them hierarchically to share resources and provide services based on users' needs.

1.2.1. Objectives

The objectives of this thesis are the following ones:

1. Study which IoT-Protocols are the most accepted, studied at an academic level and implemented at a commercial level.

   The IoT-Protocols currently used are numerous and diverse. However, three were selected due to its ability to reach the architecture's sensor layer and its applications. Among them MQTT, CoAP and HTTP-RESTful, among which MQTT was the most used in the experiments.

2. Study the architectures and applications where each of the IoT-Protocols performs best according to their functions and according to the network topology in which they are implemented.

   The architectures found so far are scarce and are defined differently with connection-oriented and software-oriented structures, going through all the topological configurations. It allows the use of Frameworks based on these architectures and takes advantage of the most used technologies. Some of them have been implemented commercially, without having gone through arduous standardization processes before, to respond immediately to the high demand from users. Therefore, although they are already being used, they still require improvements and specifications. However, his study allowed to define the architecture proposed in this thesis.

3. Selection of an IoT-Protocol designed to be used in proximity networks, easily modified in its operating mechanics to improve its architecture and put it to the test together with the other protocols already characterized, compare it and document its performance.

   The selected protocols and frameworks are open source, but the protocol that was easiest to modify and adapt to the proposed architecture operation was MQTT. At first, it was thought to work with the AllJoyn framework, but a framework is not the same as an IoT protocol, so it was decided to use MQTT. This protocol has some limitations that were gradually improved by recent versions, however, during this thesis's development, it was necessary to modify it before these improvements arrived, understanding its operating mechanics and adapting it to the requirements of the proposed architecture to verify its operation.

4. Design an architecture from on the results found to the improvements proposed in one of the IoT-Protocols, allowing both the exchange of information between all the proximity networks interconnected to this architecture and the exchange of services with centralized management.

   The proposed architecture was designed from the architectures found, observing how they have been implemented to preserve their most important parts and complement them with new functions, obtaining a hybrid architecture. This architecture allows the operation with the current frameworks widely spread among users and manufacturers. Its centralized management is based on the intelligent operation of the IoT-Gateway and intelligence in Things. What gives it an additional functionality capable of responding autonomously to changes in the network.

5. Include artificial intelligence methods and techniques so that M2M communication through the announcement of services on the network, allows them to make logical decisions without user intervention.

   The system's centralized AI was tested with different ML classifiers and different datasets obtained by creating an additional protocol that adapts to the size of the payload of each M2M protocol. With this protocol, the information required by the AI is transported through different messages to the different layers of the architecture. It is a simple protocol that kept the initial idea of taking advantage of different IoT-Protocols, including those with small payloads for reaching low capacity devices in the sensor layer. To demonstrate the use of an AI on the architecture, different IoT-Networks applications are designed, where the need to make use of AI and how it integrates Artificial Intelligent Assistants (AIA) is observed. Therefore, the central AI automates various services by grouping things involved in providing a service, and that commonly work collaboratively to achieve it. Hence the possibility that the central AI can distribute the work through direct connections between the Things involved without going through the central AI.

6. Create test scenarios based on mobile applications developed under the Java language using the Android and Android Things platform, in different types of topologies to demonstrate the architecture's viability.

   Many of the experiments were based on scenarios; the most common and the main one was the Smart home. Then it was integrated through AI in the cloud to other scenarios based on proximity networks such as Smart Office and Smart Microgrids, resulting in a Smart City. Programmable development cards with different communication capacities and modules were used for each Cosa, using Java and Python languages and some mobile applications developed for Android.

7. Measure and collect data on the network scenarios implemented through capture tools and protocol analyzers and analytically model the designed architecture.

   Several simulators were used to test the scenarios, and also it is was test in real environments with hardware devices. Measurements and data were captured with protocol analyzers such as Wireshark and other data with simulators. This data was used to observe the network's performance under the proposed architecture and the other part for the ML. The data captured from the networks designed based on this architecture showed a variation in the packets' size and speed that carried the proposed protocol. Through this protocol, information collected in the IoT-Networks allowed building a network model based on workgroups to test ML techniques. Afterwards, some of them were experimented with to find the most appropriate one according to the data extracted from each connected thing.

8. Develop and implement a prototype to evaluate and analyze the performance of the designed architecture.

   An IoT-Gateway was designed and implemented, programmable to route, store, manage, support an AI and host services. This device is the central administrator of the architecture and assigns a specific layer's roles to a device based on its functions. The Gateway uses an AI that controls different algorithms, including creating workgroups and the management of internet access.

1.3. Precedents

Before IoT, everything was computer networks, and only one way was known to connect them. Physical cabling and local interconnections were the first forms of communication between machines. When talking about the network, one could only imagine a group of interconnected computers. This interconnection spread over great distances with high volumes of information and gave rise to the Internet. With this new vision of connectivity, the evolution of the concept of the Internet and of sharing resources at a broader level, allowed the network of networks to provide greater coverage and reach everything that could be connected. The meaning of connectivity born with personal computers and that was believed to be unique to these computers went out of its standards when the network began to share its resources with other devices such as printers and projectors locally and to other networks through the Internet. At that time, there was already talk of globalization and the possibility of an interconnected world. With the evolution of electronics and faster and smaller processors, the possibility of incorporating computing and communication utilities to everything gave rise to the current mobile devices. Devices currently continue to improve their utilities with applications that range from people's everyday lives to their work and family activities. In this case, the use of the mobile phone (Smartphone) stands out as the main technological tool and the highest priority in modern life.

This idea of expanding connectivity to everything includes all kinds of objects, as long as technology and its use justify it. Therefore, it is now possible to see many implementations that include common and everyday objects.

This approach of leveraging available technologies to interconnect objects in the known world of networks is a new terrain, and integrating them creates a host of challenges and different ways of rethinking the old concept of networks and the Internet. Like mobile devices, networks and the Internet have adapted to incorporate them, but still, it is difficult to think of an Everything's Internet, so it is still premature to formulate a unification theory for all Things.

However, these objects or Things have gradually been incorporated into networks and the Internet, initially through isolated efforts of manufacturers, and then through increasingly strong strategic alliances that have formed the manufacturers' consortiums.

Some of these early initiatives were related to local wireless connections. For example, if was wanted to connect a Thing to the Internet, it only had to directly connect to the Internet through its mobile or a WiFi connection at home or office. Here is born a new concept, which offers the virtual storage and management service called the Cloud. It allows managing these Things by a user through a web or mobile application, inside or outside the same proximity area but with an Internet connection.

In some cases, there are already alternative developments that allow direct control of Things in small proximity areas or personal, through mobile applications with WiFi and Bluetooth connections, and all improved versions of these technologies. In addition to this, for long-distance solutions but with low bandwidth, other manufacturers made available other technologies different from those known, such as LoRa and SigFox.

However, many devices already had control connections such as Radio Frequency (RF) or infrared, which can be found in everyday things such as televisions, stereos, air conditioners and heating, which would later be used by smart assistants such as "Alexa" and "Google Home" among others. These connections range from simple data of ON / OFF to data streaming with low bandwidth, e.g., to adjust sound and light intensity. Nevertheless, even so, they are still seen as isolated and independent cases outside the concept of IoT.

With the arrival of IoE, the idea is to give cover all these Things and include all the other existing proximity IoT-Networks. This, in turn, integrated with the new generations of Smart Things and its connectivity technologies. In order to open new possibilities in the development of architectures and protocols, aimed at making these networks interoperable and facilitating collaborative work between Things for offering services to users.

However, the integration of these Things has been done progressively, since these conventional networks are still based on TCP / IP architecture. Therefore, to improve its operation and allow the integration of AI to improve its performance and thus obtain greater benefits, it is necessary to analyze current scenarios and refocus them with more appropriate architectures.

For example, before if it was wanted to create a proximity network between different types of Things from different manufacturers, it could only be achieved if the same type of connection was maintained, only Bluetooth or only WiFi or only Zigbee. In a few cases where the development effort at the hardware level was great, an additional router was implemented with other interconnection technologies to provide a solution to a network with Things of different technologies [1]. The problem was that the Things had to be from the same brand, from the same manufacturer.

To solve this problem of interoperability in proximity networks, it appears several architectures and protocols proposals [5] to convert conventional proximity networks into networks under the IoT concept. Simultaneously, some consortia create frameworks for this same purpose, such as the AllJoyn initiative [6]. AllJoyn was an open-source project developed by Qualcomm's innovation centre (Qualcomm, 2011), and sponsored by AllSeen Alliance (AllSeen Alliance, 2016). Its purpose was to provide a universal framework that would allow interoperability between different types and manufacturers' technology products. However, this idea persists through IoTivity, a new open-source IoT standard.

AllJoyn and IoTivity are open source M2M protocols implemented under one framework, but currently, there are others, that are not frameworks but are accepted and used by most manufacturers. Some of these are, Message Queue Telemetry Transport (MQTT), Restricted Application Protocol (CoAP), Hypertext Transfer Protocol with Representational State Transfer (HTTP RESTful), among others [7].

1.4. Memory organization

This thesis begins with the introduction in Chapter 1, which addresses how the document is developed, describing the proposal and the importance of this topic, and describing this thesis's main and specific objective.

The rest of the document is organized as follows:

Chapter 2 presents the state of the art of the architectures, concepts, protocols, devices, and technologies most used in IoT.

In Chapter 3, the architectures' features, and advantages of the current or most popular are analyzed, and the interconnection architecture of this thesis is proposed.

Then, Chapter 4 presents the proposed architecture and its model, which shows the proposed protocol for the operation of the architecture and the analysis of each model to perform grouping, routing, and quality of experience on a specific service.

In Chapter 5, the tests and evaluations are carried out based on the models proposed in Chapter 4. Therefore, some experiments based on implementations of the proposed architecture on real hardware through simulations are shown.

Chapter 6 shows how the simulations were implemented and present some applications through use cases with the proposed architecture.

Finally, in Chapter 8, the conclusion and future lines of research are presented.

Chapter 2.
State of the art

2.1. Introduction

Communicating Things with each other and connecting them to the Internet implies that all Things must first have the ability to establish one communication. That is, to achieve this, at least the combination of some processing mechanism and at least one module with some connectivity technology must be incorporated into a Thing.

From this point of view, it could be said that all the network concepts that were known up to now have evolved, taking new forms, and therefore are going through a new stage of redefinition. Part of these new challenges can be observed with the appearance of mobile devices (Cell Phones, Tablets, laptops) which brought with them the massification of new technologies (Bluetooth, WiFi and NFC) and gave strength to the need to implement a system IP addressing broader (IPv6). In addition, this type of "all-in-one" mobile devices that include voice, video, and Internet browsing among others, has changed how users use data, as the amount of bandwidth necessary to have good service is increasingly demanding.

Figure 1 shows a simple example of a conventional home network, which was previously only possible at a business level, and which also allows voice, broadband and television through a router connected to an Internet Service Provider (ISP). In this example from Figure 1, the router (WiFi and Ethernet) connects in a star topology with the old devices (black colour) and supports the new devices (red colour) such as mobile phones, tablets, laptops through the same WiFi connection.

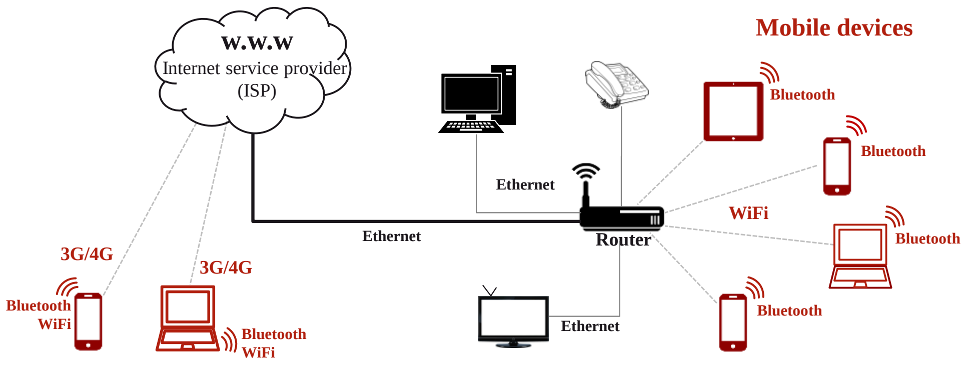


Figure 1. Conventional Network with Mobile Devices

This type of network allows us to understand how networks have grown and evolved with the appearance of new and different devices, modifying users' lifestyle in an increasingly connected world. To this evolution in the networks, the incorporation of "Things" is added as one more element of the network and then the vision changes from every point of view, giving space to new proposals that allow the integration of everything with everything.

To approach this thesis's architecture design, it is necessary first to study the architectures and approaches that currently exposed to the public to establish a difference and a contribution when designing an IoE-Network.

Therefore, state of the art presented here shows how proximity IoE-Network can be designed, combining different IoT concepts and architectures with different operating systems, programming languages, interconnection technologies, IoT-Protocols, frameworks, IoT-Platforms in the cloud, artificial intelligence algorithms and different IoT-Gateway proposals since in this last, resides the centralized management.

2.2. Concept of Internet of Things and Everything (IoT & IoE)

The Internet of Everything (IoE) is an extended concept of the Internet of Things (IoT) where Machine-to-Machine (M2M), Machine-to-People (M2P) and People-to-Machine (P2M) communications describe complex systems derived from the relations that exist between people and processes [8].

In this sense, it is important first to define the IoT's initial and operational criteria currently used to have a more accurate IoE concept.

Today's IoT devices that are design and manufacture are released to the public with a wide range of network protocols, applications, and special configurations network. In addition, the IoT technology that connects the "Things" to the Internet is done through various types of short-range or long-range wireless technologies depending on the type of network and the management policies associated with the network.

If we focus on some real implementations' specific cases, what is found are "Things" connected directly to the Internet, without any relationship with other Things [9]. This does to the concept of IoT being seen as simply connecting a "Thing to the Internet", and gives rise to confusion with other concepts such as Domotics and Telemetry. In reality, establishing a difference between them would generate even more confusion when they are such close concepts that converge with many common elements. Let us remember that Domotics began with the automation of Things in a specialized way and tailored to users' needs, making a remote control over them through available communication systems. When mobile devices appeared, and with them, the mobile applications, it became possible to perform remote control from a mobile phone or tablet. However, when making remote measurements of physical magnitudes, for example, through sensors in the house, we would be talking about Telemetry. If we join both concepts and use IoT-Protocols to seek interoperability between different manufacturers, we get a concept closer to the IoT as we know it today.

In Figure 2, an example of a network is observed where some Things with communication capacity can connect to the Internet and be managed from a mobile application.

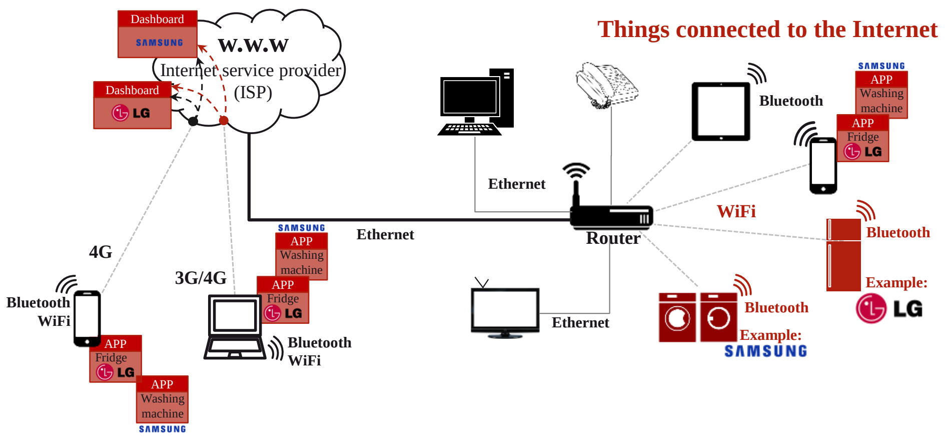


Figure 2. Conventional Network with Mobile Devices and Things

Much of what is related to this concept depends on manufacturers, as they are the ones who come up with new and innovative implementations that users consume, helping Things with these special features to become widespread.

Therefore, both manufacturers and international standardisation organisations urgently require manufacturers to include compatible technologies in their products.

2.2.1. Organizations and alliances working on the IoT

All companies, organizations, working group and consortia are made up of people who are dedicated to the study and manufacture of the elements that make up the “technological integration systems; These meet in order to determine which is the best system, to standardize it, relating it to IoT networks, thus generating an ideal language, which facilitates intercommunication between things [10].

In addition to defining standards such as IoT, they also do it in concepts related to IoE, Internet, WoT, IIoT and M2M, as shown in Figure 3 [11]. This figure shows how each concept is contained within the other as a subset of the great concept of IoE.

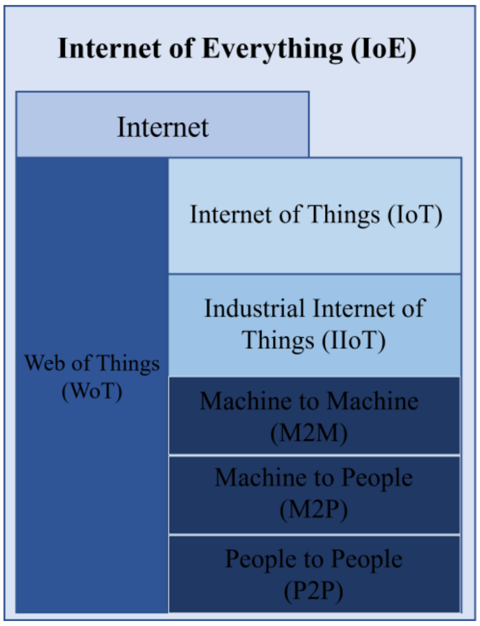


Figure 3. Internet of Everything

As seen in the figure, the IoE concept contains all the connected objects and conventional network devices. It also suggests possible architectures since the internet coverage in the figure does not completely encircle the sub-concepts of IoT, IIoT, WoT and the relations between machines and people (M2M, M2P, P2P). It suggests that it includes those not connected to the internet and could maintain their operation within a local network connection as proposed in the Fog Computing architecture. In the case of WoT, IoT platforms suggest using a Cloud Computing architecture where data processing is done in the “Cloud” and visualization on the “Web”.

- Internet Engineering Task Force (IETF)

The main institution in Internet standardization has a direction fully committed to the IoT, which together synchronizes the efforts of all its working groups, in order to review thoroughly, absolutely all the specifications, verifying the coherence of each one, and monitoring various activities, all linked to the IoT, which, in turn, perform other standardization groups [12].

- Institute of Electrical and Electronics Engineers (IEEE)

A part of the organization of this institute is completely dedicated to the IoT, about its research, implementation, application, and use, through a great diversity of technologies; structured from an information centre for the existing technical community and a magazine that publishes articles on the latest advances in IoT, along with various topics, including system architecture, network protocols and communication. Additionally, futuristic visions (based on IoT) of smart cities built entirely from IoT (walls, roads, transportation, among others) are captured [13].

The IEEE defines the IoT based on the complexity degree of the environment and application scenario:

1. Low complexity level defines the IoT as a network of objects or things connected (physical or virtual) to the internet with unique identification. For example, the fingerprint on the index finger of every human being. It means that "things" have programming and detection/performance capabilities. Whereby can extract information from objects and modify their state at anytime and anywhere [14].

2. Medium/high complexity levels propose that the IoT is an adaptable and auto-configurable network, capable of interconnecting objects or things (physical or virtual) to the internet, using standard communication protocols. These objects also can program and detect/actuate. In addition, the information extracted from each connected object contains its identification, status, location, commercial and social data, and the service it offers. This service can be provided with or without human intervention through intelligent interfaces. Its main importance is what is available to any being (human or machine) that needs to run applications under a secure environment [14].

This thesis combines the previous definitions to describe part of the architecture proposed in Chapter 4.

- Alliance for the Internet of Things and Innovation (AIOTI)

Created by the European Commission (EC) and composed of large companies, SMEs, and startups. This alliance was formed in order to support a purely European IoT ecosystem, in which it would have content and standardisation policies. More than 75% of AIOTI's functions and activities are carried out using planned innovation, research and standards working groups, which focus their unified effort on fully defined areas of development [15].

- Safety Committee for Industrial Control and Automation Systems (ANSI / ISA)

This committee is in charge of developing standards, protocols, procedures, and technical reports to implement safe automation and control systems. It always has the priority to protect industrial automation and control systems against cyber attacks. Without the protection of said committee, it may be subject to external or internal malicious penetrations, causing losses in production levels and violation of regulations established or environmental damage, among other negative consequences [16].

- Internet Industrial Consortium (IIC)

This consortium is developing in line with IIoT (Industrial IoT) innovations, with activities such as updating the industrial IoT vocabulary, specifically about architectures, security and analytics, which organisations can use to improve their communication.

In order to accelerate the delivery of an industrial IoT architectural framework, IIC partnered with the Open Interconnection consortium [17] to publish in 2015 such an architecture, which would serve as an IoT reference [18].

- International Organization for Standardization (ISO)

This institution has a “non-governmental” and non-profit philosophy; developed for international norms and standards applicable to manufacturing and services. It provides a “map” of the various layers that receive and send data. Each IoT protocol, belonging to the IoT system architecture, provides communication between various devices [19].

One of the first preliminary reports on IoT and later on “smart cities” was published in 2014, under the file name: ISO/IECTC-1 [20].

- International Telecommunication Union (ITU)

This governmental organization regulates telecommunications worldwide, being the oldest (since 1865). Initially, it was created to control the interconnection of existing telecommunication systems between all countries.

In order to create global standards for IoT, in 2015, ITU developed a new study group whose central axis is reflected in applications in IoT [21].

La ITU bajo una visión normalizada, define a IoT como una infraestructura mundial, para uso y desarrollo de la sociedad de la información, la cual permita utilizar servicios complejos, mediante la interconexión de objetos, ya sean físicos o virtuales, todo a través de la interoperabilidad entre tecnologías en comunicación e información, tanto a nivel actual, como a futuro.

The ITU, under a standardized vision, defines IoT as a global infrastructure for the use and development of the information society, which allows the use of complex services, through the interconnection of objects, whether physical or virtual, all through the interoperability between communication and information technologies, both currently and in the future [22].

- Organization for the Advancement of Structured Information Standards (OASIS)

It is an international non-profit consortium oriented towards the adoption and convergence of structured information standards in electronic commerce and web services.

OASIS works on three open-source standards such as the Message Queuing Telemetry Transport Protocol (MQTT), Advanced Message Queuing Protocol (AMQP) and OASIS Open Building Information Exchange (OBIX) in order to guarantee interoperability in IoT. The IoT protocol chosen by the working group is that of MQTT, and the MQTT-SN protocol, which is adapted and optimized for wireless sensor networks, the latter in order to provide compatibility with different technologies, such as M2M [23].

- Internet Protocol for Smart Objects (IPSO)

This non-profit organization founded in 2008 was a forum of allied companies that promoted a protocol for smart objects or things. One of its main goals was to lead the use of open IoT standards and establish the use of the Internet Protocol (IP) for the connection of intelligent objects through the applications of IoT and M2M [24].

2.2.2. Concept according to the IETF

According to the IETF, the IoT is the network of physical objects and “Things” integrated with electronic components, software, sensors, and connectivity to allow objects to exchange data with the manufacturer, the operator and other connected devices.

It further defines IoT as devices that are often limited in their computing and communication capabilities and are now more commonly connected to the Internet and various services that build on the capabilities that these devices jointly offer. This development is expected to generate more M2M communication using the Internet without user intervention.

The IoT is a fast-growing technology trend that adapts with other emerging technologies. Several IETF working groups from different areas are developing protocols that are relevant to the IoT. These protocols are used by various companies, as well as organizations and alliances dedicated to creating IoT standards, in order to obtain specifications for interoperability. Due to the diversity in the development and use of IoT protocols, it is usually necessary to coordinate between the different groups working on IoT to reach agreements.

2.2.3. Concept according to some manufacturers

Some technology manufacturers have followed the IETF concept to apply it to their products and offer them to users as soon as possible. However, its main objective is to achieve a presence in the new IoT market just as device manufacturers do in the industry as quickly as possible. This eagerness to enter the market quickly makes many of these products difficult for users to handle and install in the home and industrial networks. Therefore, few people are interested in this, as did those who acquired the first personal PC for the home and those who relied on electronic commerce in its early days. Now the challenge is that people are interested and trust in all their objects connected to the internet. However, step by step, people have become familiar with these technologies since mobile devices have been part of people's lifestyles since before, so it is possible that they also trust that they now have everything connected. However, it is also necessary that manufacturers continue to improve their products so that the connectivity, installation and handling of the objects are easy to use as smartphones. In this way, the future in the trade of these connected objects would lead to massification, becoming a further opportunity for digital transformation (evolution of things).

Next, some manufacturers are presented and how they conceive the connection of their products in a home network, making use of mobile apps, cloud platforms, and additional network elements to make it possible to connect objects to the internet.

- Samsung SmartThing

This brand of manufacturers, as well as others, are connecting their products to the internet. These products are mostly household appliances. It adds value to their products as users can control and monitor them on and off the local network through the internet. In this way, a user can access from the office or anywhere in the world know the on/off status of their devices and put them into operation before arriving home or during the journey home.

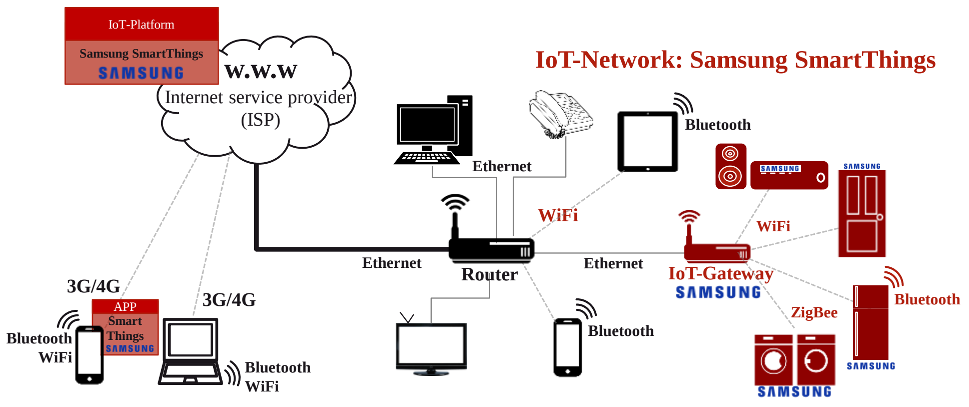


Figure 4. IoT-Network: Samsung

In the network connection example in Figure 4 an intermediary gateway is required between the appliances and the home router. Current versions already allow direct connection through the home router using WiFi. However, its administration only works with objects from the same manufacturer and those of some allies.

- Ozom 1.0

Ozom [25] is an IoT brand that launched devices for everyday use in the home and connected to the internet. Most of these devices are security and lighting devices such as cameras, sensors, light bulbs, among others. The brand requires that the objects be connected through an additional router (ozom box) with WiFi and ZigBee connectivity, which connects to the home router. In addition, the brand provides an Android or IOS mobile application for monitoring and control. Figure 5 shows its connection and describes how the application is used within the home (local network) and outside the home (via the internet). According to the figure, acquiring a new brand different from those traditionally acquired by users in the trade is necessary. It means that if a user decides to have his home or office interconnected, he will have to buy a single type of brand and in which, in this case, it does not include electrical appliances. So, if he wanted to connect with electrical appliances as described in Figure 4, it would not be possible since they are different brands, have different applications and different routers. Interoperability for a user would be complicated to manage due to the large number of applications used for each thing acquired.

Figure 5 describes a typical connection of a mixed network in a house where it is necessary to use an ozom box. Once installed and connected to the main router, ozom brand objects can be connected to the internet and controlled through the ozom 1.0 App. Some sensors of this brand use ZigBee technology, for which this intermediary is necessary. Once logged in, the user must register on the platform via the app, where all the usage data of each object will be recorded every time it is used. In this version, it is not yet possible to download the data set to be managed by the user. They can only be accessed for use through usage rules and the history of some objects.

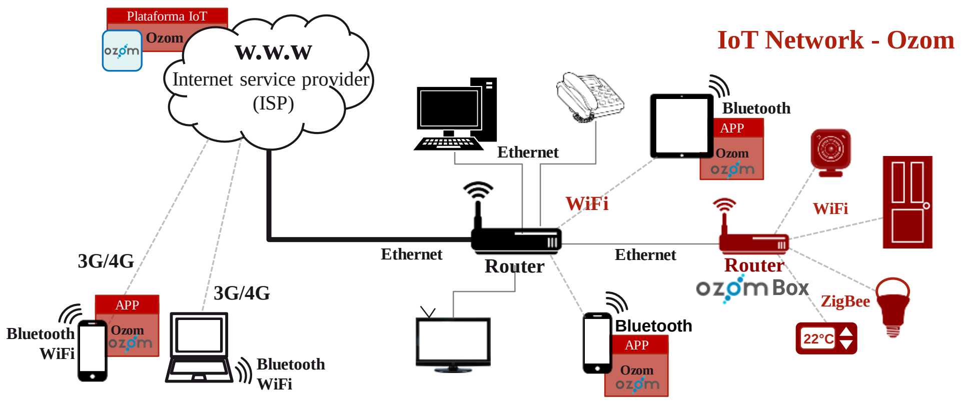


Figure 5. IoT-Network: Ozom 1.0

- Ozom 2.0

In this improved version, an alliance has been created between different manufacturers of objects connected to the internet. It allows for presumed interoperability, but this can only be done at the unified cloud level on a single platform. In this case, the platform that unifies the management of devices from various brands is ozom 2.0 [26]. However, each manufacturer has its app to control the devices it sells, so when the ozom app is used, these objects are under the control of this single platform. One way to achieve interoperability is to do it through the cloud, but this would imply that the user would continue to be limited to using a specific number of brands that do not necessarily cover all types of objects, including household appliances. Something similar happens in the industry; however, making this comparison is easier with the objects of a smart home. In Figure 6, it is shown that the system has also been improved since only a WiFi connection is enough, so with the conventional router available in homes, it is enough to integrate the objects into the home network. It is provided that the objects purchased belong to the group of brands compatible with ozom and WiFi. That is that each object can be controlled by the user or by preset rules as long as there is internet service. If there is no internet, the objects cannot be controlled; this becomes a problem because, in most cases, they cannot be operated locally, blocking user activities. For the network design described in Figure 6, the objects are integrated into a conventional mixed WiFI network, sharing Internet access with mobile devices and computing devices. The integration of smart assistants such as Google Home [27], [28] and Amazon Alexa [29], is also observed, from which objects can be controlled using voice activation.

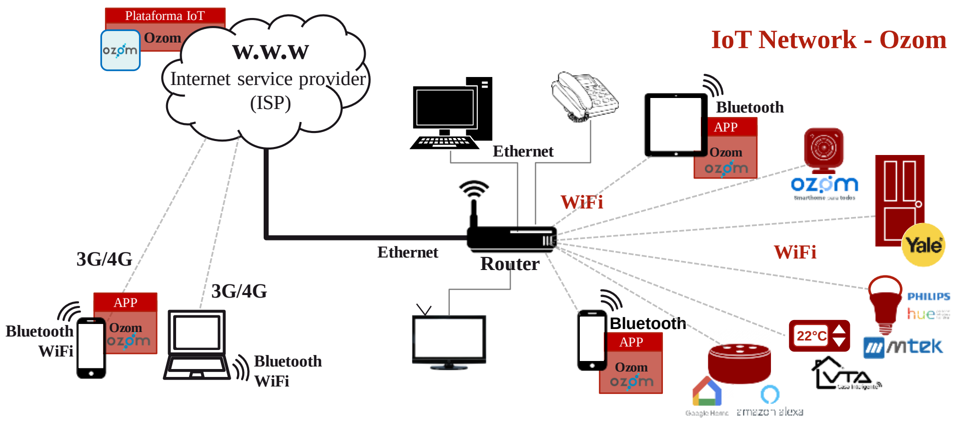


Figure 6. IoT-Network: Ozom 2.0

As in the version of ozom 1.0, seen in Figure 5, this new version covers a greater number of objects: lighting, cameras and security, alarms and sensors, pets, controls, electrical, among others, but not electrical appliances [30]. In this version, it is also not possible to download the dataset of all features and events of things, together with their network parameters for the user to analyse.

2.2.4. Concept according to Cisco

The concept of the Internet of Everything (IoE), according to Cisco [31], proposes IoE as "the intelligent connection between people, processes, data and Things" since, in IoT, all communications are between machines. However, IoT and M2M are considered as its synonyms. It means that IoE is the extended concept of IoT, which includes, in addition to M2M communications, machine-to-person (M2P) and person-to-person (P2P) relations [32].

This IoT concept extended includes any physical or virtual object or element with a communication module, transmitting data without human intervention (P2M input). In addition to being everyday elements such as sensors and electrical appliances, these objects also include any other element that had not been connected before. Moreover, how communication is interconnected and automatically established, including those established by the user, is also a central part of this concept. On the other hand, the IoE also includes the coverage and bandwidth of each connection and the relationships between the devices on the network.

Figure 7 shows a typical connection under this concept in which a control agent is necessary where the connection rules between things are established. In the local network, these rules are programmed on a server, or if there is one, the cloud platform can also be established there. Either case can be used, or even both, as they are not mutually exclusive.

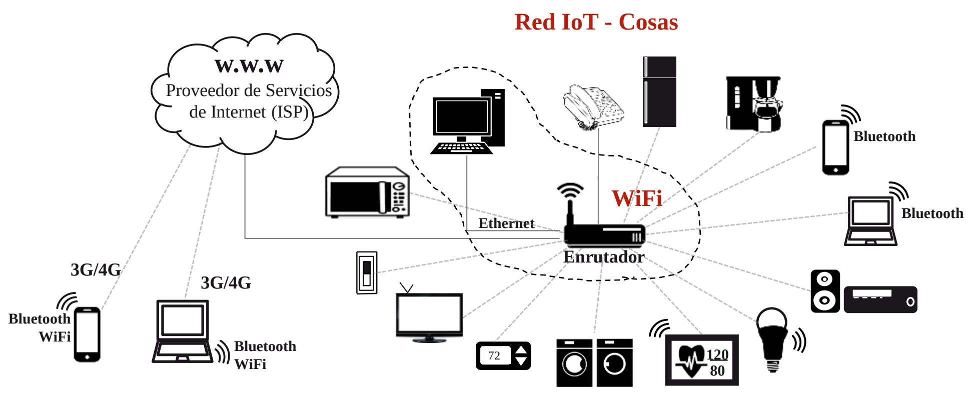


Figure 7. IoT-Network according to Cisco

2.3. IoT-Networks

Networks have evolved and have been integrating for new elements, among them the "Things". These are the connected objects, called the internet of things, which vary according to their type of connection, information protocol, topology, and architecture. However, so far, they all operate on the TCP / IP protocol stack. Therefore, it continues to operate under the traditional network concept with the difference that it was extended to other types of network elements (Things) with different capabilities.

These networks are also classified according to their coverage, such as proximity networks within a local range and commonly use technologies such as WiFi and Bluetooth, among others. Since the integration of objects to a connected world came at the full wireless networks boom, most objects connected to the internet wirelessly. These networks can also be called IoT-WLAN since they use a LAN domain with a wireless connection. The traditional network concept is defined by the architecture based on the TCP/IP protocol which most networks use and within this the different network domains, in addition to addressing, segmentation and routing. Networks with connected objects do not yet have standardised network domain policies and still follow the same conventional TCP policies. Networks with connected objects do not yet have standardised network domain policies and still follow the same conventional TCP policies.

2.4. IoT-Architectures

Currently, standardization organizations are working on the proposal of a standard architecture that is implemented all over the world. For this reason, research groups and manufacturers are still proposing architecture models that serve as a reference for a possible standard. However, although some proposals have implemented more than others, these are still in the development and testing phase. Some of these architectures are implemented based on the concepts and definitions previously described. For example, there is a European project called the Internet of Things Architecture (IoT-A), which also proposes, based on a concept, the theoretical basis for designing IoT architectures.

As long as there is no general agreement between all the actors and their proposals, the challenge will continue to be the unification of a reference IoT architecture. Creating a new proposal should take into account the previously mentioned concept aspects and the design of the architectures that will describe in this section. Then it will be possible to obtain an architecture that integrates the most relevant attributes of the various IoT architectures to build a hybrid or mixed architecture.

With this idea is intended that the new architecture can respond to the handling of events and times, that is, an architecture oriented to events, and based on the hardware capabilities of the elements that compose it. These events would respond to the integration of AI models to put available new services depending on the requirements of the network and users.

From the multiple proposals is derive a generic level-based model, starting from three-level models to five-levels. Then is presents the architectures based on processing and the proposals of architectures that are specific to the application level based on the concept of the framework. The following architectures are currently the most relevant [33]:

2.4.1. Level-based architectures

As there is no reference or standard model for IoT as is the OSI model for TCP / IP, most of the proposals deployed based on generic three, four, and five-level models. Which separate and organize the devices according to its function in different planes called layers. In these plans, roles and responsibilities defined independently and its functionalities can relate through interfaces.

2.4.1.1 Three levels architecture

This is one of the most knows and popular architectures in the different websites that talk about emergent IoT technologies. These refer to a typical system that is composed of three layers: detection, communication, and application, and were initially structured for the control and monitoring of sensors. This a basic architecture, and in some other sources, it is known by other similar names but with the same function, such as perception, network, and application. The objective is the same, actuators and control. Each layer is described below in order from bottom to top:

- Detection or perception level

It is the physical level, where the sensors capture the physical signals, transform these into information, collect it, and then send it. Each sensor as an end device responds to the requirements of the network layer according to its capabilities in terms of information protocols, interconnection technology, and topology.

- Communication or network level

It is commonly present in most IoT-Architectures that provide medium access control through a switch or a router for local interconnection and access to the internet. The configuration and requirements of the network will depend on the type of data and the function of the sensors.

- Application level

It is the level where the information is processed and depends on the use that gives to the data. Commonly it is associated with some type of server that hosts a service related to the monitoring and control of the sensors. Its location can vary between offering its services from the Internet as a cloud or something local that does not require an Internet connection.

2.4.1.2 Five levels architecture

This architecture, unlike the previous one, has the same layers, but with two additional layers distributed as follows: levels of perception, transport, process, application, and business.

- Process level:

This level separates the information processing function that was previously part of the application layer, leaving it as an independent layer. This level separates the information processing function that previously formed part of the application layer, leaving it as a separate layer. In this level, it is possible to store, analyze, and process large volumes of data (Big data) collected from the perception level. In addition, it is possible to manage data through databases and cloud computing.

- Transport level:

This level is similar to the communication or network level presented in the previous architecture, but with extended functions. In this level includes IoT interconnection technologies that are new or that have evolved from the best known. Such is the case of technologies such as RFID, Bluetooth, WiFi, and 3G/4G, which evolved improving their bandwidth, topology, and coverage. It is the intermediate level between the levels of process and perception through which transport the data of the sensors.

- Business level:

This level object to obtain an economic benefit from the increasing interaction of users with machines and from generating greater consumption of technology by people. The applications are endless in multiple fields and ecosystems, but this layer dedicated to the specific business model that generates profitability in each designed application. Some business models currently have incorporated the concept blockchain, oriented to the encryption of information, some more common applications are cryptocurrencies, database management, bank transactions, smart contracts, energy trading, and electronic signatures. This level uses its own protocols for the transport of information; however, there is a recent open-source proposal is the IOTA protocol. This protocol has its own architecture (Tangle) and offers better performance than blockchain, but it is still under development and testing.

2.4.2. Computing based architectures

In addition to the previous architectures, others show a clearer interaction with the Internet and data processing. The connection of objects to the internet and the possibility of establishing the processing in a shared infrastructure depending on the advantages of its location gives way to the following architectures.

2.4.2.1 Cloud computing

The infrastructure that processes the data is in the cloud and organized in three layers, level 1 the devices as the data source, the process level is the intermediate or central level, that is, the cloud, and in level 3 the applications.

When it was possible to connect to the Internet with better bandwidth, and when applications began to require more storage space, it was then that companies and individuals decided to migrate their storage infrastructure from local to virtual servers managed by third parties through internet providers (ISP). This practice became more and more common, giving force to the massive storage of information in high-capacity virtual media called the cloud. Now, the cloud is essential in the work done with IoT; since, in addition to offering data storage services, it allows hosting platforms to do its processing.

This architecture model allows a direct connection to the cloud through any Internet access or with access from the same device. Moreover, it can access the consultation, exchange, download, and storage of the data (processed or unprocessed) from any place or machine. The versatility described in this last type of access is a quality that gives rise to the concept of ubiquitous computing. Therefore, the National Institute of Standards and Technology (NIST) of the United States of America [34], defines this architecture through recommendation SP 800-145. Which presents cloud computing as a model that allows network access under the concept of ubiquitous computing with on-demand processing. In turn, it shares resources such as applications, storage, and services, quickly with a minimum of effort on the part of the ISP. This model is exposed through three principles: essential features, service models, and deployment models.

The following three essential features described thus:

- **On-demand self-service:** Clients can reserve and do use of the server and store information automatically when these need it, without human interaction with each ISP.

- **Broad network access:** Accessibility to cloud capabilities can be done from any type of client (mobile phones, tablets, laptops, and desktops) using any conventional means of communication available to users.
- **Resource pooling:** All the provider's resources are pooled and made available to multiple users, dynamically assigned according to demand. The user is unknown the location of these resources. However, know some of these resources (storage, processing, memory, and network bandwidth) and their use.

Service Models:

Initially, a cloud was considered a place or storage space; Later, as it evolved, it became a resource management method in IT information technologies over time.

The term "Cloud Computing" is currently a technological trend that reflects any action involving providing content services within the Internet, limiting its charge only for "what is consumed." These services are divided into three main categories:

- Software as a Service (SaaS).

Model in which a cloud provider manages infrastructure and platform components and put to available to users.

- Platform as a Service (PaaS).

It facilitates companies and industries' development, execution and management of applications (increasing their development speed, since they allow programming at a higher level), more simply and flexibly, at a much lower cost than that of "maintaining some platform directly within a facility.

- Infrastructure as a Service (IaaS).

This model provides companies with the way to host IT infrastructures and access to computing, storage, and network capacities, always in an ascending way. Its subscription models enable IT to cost savings. In this case, the user is solely responsible for administering a specific operating system and related platforms for this purpose, while the responsibility to support and maintain the infrastructure falls directly on the provider.

2.4.2.2 Fog computing

Considering that the infrastructure responsible for data processing, when it is in a fog or a "cloud closer to the ground", under the perspective of the "fog", its processing is carried out directly between the Smart Objects, without needing to send them to the cloud, whose functionalities range from data processing and decision-making. This type of architecture is open and standard (essential for development in the IoT ecosystem), unlike closed and private solutions.

- SmartFog

This architecture can take computing beyond where the devices are or where the sensor network is located, having the ability to adapt according to the application's needs. The “fog nodes”, unlike how they are perceived in a network or electrical circuit, is a dynamic system capable of allowing applications and functionalities to be implemented in different layers within itself.

- Cisco fog computing

For Cisco, fog computing (seen as “a cloud closer to the ground”), is a platform that has a high level of virtualization, capable of providing computing, storage, and networking services between the “Big Data” (located in the cloud) and “Smart Objects”.

2.4.2.3 Edge computing

This type of computing occurs, either in the user's physical location or where the data source is. As a result of this, users obtain faster and more reliable services, having, in turn, the possibility that companies can distribute resources to different locations.

2.4.2.4 Mobile edge computing (MEC)

“Multiple Access Edge Computing” amplifies your capabilities in the cloud when you take it directly to the edge of the network. As computing occurs on remote servers (away from the user and the device), MEC makes the processes take place in base stations in the network, reducing latency and congestion in mobile networks.

2.4.3. Framework based architecture

It reflects the encapsulation of a basic set of practices and requirements which are used by devices or artefacts that delimit the architecture of a given system.

2.4.3.1 OCF Architecture

It is based on the REST architectural style, being resource oriented. Additionally, OCF specifies functional interactions such as CRUDN, messaging, maintenance, and monitoring. Additionally, this architecture has a resource for assigning the AllJoyn interface and the OneM2M module.

2.4.4. REST architecture

The term originated in 2000, coined by Roy Fielding (creator of the HTTP protocol). It reflects a software architecture style, designed for “hypermedia” systems, called or distributed as World Wide Web (W.W.W).

2.4.5. HTTP Proxy Service (HPS) architecture

Taking into account the concept of proxy as an "intermediary" between requests for resources made by a certain client to a certain server, on the web said proxy must be used with the HTTP protocol, providing a "cache memory that can be shared to web pages and all the contents that have been downloaded, this is called: proxy-cache; As all users share this memory, access times are improved.

2.5. M2M Interconnection

In the classic Internet model, where servers, routers and various clients interact with each other, direct M2M communication was already known, referring to the solutions that allow said communication between devices (machine to machine) of the same class and specific application, highlighting the sensors (which detect changes in their environments and communicate through electrical impulses) as the main elements within this technology, and always connected at all times to a wireless network.

M2M generates a base level in communication to be used by the IoT; This type of intercommunication is taken into account when considering any technology capable of allowing different devices that are in the same network to share information mutually and to carry out actions completely autonomously and independently, without the help of another external machine or human operator.

2.6. IoT Interconnection Technologies

IoT stores an immense amount of data due to many connected devices in a diverse variety of connectivity technologies such as:

- Wired
- Short-range wireless (wifi, Bluetooth, Zigbee and Z-Wave)
- Long-range wireless (cellular technologies such as GSM, LTE, LTE-M, NB-IoT, EC-GSM-IoT and 5G)
- Low power (unlicensed such as SigFox and LoRA)
- Satelite

The current 5G network significantly reduces the delays in interface communication, becoming super reliable and low latency (URLLC). However, it cannot improve the "quality of status update," causing control and performance to depend mostly on wireless communications.

2.7. IoT-Protocols

They allow communication between IoT devices with the network; they guarantee that all "Smart Objects", gateways, or services, can come to understand all the information sent by other objects.

In the "application interface", it will find:

- Advanced Message Queuing Protocol (AMQP)
- Restricted Application Protocol (CoAP)
- Data Distribution Service (DDS)
- Message Queue Telemetry Transport (MQTT)
- HTTP Restful

At the transport level there are:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

At the network level, there are:

- IP
- 6LoWPAN

At the data link level are:

- IEEE 802.15.4
- LPWAN

On the physical level, there are:

- Bluetooth Low Energy (BLE)
- Long-term evolution (LTE)
- Data transmission in proximity (NFC)
- Power Line Communication (PLC)
- Radio Frequency Identification (RFID)
- Wi-Fi / 802.11
- Z-Wave
- Zigbee

2.8. Router

Among all the network equipment necessary to form LAN and WAN networks, the router stands out for this specific case, which is key in interconnecting a network. This equipment, clearly initially developed to operate under TCP / IP architecture, have provided networks with the ability to provide IP addresses and route packets between different networks, in addition to providing Internet access. Before, these networks were exclusive to companies or providers of communication or Internet services, and they were reaching homes little by little. With the evolution of electronics and the need to improve services at the residential level, given the growing demand for the use of the Internet in homes, the costs of this equipment were decreasing. This meant the possibility of residential internet access through routers with the capacity limited to the type of service needed. Although this is a Layer 3 device and is named after business routers, its functions are not the same, and it is used in home environments as a more basic version. The best-known cases are routers with unlimited local telephone, television and internet access.

Just as these devices were adapting to the needs of these services, new needs were also generated with the arrival of new IoT technologies. Some of these IoT routers, as shown in Figure 4 and Figure 5, were created to connect everyday Things on the same network but under the dominance of a single type of manufacturer. Some things with the ability to communicate that have already been introduced in the market are available for users to purchase using an additional router. Although it can be a low-cost device, one of the difficulties faced with this type of solution is the dependence on a single type of manufacturer, which makes the user must buy things only from a specific brand. This would limit the user in choosing it and would take away the option of acquiring other types of Things with better functionalities offered by other manufacturers.

In the same way that the router is created for business networks, IoT routers also offer different services with good technical capacity but at very high costs. These routers try to respond to cover the most used IoT interconnection technologies, even if they do not cover other needs. Therefore, the need for interoperability in the IoT ecosystem becomes evident as a goal to which an IoT router must respond. In addition, these routers mostly have only ethernet and WiFi interconnection technology, which makes it difficult to communicate with objects that have other types of technologies.

The idea is to bring together in a single routing device both needs, interoperability and low cost, and the possibility of expanding its capabilities to the point of allowing the reprogramming of its functions. This, together with artificial intelligence, can lead to a Smart IoT-Gateway, such as the one proposed in this thesis.

2.9. IoT-Gateway

The IoT-Gateway is the device that is expected to replace the conventional routers used in proximity networks. An approach to this device has already been given through gateways sold by some manufacturers' brands to connect their objects to traditional networks as an extension of the router on the local network. However, it still lacks features to become an IoT-Gateway. Some of these features are resource sharing, data storage, multiprotocol, and multi-connection, manageable and secure. This thesis uses a prototype that allows data storage and internet connection management through an AI. This thesis uses a prototype that allows data storage and internet connection management through an AI. Also, to be the device responsible for centralizing network information, it maintains its organization through the proposed architecture. In this case, the AI of the IoT-Gateway is responsible for segmenting the network into workgroups, resource sharing and routes the objects using the mechanism for selecting the best node.

A gateway, in general, is a device that is used as a connection point between one network to another using different protocols and architectures. Although it can be used the same as the router, it can communicate different IoT environments, protocols, and architectures. That is, the local or main network does not need to use the same communication protocol as the internal or external networks. This device can act as a translator between protocols carried from within the network over different connections and send it over others outside the network. However, depending on the network layer and the environment, it can also be used for different functions. Among them, the simplest is the outgoing connection or internet access.

The main difference between the router and the gateway is that the gateways are used in many IoT networks, from corporate environments to home environments, but with more advanced functions than a conventional router.

2.9.1. Centralized Management

Of the three types of the grade of centralization explained above, centralized management is the main requirement of the architecture of this thesis. Therefore, the function of a device that performs this type of operation will be briefly described.

Considering that there is currently a large increase in the number of interconnected smart devices, the vast majority still have low storage, processing, and connection capacities, which requires reprogramming and updating. Therefore, a network management system is necessary, centralized, and synchronized with the applications in the IoT devices, which allows managing the sensors used by the IoT devices easily.

2.9.2. Gateway agent

When there is no physical device, that is, hardware dedicated to gateway functions, a software agent can be used, which for this case is the virtual or digital representation of the physical device. This can be housed within any other physical entity with sufficient hardware characteristics to emulate the operation of a Gateway. Each agent has specific tasks; in this case, the IoT agent within the gateway assumes the translator role between specific protocols. This can be housed within the hardware of a Gateway or directly within the end device (sensor); however, it maintains its dependence on the local network router in any of these scenarios. In the proposal of this thesis, the router and the gateway are integrated into a single device together with artificial intelligence, which manages internet access and proposes a different way of segmenting and routing things connected to the network.

2.10. IoT-Platforms

These components are considered the most important within the IoT ecosystem. They are used for data processing, data collection, visualization, and management in data devices, allowing interaction between devices and applications to transmit information through the standard protocols existing on the Internet.

The main applications that facilitate the development of projects in IoT are:

- Azure
- Zetta
- Arduino IoT Cloud
- ThingsBoard
- ThingSpeak
- Thinger.io
- Nodo-RED

2.11. IoT security

Consumer applications aimed at the control of everyday things by users under the IoT concept and the desire of manufacturers to accelerate their acquisition has put security aside. Few are IoT objects that are safe; they are still working on a security scheme that allows users to give peace of mind when taking an IoT product to their homes. The vulnerability of the information is high before a model of commercialization of Things connected to the internet, without having previously gone through arduous standardization processes or being under architectures with good security schemes.

2.12. Artificial Intelligence in IoT (IoT-AI)

AI is a data analysis model that can automate complex processes where traditional algorithms cannot work due to large volumes of data processing. AI establishes patterns and generates actions from these patterns using different learning models. AI is a transversal system for all applications in different areas of knowledge, including IoT. The data obtained from connected objects in IoT has been little explored since this data is not always available to users for analysis. However, this thesis analyses the data obtained from connected objects through the proposed architecture, giving functionality to the entire IoT-Network.

2.12.1. Machine Learning (ML)

2.12.1.1 Selection of the ML algorithm

Some AI techniques like Machine Learning can be chosen for an application according to its dataset size, training speed, interpretability, tuning [35]. Below is a brief description of each of them:

- Dataset size

ML algorithms depend on the size of the data set. However, no specific rule indicates which algorithm should be used according to its size (less than 50 MB or more than 1 TB). Table 1 shows the algorithms that are used according to the amount of data. Therefore, these are the algorithms that can be used considering that the sample dataset is balanced [36].

Table 1. ML algorithms accord dataset size

Small	Medium	Large
Decision trees Linear models (including logistic regression and linear discriminant)	(Nonlinear) SVM Naïve Bayes Nearest neighbor Neural network (shallow)	Deep nets Ensembles

- Training Speed

Training speed is the time it takes for a model to build and train to solve computational problems. Some factors such as algorithm architecture and time complexity can affect the speed at which the model will be trained. Table 2 shows the algorithm according to its speed if the system is required to be fast in training [37].

Table 2. ML algorithms according to their training speed

Very fast	Moderately fast	Moderately slow	Very slow
Decision trees Linear models (including logistic regression and linear discriminant) Naïve Bayes	Ensembles Nearest neighbor Neural network (shallow)	(Nonlinear) SVM	Deep nets

- Interpretability

ML models can be unintuitive and difficult to understand since the best models are usually the least interpretable. These models look for patterns through relationships within a high number of variables in a high-dimensional space. Therefore, they can identify relationships and information that are very difficult to capture and interpret by a human being. Therefore, when applying an ML model in real problems, it must choose between two aspects, accuracy or interpretability. The simple model is less precise but easily understood, or the complex model is very precise but less interpretable. Table 3 shows the classification of some algorithms according to the level of difficulty in interpreting them [38].

Table 3. Interpretability

Easy to interpret	In the middle	Difficult to interpret
Decision trees Linear models (including logistic regression and linear discriminant)	Nearest neighbor Neural network (shallow) Naïve Bayes	(Nonlinear) SVM Ensembles Deep nets

- Tuning

The tuning is used when changing the parameters or hyperparameters that modify the operation of a model, improving its result. Some algorithms do not have many parameters or hyperparameters, which limits the optimization of the model. Therefore this tunning can determine the performance and precision of a model and affect its optimization. Table 4 shows how to choose the model for training based on the amount of tunning required to achieve the desired result [39].

Table 4. Tuning

Minimal	Some	Lots
Linear models (including logistic regression and linear discriminant) Nearest neighbor	Decision trees (Nonlinear) SVM Ensembles NaÏve Bayes Neural network (shallow)	Deep nets

2.13. Conclusion

The previous chapter shows a literature review of the concepts, technologies and architectures most commonly used in IoT and related to AI. Without the previous review of these concepts, the proposed architecture would not have been possible. After comparing and analysing in-depth the existing works, the solution proposed in this thesis will be presented in the following chapters.

Chapter 3. Proposed interconnection architecture

3.1. IoE concept applied to this architecture

This architecture integrates the Cisco IoE concept from a Fog and Cloud computing architecture with a higher degree of troubleshooting at the Fog level. However, it also takes the Edge computing concept applied to objects, since if they can host an AI, it will store and process its data to provide and improve services directly on users. Outside the main network, the Edge Movile Computing concept is applied, which maintains a connection with the user's main network and groups all the objects belonging to the same user. In the same way that this proposal is conceived with integrating the most popular architectures, it is done with the concepts, such as the concept according to IETF integrated with the Cisco IoE concept and others.

The objects connected in the network can be of any type, with any IoT connection and IoT protocol. The topology and coverage do not imply a problem for the network either, since each object is connected following the order of the architecture in Figure 8, and artificial intelligence is in charge of assigning it a work role within a group and one architecture layer. In this way, a control is maintained in the network organisation, and each object takes on an identity depending on its features.

The fusion of these four architectures and the inclusion of AI for their administration gives rise to an IoT-SmartArchitecture, presented as a proposal in this thesis.

3.2. Centrally managed architecture

The degree of centrality for the case of this architecture is located in the IoT-Gateway, which allows concentrating the data and the data flow in a single point. Although some objects or sensors can connect directly to the cloud and work in cloud computing architecture, the idea is that they follow this architecture and always report their data to the IoT-Gateway.

3.3. Architecture description

We propose a grouping model for different types of objects that attend the same service type to a user. The grouping is carried out by an AI hosted within an IoT-Gateway and is based on ML techniques to analyze the data extracted from each object connected to the network.

The AI controls the algorithms that make the network operational and interconnects with the other AIs located in the different architecture layers through an interface. This interface is described in the operation of the architecture on which this proposal is based. The workgroup creation algorithm, which uses ML, analyzes the features that define an object's functions and capabilities. With this information, the ML classifies an object and assigns it within a workgroup according to the type of service it can collaborate and assigns it a role within an architecture layer. All objects are different in terms of capabilities and functions and can provide different services and share different resources; therefore, an ML can help predict which group an object can belong to and in which layer should be assigned.

3.4. Architecture components and their functions

This IoT-Architecture with AI (IoT-SmartArchitecture) and centralized management encompasses the most important aspects of Cloud, Edge [40] and Fog computing through SmartFog [41] and integrates them with other architectures to give place to smart objects in a IoT-Network. It is an architecture with additional functionalities that allows, through the fusion of different architectures [42], solving problems of adaptation and recognition of functions and services in connected objects. This architecture organizes objects in 5 layers with an ascending hierarchical structure in a tree or star between layers 2 to 5 and an ad-hoc structure in layer 1. With this structure, it is possible to propose new routing and grouping algorithms for a Smart IoT-Network, designed under this architecture.

Each layer has a predetermined functionality, so the AI assigns a role to an object according to the degree of coincidence of its features with these functionalities. It does not matter which layer an object is assigned, due to the flexibility of the architecture, the same structure is still maintained. The objects' features are directly related to the processing capacity, memory, and connectivity of the electronic device. That is, the assignment of the role is oriented to the hardware's capacity of the objects and the functions that they can perform.

According to Figure 8, this architecture comprises the next five layers: Internet, Management, Artificial Intelligence Assistant (AIA), Things, and Sensors. Each one of them is briefly described below in descending order. Moreover, shows the three processing levels (AI-Interface) over the three main layers.

3.4.1. Layer 5 (l_5)

Internet layer attends to queries from local IoT-Networks identified with the same user profile, that is, it can maintain a connection and provide service to the user inside or outside their home, making parameter groupings. The AI within Cloud assumes the user's local home network to use as the main network, and its functionality is based on maintaining and restricting access to queries, only to the AI of the IoT-Gateway of local networks, such as home, office, and any place of the city, connected under the same type of parameter [43], [44].

3.4.2. Layer 4 (l_4)

Management layer is the main layer of the architecture and is controlled by an AI within an IoT-Gateway. This a multi-protocol gateway with decision-making that also works with different interconnection technologies [45]. It is similar to the network layer that uses the other architectures, but with the difference that it is interoperable and intelligent, and it centralizes de information. Its main and most important function is to create workgroups and manage internet access. AI controls the algorithms that interact with the other AIs in the upper layer (Internet) and the lower layers (AIA, Things, Sensors) through interfaces. This interface is achieved using DFSP [45], a simple protocol that adapts to each IoT-Protocol payload's sizes.

When the AI does not know the IoT-Protocol, it consults the AI in the cloud for its structure, learns it, adapts DFSP to its payload size. In this way, the AI extracts the data from each object's features to learn it and classify it. When a new object is connected to the network, the AI does not know what type of object it is, so it evaluates it through its features and, depending on the nearness of these features with the objects established previously in the network, assigns it a role. With this role, the AI knows if the object is a Cloud, a Gateway, an AIA, a Thing, a Sensor, or an Actuator.

In other words, the device on which the Gateway is built under this architecture must have the capacity to control several algorithms such as routing and grouping, store, AI support, M2M broker support, and manage Internet access.

3.4.3. Layer 3 (l_3)

The task of AIAs is the usual and similar to virtual assistants. These capture command and execute it, but with the difference that it is no longer done under cloud computing architecture, that is, information processing is no longer done in the cloud. When the AIA receives a command to activate a object, it is first analyzed by the Gateway's AI, and based on its evaluation, it will decide if it is necessary to do a query or processing in the cloud or to process it locally. Moreover, to achieve that an object is actuated locally, the commands must be sent over some IoT-Protocol and perform an M2M connection. For this, the IoT-Gateway must have the ability to host a Broker that allows objects within the architecture to connect via M2M.

3.4.4. Layer 2 (l_2)

Things layer identifies objects of more complex capacity with diversified uses and physical structures designed for specialized tasks. As the case study of this work is on Smart Home, most of the objects are built from large capacity devices designed to operate as household appliances. These devices have good processing capability since they can support an AI and, in some cases, handle various types of connection. The AI in these devices has two fundamental tasks: learning from the usage habits of its user and reporting the changes and features to the central AI. Most of the time, objects classified as Things contain sensors for their operation, which does not imply dividing the object into two layers. However, if the Thing is a device responsible for collecting information from sensors in a system, the architecture does divide them.

3.4.5. Layer 1 (l_1)

Sensor layer is a layer that contains sensors and actuators or objects with both functions. In this layer, these objects can capture analog and digital signals and transmit it, or even depending on their ability, can process them. Most of these final objects perform type ON / OFF or data streaming actions and can operate directly by connecting to the IoT-Gateway or the Cloud. However, according to this architecture, its connection point must only be the layer immediately above (layer 2). The communication between them keeps under the same M2M connection policies that the IoT-Gateway. However, the connection may not support an IoT-Protocol. Therefore, to use an end-to-end IoT protocol in this architecture, it is recommended to use, e.g., MQTT-SN, in this layer [46].

3.4.6. Processing levels (L)

This architecture operates under three processing levels integrated and communicated through an interface. Level 1 begins where the data is generated through objects with AI (SmartThings), which process or pre-process information about the user’s daily activity. Level 2 or central, is the IoT-Gateway with AI (SmartGateway), which centralizes, processes, and classifies the information. Level 3, the cloud with AI (SmartCloud), processes high volumes of information and compares and relates them to those of other clouds. These processing levels allow the separation of functionalities, decrease latency, and reduce the response time when there is a loss of communication with the cloud. This processing distribution gives autonomy to the SmartThings to process data from Sensors and its users' direct obtained. That, in turn, lowers Level 2 latency. Moreover, it is possible to extend this autonomy if the central AI allows a partially distributed management under its supervision. That is, things are directly connected to other things without the intervention of the SmartGateway. That also lowers Level 2 latency. However, this work will not use this type of connection.

3.4.7. AI-Interface

This interface is made to transport the information required by the AI between 3 different processing levels and coordinate their priority. These levels are communicated through DFSP messages, specifically the COMPUTING message. Every time a user uses an object, the internal AI stores the information (Level 1) and processes them to statistically learn which resources are commonly used and what purpose they are used. It then uses the interface to send the information to the next level of processing. This information travels through the DFSP messages and allows the centralized AI to build a database of all connected objects' features through the following profiles of features, Functions, Services, Resources, and Capabilities. The centralized AI informs the connected objects using COMPUTING messages about the number of features and the order they should be sent.

3.4.8. Pre-processing

This event occurs before sending the features to the Smart Gateway. The features' data is organized according to the specifications of the central AI. Some of them are converted to binary data types to reduce the payload's size and sent in JSON format. Furthermore, there may be no coincidences between the manufacturer's features concerning those requested by the central AI. Therefore, it transmits the order of the most relevant features and a translation table. With this table, the text is processed, and the AI of the object only returns binary values corresponding to whether the feature exists or not. In turn, if the feature is a text value returns the corresponding number assigned in the table. The rest of the features with integer or decimal values return this value with the name feature associate.

On the other hand, the object's AI can learn which features have been accepted correctly in each request and improve this table. In addition to this, the object's AI can pre-process data related to user preferences based on its use and create different profile users.

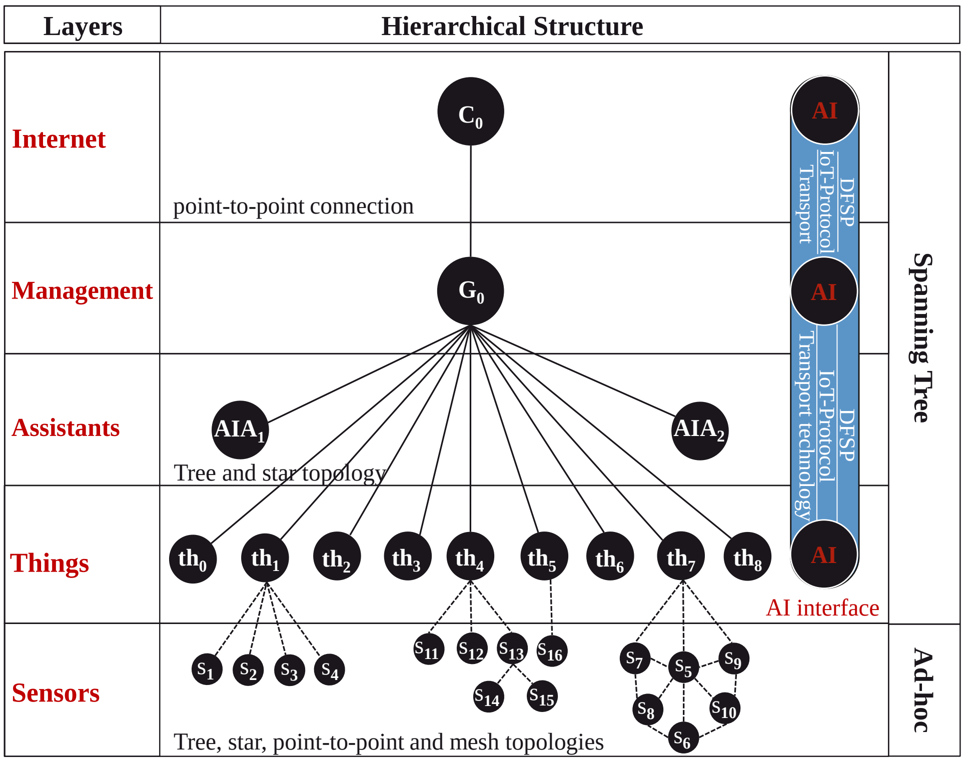


Figure 8. IoT Stack SmartArchitecture

3.5. Operation of the Architecture

This architecture is organized in five layers called the Internet, Management, Assistants, Things and Sensors with ascending hierarchy and tree structure. The network model of this architecture allows elements based on M2M protocols to be grouped and routed at the management layer level, and by groups of parameters, at the Internet layer.

Furthermore, this architecture is flexible and designed to allow artificial intelligence (AI) algorithms. The intermediary device is the management layer node that hosts an AI and manages Internet access.

AI performs troubleshooting locally through the central administration of the Gateway and the information provided by the group of things involved in the execution of a service. Managing this access is primarily based on reducing dependency on the cloud platform to solve a problem at the local network level. For example, the AI will decide when to query the AI in the cloud, give objects access to the Internet, or block any access until the problem is resolved. Thus, this carries with it good fault tolerance and improves network security.

3.6. IoT-Gateway

The current data networks allow to connect “Things”, but under their policies and protocols. Although protocols such as transmission control (TCP/IP) are flexible and allow transport to other protocols in the network, it is a low level protocol, which does not solve all the problems that appear in an IoT network [47]. Things must be managed differently and the resources that Things need are not the same as a conventional network.

Some IoT-Protocols such as MQTT can exchange messages on these types of networks, as well as the Constrained Application Protocol (CoAP) and Representative State Transfer (REST) [48]. MQTT and CoAP are M2M protocol and REST uses HTTP to perform operations between client and server.

However, the operation of these protocols is not designed to share, use or allocate resources on the network.

For that, is necessary a protocol or autonomous entity that allow to Things talk between them without human intervention, only machines. In this case, it has been decided to work with MQTT protocol [49], since is open source and allow modifying its operating mechanics. MQTT performs the M2M communication through a central server (MQTT Broker) inside a Gateway or directly through the cloud (Cloud Broker). This is a publication/subscribe protocol and was initially aimed at IoT sensors networks [50], since its main target was to optimize bandwidth and minimize the hardware and the processing [51].

However, is possible that when creating an adaptive algorithm, it is can take advantage of its own messages to send over they the new messages of the DFSP protocol designed for this proposal.

Another problem is that wireless routers in home networks are not flexible and do not allow reprogramming of protocols. Commercially this type of devices are obtained, but with very limited functions. To achieve modifications in the network and that Things operate under another type of architecture, it is necessary to change the traditional router to replace it with a reprogrammable one.

In addition, Things do not have the capacity to process information because their factory functions are limited and they cannot collaborate and interact with other things in a network. It is not enough to have a means of communication to the internet, it is necessary to increase the processing capacity.

From the above, it is evident that the devices that make up a conventional data network, such as the host, the router and the cloud platform, must be reconsidered.

3.6.1. Proposal

The next proposal consists of creating three control agents of the same type but in different environments. This control agent is the AI, which will be inside the smart things, in the Gateway with centralized administration and the platform in the cloud. The AI is based on the same principle and has two main functions, managing the algorithms based on Machine Learning techniques and learning the relations Machine to Machine (M2M) as the product of the joint work of a group of Things.

It is necessary to see the problem first since the perspective of the "Things", because these just work when its user controls it or if before it had a previous programming. For that, the things carry its own control, it is necessary to know that resource, function or service is shared between the things.

This paper addresses the study of the problem through the analysis of the exchange of DFSP protocol messages between smarts things, the gateway, and the cloud platform. In this way, it is can see how each Thing that connects announces its functions and services and shares it on the network.

On the other hand, the AI that administers the algorithm that assigns the role to each Thing decides that Thing is a resource or makes available a resource. This is an adaptive algorithm and changes according to the role that each Thing assumes within the group work.

3.6.2. Use of this architecture

To understand the architecture that will use in this proposal is fundamental to explain it through of three actors and its relation: Smart Things, the IoT-Gateway and the IoT-SmartPlatform, each contain an AI.

Through the communication of the AI between these three devices, it is possible to achieve a harmony of collective learning in the entire network.

The design of this architecture is thought for a Wireless Local Area Network (WLAN), connecting Things with different interconnection technologies and IoT-Protocols, depending on their use, bandwidth, processing capacity, and distance [52]. The idea is to modify a network home conventional and become it in a WLAN for IoT. The Figure 9, show an example of IoT-WLAN network, with different topologies and IoT-Protocols.

The connection of the Gateway to the intelligent platform in the cloud is done through the Internet and constantly monitors what happens in the house. If it is realized queries to the cloud, the platform validates the type of request, classifies it and decides if it is necessary to connect to other platforms else the AI into gateway decides if the problem can be resolved on the local network through the M2M connections [43].

3.6.2.1 Object + AI (Smart Things)

The importance of things acting intelligently, not just improves its internal functions, but it also allows defining functions and services that put to disposition to other things in the network.

Things, in this case study will be appliances inside a house. These require the ability to process and communicate with different communication technologies.

In the architecture of Figure 9 accord to Figure 8, it is can see how everything connects directly to other things or through the Gateway. This connection type will depend on the permissions and relations established by the Gateway.

For example, yellow line of Figure 9 depicts the relation M2M established for the gateway where creating a group with something in common.

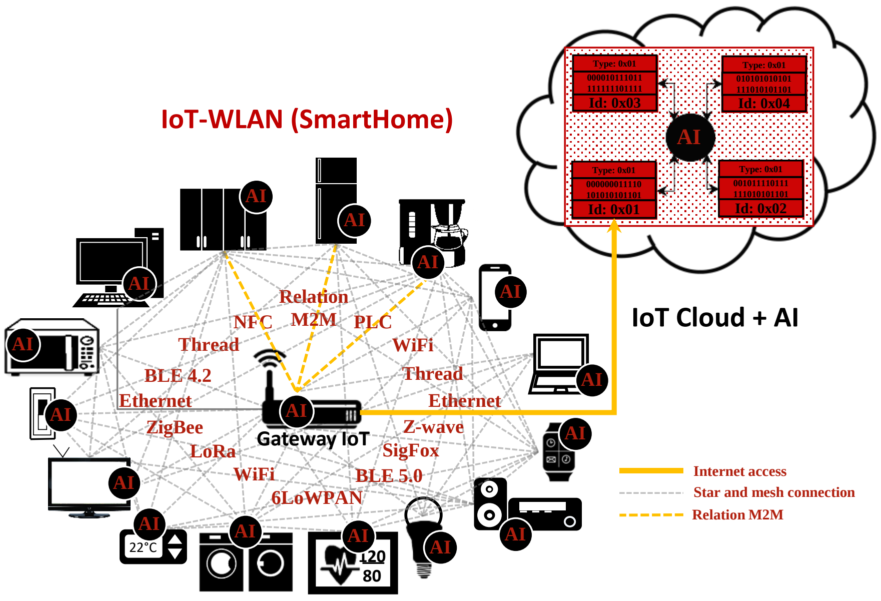


Figure 9. **Network IoT-WLAN architecture**

3.6.2.2 Router + AI (IoT-Gateway like G_0 node)

This multiprotocol device allows managing and centralizing all of the information regardless of the underlying technology of interconnecting. It carries out the management in a centralized way in the network and allows all the Things in the network to connect with each other, initially passing through it. In this way, the M2M relation are only made at the gateway level, that is, the central server (MQTT Broker) is not in the cloud.

For this reason, the AI learns about the relation that are commonly established between Things when a group of Things works together. Another indispensable resource is to manage access to the internet. The AI will decide whether the packets are forwarded within the same network or outside the network, based on the relation, the resolution of a problem and a clear reason for the requests to be sent out of the network to an IoT platform in the cloud.

3.6.2.3 Monitoring dashboard + AI (IoT-SmartPlatform)

Its function is to maintain network monitoring through the gateway and deliver information only when the gateway requests it.

With the information consulted to Internet through of platform in the cloud [52], the Things on the network can complement the local information and make a better decision.

3.6.3. Actors' integration

Using the architecture of Figure 9, all things are connected to each other through the Gateway (MQTT Broker + AI). Only the MQTT protocol will be used to establish M2M relations over WiFi connections. The AI that resides on the Gateway reviews and analyzes the payload of each MQTT packet continuously and creates statistics for each request. When it detects that there are continuous requests for common activities between the machines, it creates logical workgroups. In Figure 10, these three things highlighted in Figure 9 are isolated to explain their relation in more detail. In this case, it has not yet been established who or what resource is needed. This assignment, is the job that the adaptive algorithm of resource allocation controlled by the AI must perform, see Figure 11.

Figure 10 shows a group of Things doing a joint work. The M2M relation that the AI establishes depends on the functions and the service provided by the Things. For example, it could be assumed that something in common between this group of Things is the preparation of Coffee as a service to the User and each Thing can perform a function that helps with the provision of this service. Coffee is a resource that has the kitchen cupboard and milk is a resource that has the refrigerator. The AI activates the algorithm that is responsible for the allocation of resources and the DFSP protocol collects the information.

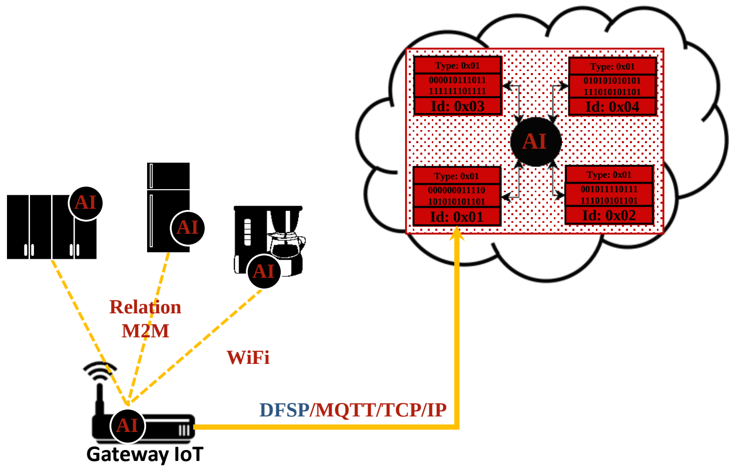


Figure 10. Allocation of resources by group of things

In Figure 11, the IoT-Gateway controls different types of algorithms through the AI and learns from them through the training matrix they provide. Each training matrix is a data file with information about the usage habits and how a machine has been used. This provides information to the system about what kind of operations the user usually performs in front of the machine.

Each petition or service is registered and then analyzed to obtain a statistical history of behaviour. After analyzing it with the AI algorithm, things send the resulting matrix to the centralized AI in the gateway. In Algorithm 1, is observe the general steps of the pseudocode of the control algorithm that allows the IA to delegate tasks.

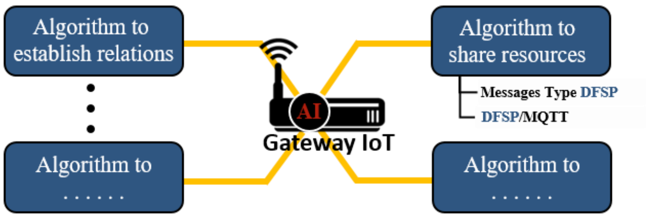


Figure 11. AI controlling the algorithms

Into the gateway, the AI can handle a number of algorithms and can accommodate as many as necessary and if the capacity of the machine allows it.

The algorithm "share resources" works with the information sent from the Things in the network through the MQTT payload. In the payload, another protocol has been specified that determines the type of information depending on the type of message, called DFSP.

The other algorithms provide information and perform tasks in parallel to the other tasks of the network depending on the decisions made by the AI. In this case, it will only control two algorithms, the algorithm to establish relation and share resource.

Algorithm 1: Control into IoT-Gateway

```
1.  Set an event listener (Connection request from Clients)
2.      Initialize MQTT_Broker
3.      Received updates on a subscribed topic
4.      Update the information of connected Things
5.      Activates the algorithms “establish relation” and “share resources”
6.      Supplies initial conditions to the AI
7.      Read training matrix of Things
8.      Set an event listener (MQTT_Broker)
9.      If MQTT_Broker is down then
10.         Disconnect from the Clients
11.     end if
12. End.
```

3.6.4. AI Operation

As noted in the Algorithm 1, the purpose of the basic algorithm AI of control in the network, is that of administering to the other algorithms that are executed in secondary and independent threads. Each algorithm is responsible for a different task but it depends on the problem that needs to be solved.

The basic target of an intelligent network is to be able to provide better attention to the user without the need for the latter to control the machines. The idea is that the machines do so autonomously through the AI.

In Figure 12, the AI is observed administering only two algorithms and learning from the user's habits when he is at home. When a problem cannot be resolved internally, the AI decides to search the Internet for possible solutions, learns it and saves it for future problems. The circle that indicates the AI in Figure 12 has the two algorithms that it controls on the sides. Below, the AI analyzes the MQTT payload with the information from the DFSP protocol that comes from Smart Things. Both at the gateway and in smart things AI are present. However, in the flowchart has represented as only one because both have the same information exchange technique.

The next thing the AI assesses is whether the information that comes through the DFSP is a resource or is a conversation between things. If it is about a conversation that tries to solve a problem, learn it and store it. If it is about a resource then it classifies and stores it. In both cases, the iteration is direct between things, but the information passes through the gateway. Unlike other architectures, the MQTT Broker is present in the Gateway and not in the cloud. This possibly decreases network delays, since for any operation at the local level, it is not necessary to go to a server in the cloud that is far away.

To control internet access, the gateway evaluates whether the requests for things merit it. If the gateway already has, this information previously stored or because you learned it before, then answer the request without having to go to the internet. The cloud platform delivers information requested by the gateway and monitors only what the gateway allows.

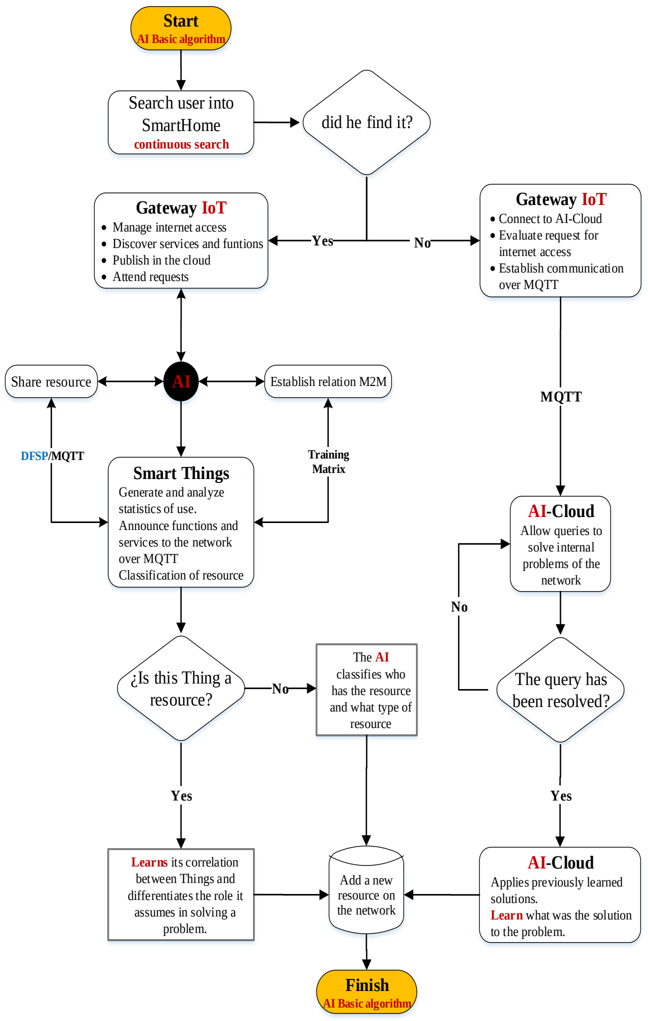

Start
AI Basic algorithm
Search user into SmartHome
continuous search
did he find it?
Yes
No
Gateway IoT
Manage internet access
Discover services and funtions
Publish in the cloud
Attend requests
Gateway IoT
Connect to AI-Cloud
Evaluate request for internet access
Establish communication over MQTT
Share resource
AI
Establish relation M2M
MQTT
DFSP/MQTT
Training Matrix
Smart Things
Generate and analyze statistics of use.
Announce functions and services to the network over MQTT
Classification of resource
AI-Cloud
Allow queries to solve internal problems of the network
No
¿Is this Thing a resource?
No
The AI classifies who has the resource and what type of resource
The query has been resolved?
Yes
Yes
Learns its correlation between Things and differentiates the role it assumes in solving a problem.
Add a new resource on the network
AI-Cloud
Applies previously learned solutions.
Learn what was the solution to the problem.
Finish
AI Basic algorithm

Figure 12. Flowchart of the IoT-Gateway + AI

3.6.5. Algorithm to share resources

All the algorithms that the AI manages reside in the IoT-Gateway and are based on computational learning techniques (Machine Learning). This algorithm is who allows the Gateway to respond to requests and announcements of resources through the DFSP protocol. The messages of this protocol are encapsulated within the MQTT messages. Its logic is fundamentally based on establishing the role that each Thing plays and the assigning of resources.

The algorithm must know how to differentiate a function or a service from a resource depending on the joint work performed by a group of Things. From this, another algorithm controlled by the AI determines the M2M relation of a group of Things. With this information of the groups created by the AI, the algorithm can be adapted dynamically according to the service required by a user.

3.6.6. DFSP Protocol

Function and Service Discovery Protocol (DFSP). This protocol was developed to be introduced within the payload of the PUBLISH messages of the MQTT protocol. That is, everything that is contained in this protocol is a string encoded in UTF.

The main function of this protocol is to discover the functions and services that announce the “Things” when connecting to the network. The algorithm that allows sharing resources in the network takes this information and decides between these two features, if a Thing can be a resource or have a resource. Once the function of this protocol has been defined, the types of messages, their formatting, and the message exchange rules are presented. The header and body of the message of this protocol are simple and like the MQTT header, the size is fixed. Figure 13 shows the protocol format with an example of the ANNUNCEMENT message that transport the FUNTIONS of a thing.

3.6.6.1 Header

It has a fixed size of 1-Byte with four fields, each of 2-bits (Message Type, Data Type, Flags, RESERVED). In the red box in Figure 13, highlight the protocol header and below is the body. The format of the messages is shown in 8-bit linear datagrams per n-Bytes (8-bits x n-Bytes) one above the other.

3.6.6.2 Body

The body carries the information of the type of message that has been defined in the header. This information is a string encoded in UTF, starting from the second Byte up to 250-MBytes. This information is concatenated through commas (,) to separate each of the items from functions and services from things.

3.6.6.3 Message exchange rules

The exchange of these messages depends on the mechanics of exchange of MQTT. The AI of each thing sends an ANNOUNCEMENT message over the MQTT PUBLISH message to the MQTT Broker.

Message Type	**Data Type**	**Flags**
0 - ANNUNCEMENT	0 - FUNTIONS	0 - Message sent
1 -DISCOVER	1 - SERVICES	1 - Message received

bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 1	Message Type		Data Type		Flags		RESERVED	
	0	0	0	0	0	0	x	x
Byte 2	"H" (0x48)							
	0	1	0	0	1	0	0	0
Byte 3	"E" (0x48)							
	0	1	0	0	0	1	0	1
Byte 2	"L" (0x4C)							
	0	1	0	0	1	1	0	0
Byte 2	"L" (0x4C)							
	0	1	0	0	1	1	0	0
Byte 2	"O" (0x4F)							
	0	1	0	0	1	1	1	1
Byte 2	"," (0x2C)							
	0	0	1	0	1	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Byte n	x	x	x	x	x	x	x	x

Figure 13. DFSP protocol format

This is a one-way message generated automatically by the AI every time a Thing is connected to the network. The identifier (Id) of the Smart Thing that was connected is found in the same PUBLISH message with the field named topic name.

On the other hand, the DISCOVER message is sent from the MQTT Broker every time a change occurs in the network or when the AI within of the IoT-Gateway needs to update information.

3.6.7. M2M Communication

Information of the start sequence of M2M communication algorithm is explained in Algorithm 2. The observation of the algorithm shows that, at the beginning a listener is provided to process connecting the IoT-Gateway then the start sequence algorithm allows the AI in the Gateway to establish relations between the machines. When a relation is created between MQTT client and broker, the AI creates a thread of independent processing for the Things to talk to each other.

Therefore, in Figure 14 depicts the exchange of messages between a group of things through the DFSP/MQTT. The IoT-Gateway is a common netowk system to share the things among the devices. It is also observed between the messages the control that one machine can have over another through an M2M protocol like MQTT. Therefore, the case is observed according to the figure, the autonomy that the machines can have around a common resource after the AI know through DFSP the functions and services of each

Thing. It can also be observed in the exchange of the messages as dipected in the figure, the control commands on/off of the machines.

The protocol messages create a table with the functions and services of each Thing in the IoT-Gateway that can be consulted by any algorithm and the AI.

Algorithm 2: M2M Relations Establishment

```
1.  Set an event listener (Process connect to IoT-Gateway)
2.      The AI takes control and manages the Things
3.      The AI initializes MQTT_Client
4.      The AI connects to the MQTT_Broker
5.      The AI subscribes to a topic
6.          The main topic is the name of the smart thing
7.      The AI publishes with the main topic
8.          Functions and services through DFSP
9.          Matrix training
10.     The AI evaluates which machines need common resources.
11.         If the machines are related then
12.             The AI directly controls the related machines
13.     If MQTT_Broker is down then
14.         Disconnect from the Clients
15. End.
```

3.6.8. MQTT PUBLISH/SUBSCRIBE Architecture

The MQTT architecture over the architecture of this proposal is simple.

The AI of each thing subscribes to the gateway (MQTT Broker) with its identifier (Id) and publishes its functions and services following the DFSP format.

In this architecture, all things are at the same level so it is not necessary to use a level separator (/).

It is could use something like /home/living-room/bulb or /home/kitchen/bulb. However, it this would only apply if all things are of the same type and therefore it is need to know their location. In this case, things are different and what matters is knowing their relation according to the work together. The syntax would be as follows: Topic/Payload/QoS. Then, the AI of each thing publishes the following structure: Id/Payload[DFSP]/0, every time it wants to talk to something else or control it.

In this structure, the cloud platform has not taken into account because communication between the gateway and the cloud have other structure.

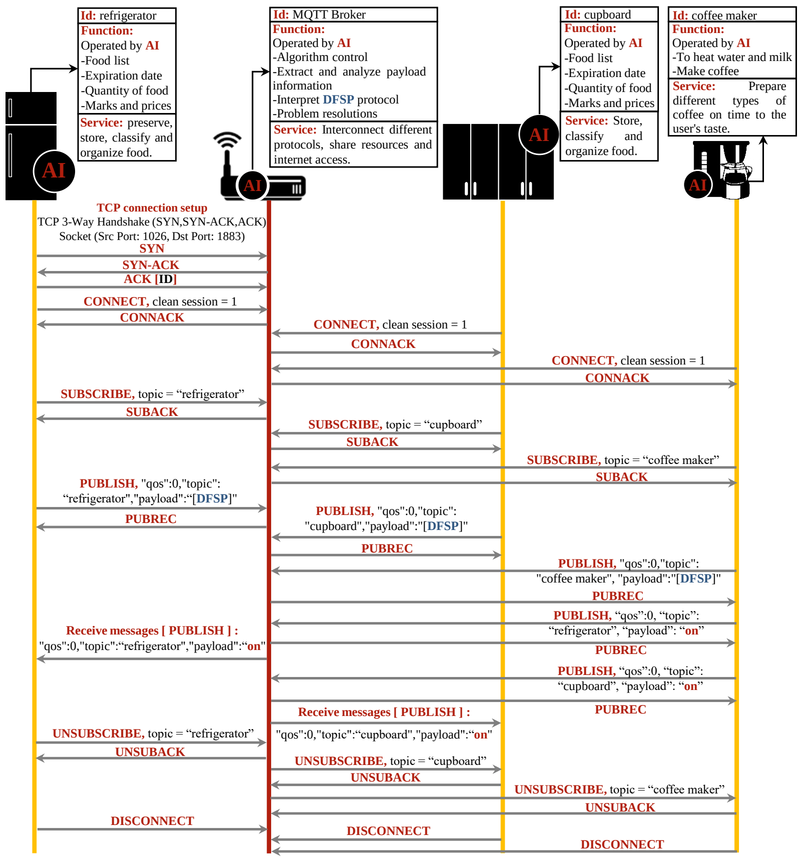

Id: refrigerator
Function:
Operated by AI
-Food list
-Expiration date
-Quantity of food
-Marks and prices
Service: preserve, store, classify and organize food.
AI
Id: MQTT Broker
Function:
Operated by AI
-Algorithm control
-Extract and analyze payload information
-Interpret DFSP protocol
-Problem resolutions
Service: Interconnect different protocols, share resources and internet access.
AI
Id: cupboard
Function:
Operated by AI
-Food list
-Expiration date
-Quantity of food
-Marks and prices
Service: Store, classify and organize food.
AI
Id: coffee maker
Function:
Operated by AI
-To heat water and milk
-Make coffee
Service: Prepare different types of coffee on time to the user's taste.
AI
TCP connection setup
TCP 3-Way Handshake (SYN,SYN-ACK,ACK)
Socket (Src Port: 1026, Dst Port: 1883)
SYN
SYN-ACK
ACK [ID]
CONNECT, clean session = 1
CONNACK
CONNECT, clean session = 1
CONNACK
CONNECT, clean session = 1
CONNACK
SUBSCRIBE, topic = “refrigerator”
SUBACK
SUBSCRIBE, topic = “cupboard”
SUBACK
SUBSCRIBE, topic = “coffee maker”
SUBACK
PUBLISH, "qos":0,"topic": “refrigerator","payload":“[DFSP]"
PUBREC
PUBLISH, "qos":0,"topic": "cupboard","payload":"[DFSP]"
PUBREC
PUBLISH, "qos":0,"topic": "coffee maker", "payload":"[DFSP]"
PUBREC
PUBLISH, “qos”:0, “topic”: “refrigerator”, “payload”: “on”
PUBREC
Receive messages [PUBLISH] : "qos":0,"topic":“refrigerator","payload":“on"
PUBLISH, “qos”:0, “topic”: “cupboard”, “payload”: “on”
PUBREC
Receive messages [PUBLISH] : "qos":0,"topic":“cupboard","payload":“on"
UNSUBSCRIBE, topic = “refrigerator”
UNSUBACK
UNSUBSCRIBE, topic = “cupboard”
UNSUBACK
UNSUBSCRIBE, topic = “coffee maker”
UNSUBACK
DISCONNECT
DISCONNECT
DISCONNECT

Figure 14. DFSP message exchange

This platform ha based on connections by group of parameters between other clouds using AI [43]. In both cases, the techniques are similar, local level the AI creates working groups and in the cloud is done by groups of parameters. Figure 15 shows the operation of the workgroup that attends the "coffee for the user" service, sharing and requesting the resources available to fulfil the service.

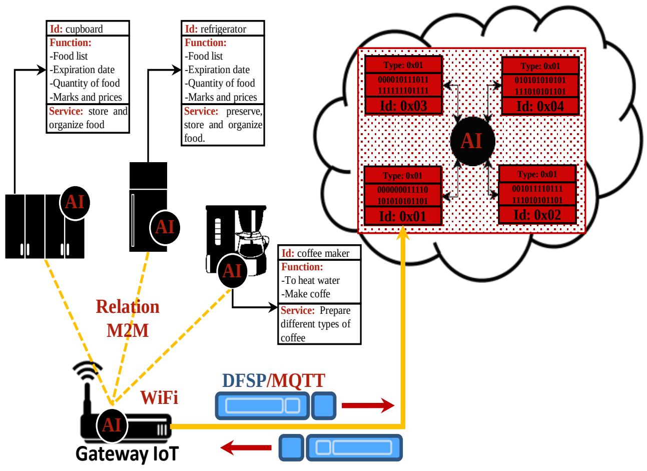


Figure 15. Operation of the proposal

3.6.9. Model for auto-tuning DFSP to IoT-Protocol

In this case, the MQTT Protocol is used, analyzing the Remaining Length (RL) algorithm, to calculate the size of DFSP that can be transported Figure 15.

This algorithm can also be understood mathematically knowing that the MQTT packet or message format consists of a Header (always present) + Variable Header (not always present) + Payload (not always present) [53]. The above depends on the type of message. The Figure 16 illustrates the MQTT packet format.

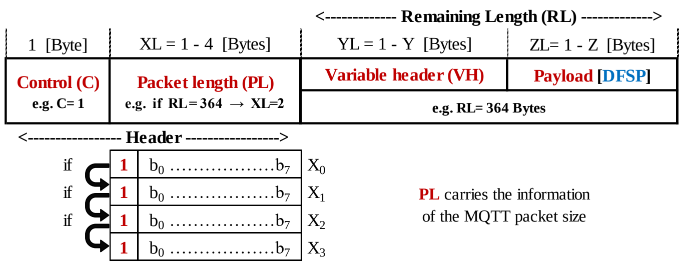

<------------- Remaining Length (RL) ------------->
1 [Byte]
XL = 1 - 4 [Bytes]
YL = 1 - Y [Bytes]
ZL= 1 - Z [Bytes]
Control (C)
e.g. C= 1
Packet length (PL)
e.g. if RL=364 → XL=2
Variable header (VH)
Payload [DFSP]
e.g. RL= 364 Bytes
<----------------- Header ----------------->
if
if
if
1 b0 ....b7 X0
1 b0 ....b7 X1
1 b0 ....b7 X2
1 b0 ....b7 X3
PL carries the information of the MQTT packet size

Figure 16. MQTT packet format

The MQTT packet size (PS) is calculated according to the type of message through the following Eq. 1.

$$PS = [C + XL] + RL, \text{Bytes} \qquad \text{Eq. 1}$$

The length of the Control field (C) will always be 1 Byte and if the value to be saved in the Packet length field (PL) is less than or equal to 127, then the header will have a fixed value of 2 Bytes. However, if the value calculated by the RL is greater than 127, the length of the PL field changes, modifying the total size of the package (PS).

The packet length field (PL) saves the size of the calculated packet and can occupy a length of between 1 and 4 bytes. To know how many bytes it is used and how is to save the size in this field, it used the following Eq. 2.

$$XL = \sum_{n=0}^{3} X_n * 128^n \qquad \text{Eq. 2}$$

XL is the calculation in Bytes needed to represent the number that corresponds to the value of RL, and that will be transported within the PL field located in the header. The representation of this number in Hex can occupy a space of the range between 1-Bytes to 4-Bytes. Since the length of this field is 8-bit linear, Xn is used to evaluate with the restrictions of the Eq. 3, if this number requires more than 8 bits to be represented. For this, the 7 least significant bits (LSB) are used and bit 8 is left as carry to indicate the next field as shown in Figure 16.

$f: \mathbb{R} \rightarrow \{1, 2, 3, 4\} = \#Bytes$

$$XL = \begin{cases} 1, & \text{if } 0 < X_n \leq 127 \\ 2, & \text{if } 128 < X_n \leq 16383 \\ 3, & \text{if } 16384 < X_n \leq 2097151 \\ 4, & \text{if } 2097152 < X_n \leq 268435455 \end{cases} \qquad \text{Eq. 3}$$

The Remaining Length algorithm (RL) calculates the package size depending on the type of message, in some cases; the variable-header is not present. The equations for each of the cases is shown in Table 5.

Table 5. Equations of the Algorithm RL

Message MQTT	Remainning Length (RL), Bytes
CONNECT	$RL = 2 + (\sum_{i=1}^{n} ProtocolVersion[n]) + 1 + 1 + 2 + 2 + (\sum_{i=1}^{n} ClientID[n])$
CONNACK	$RL = 2$
PUBLISH	$RL = 2 + (\sum_{i=1}^{n} TopicName[n]+Payload[n])$
PUBACK	$RL = 2$
SUBSCRIBE	$RL = 2 + (\sum_{i=1}^{n} TopicName[n]) + 1$
SUBACK	$RL = 2$
UNSUBSCRIBE	$RL = 2 + (\sum_{i=1}^{n} TopicName[n])$
UNSUBACK	$RL = 2$
DISCONNECT	$RL = 0$

The minimum packet size (MIN-PS) is just 2-Bytes and the maximum packet size (MAX-PS) is 256-MB. E.g. the DISCONNECT message is 2 Bytes. Like the existing condition in the MQTT Broker, if the Client ID contains more than 23 characters, the broker responds to the CONNECT message with a CONNACK return code 2: Identifier Rejected [32]. A condition was created to control the minimum and the maximum message size. Considering that XL = 4-Bytes for any PL between 2-MBytes at 256-MBytes and the MAX-PS is 256-MBytes. Then the payload of a PUBLISH message that corresponding to DFSP is 250-MBytes. PL will carry the information obtained from the calculation of PS = 1M-Byte + 4-Bytes + 250-MBytes without exceeding the MAX-PS. Therefore, the calculation of RL does not affect the DFSP protocol because it already takes into account in PS.

3.7. Clustering

As a case study, the behavior of connected objects within a house is analyzed through how they connect and share information between them and Internet. Currently, a user can acquire everyday objects for their home based on electronic devices with the ability to process and communicate. However, the service they provide to the user through their main function is basic, independent, and isolated, and not based on cooperative behavior. An extension of its main function is limited to its control and monitoring over Internet or through a local connection. For example, an IoT coffee maker is an object with the ability to prepare coffee and connect to Internet, but it depends entirely on the user's control, it is not autonomous, nor does it work collaboratively with other objects within the house. Most of these objects work in the same way, although commercially, the arrival of objects based on high-capacity devices that include AI (Smart Things) is expected. Nevertheless, the conventional network of a house limits them since the router is low capacity. Moreover, the architecture on which networks are currently based is not intended to transport the information required by an AI between connected everyday objects.

In other words, IoT objects are not joined to meet the needs of a user through the same service automatically. They work independently and through different types of commands for their activation. An IoT object does not auto-identify or auto-classify, neither assume work roles in the network according to its function, nor create automatic relations with other objects. Much of this problem may be due to the element that performs the network administration, in which, for this case study, it could be due to the router. This network element is a device that only performs two main tasks; it processes automatic network addressing and Internet access, although it allows connectivity between devices on the same network. It is a limited capacity device with low processing and memory level, it does not store high volumes of information, and it is not reprogrammable. Another additional problem is that it only offers two interconnection technologies, WiFi and Ethernet. Regarding the architecture, it is important to mention that, although up to now it has been possible to work with traditional architectures to interconnect IoT-Networks, such as Transmission Control Protocol (TCP / IP), there is no standard architecture to do so. There are some proposals, but none of them is designed to provide interfaces between layers to an AI or handle multiple connections of different technologies or multiple IoT-Protocols. That is, there may be a cloud that supports AI and objects that support AI, but there are still neither Gateway devices on the market that supports AI, nor an architecture to allow their integration.

3.7.1. Collaborative workgroups

The literature reviewed in section 2 shows that some group-based networks are grouped according to something in common. E.g., groups of nodes share the same type of resource, function, relations, topology, data, interconnection technology, bandwidth, route, capacities, parameters. In other words, this type of groupings only select elements of the same type. In the case of grouping by services, an element of a group could not contribute to other groups because they do not have the same service. On the other hand, in this proposal, each object is analyzed and classified within a group if the features related to its resources, capacities, and functions, can help other objects to fulfill a service. Moreover, they can collaborate in one or several workgroups. In other words, this proposal, based on collaborative workgroups, are groups of objects that share a common work to serve a user. In this sense, the AI can decide which group to assign an object to and coordinate its intervention to collaborate within the group, measuring its percentage of participation and its degree of importance. The work to be done by each group is previously characterized by the IoT-Gateway's AI to fulfill a service. These services are defined from interprets' the object's AI's data collected about its users' lifestyle habits. Our goal is to automatically let any new object join the IoT network and serve the users without their intervention, just Plug-and-Play (PnP).

3.7.2. Network Model

Let U be the universe of Smart IoT-Networks (SmartHome, SmartOffice, SmartFactories, SmartCity), connected through clouds. According to the design of Figure 17, we will model the case of Smart Home. Where B and C are two disjoint sets (different networks) connected under the same architecture. The B set spans layers 1 to 4 of the architecture while C set is located in layer 5.

The B set is defined as a Wireless Local Area Internet of Things Network (IoT-WLAN) like a Smart Home or any other like a Smart City with this same architecture, which can be defined and mathematically modeled as follows. Let $B = (W, \lambda, E)$ be a network of connected objects (shown in Figure 17), where W is the set of objects, λ is the set of their capacities, and E is the set of links between objects. All this collection of sets and subsets are distributed in the different layers of the architecture (shown in Figure 8).

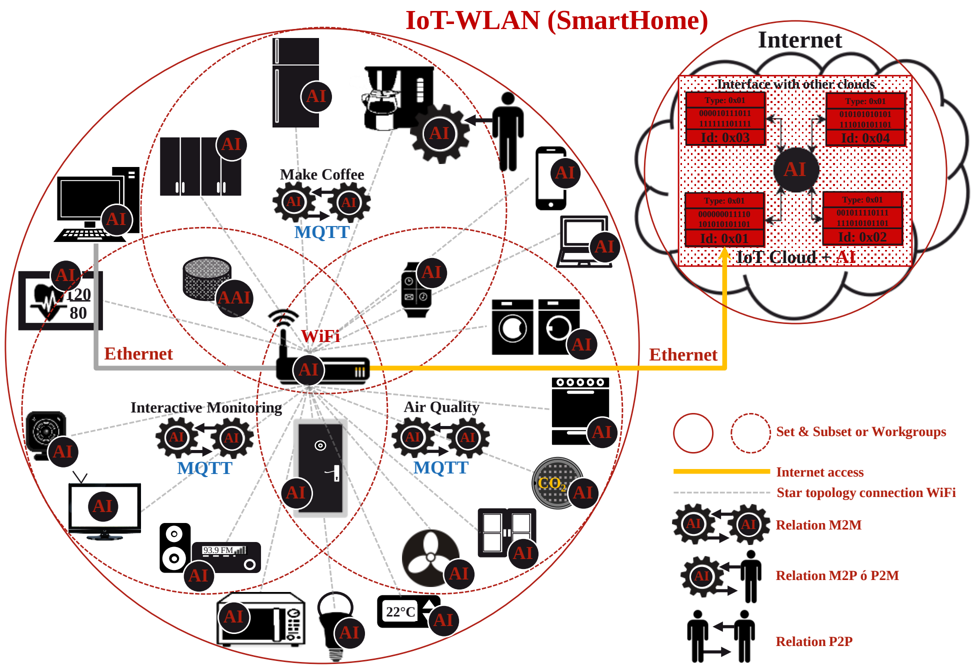


Figure 17. Architecture of smart IoE proximity networks

Figure 18 shows the connected objects according to the network shown in Figure 17. Nodes are grouped into sets and subsets. Here, W = {b_0, b_1, b_2,…, b_n}, where b_i represents the *i-th* object before assigning it a workgroup. W_i represents the subsets called workgroups and W^c the nodes that are not within a group. b_i could be in two different groups, then the workgroups are not disjoint sets.

Considerations:

1. (n + 1) it is the total number of objects since b_0 is the only IoT-Gateway node within the network, and where all other nodes are connected to. Then the equation given for W is shown in $W = (\bigcup_{i=1}^{m} W_i) \cup W^C$ Eq. 4, where m is the total number of subsets.

$$W = (\bigcup_{i=1}^{m} W_i) \cup W^C \qquad \text{Eq. 4}$$

It can also be noted that:

$$n + 1 = |W^C| + |\bigcup_{i=1}^{m} W_i| \qquad \text{Eq. 5}$$

As the IoT-Gateway node (b_0) will be part of all subsets W_i, it is clear that:

$$\left|\bigcap_{i=1}^{m} W_i\right| \geq 1 \qquad \text{Eq. 6}$$

2. Initial working conditions are pre-established from the design shown in Figure 17. It assumes that there are already three workgroups and that the objects are organized in the architecture shown in Figure 8. Under these conditions, it is possible to characterize each of the objects and structure a dataset used to find the best classifier.

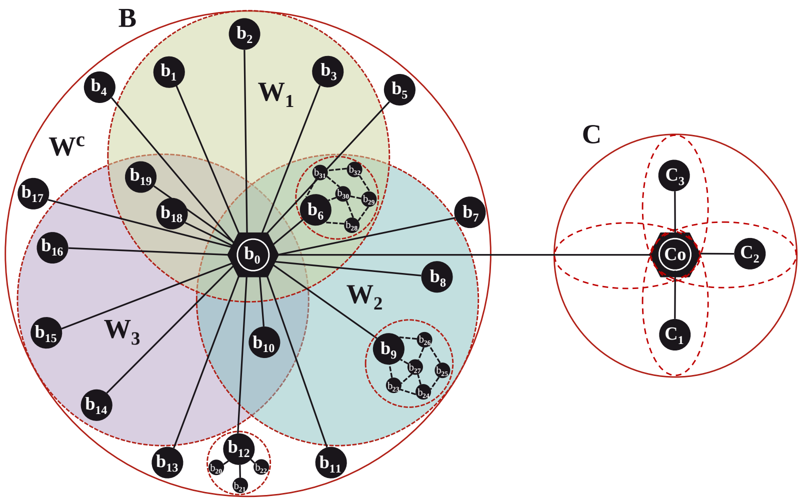


Figure 18. IoT-Network model workgroups-based

In Figure 18, these groups are made up of heterogeneous objects (b_i) called nodes and whose only similarity will be based on the degree of collaboration to attend a service. The dataset obtained contains the features of each node's functions and services previously classified within a workgroup and with a role assigned according to the architecture layer. With this dataset, when entering a new instance, b_i will be assigned as C, G, AIA, Th, or S according to the layer, and to a w_i according to the workgroup.

Whereby is defined:

- IoT-Gateway: $b_0 = G_0$, keeps network management.
- $e_i \in E$, where e_0 is the connection between set B and C.

• Objects connected in the network:

- nodes: $b_i \in W = \{b_1, b_2, b_3, \dots, b_n\}$.
- feature: $f_i \in F = \{f_1, f_2, f_3, \dots, f_m\}$.
- function 1: $f(W) = \{f: W \rightarrow \{0, 1\}\}$.

• Register of features in a dataset:

- class: $\mathscr{C}_j \in \mathscr{C} = \{\mathscr{C}_1, \mathscr{C}_2, \mathscr{C}_3, \dots, \mathscr{C}_k\}$.
- dataset: $d_i \in D = \{d_1, d_2, d_3, \dots, d_p\}$, where $d_i = \{f_1(b_i), f_2(b_i), f_j(b_i), \dots, f_m(b_i), \mathscr{C}_j\}$.
- function 2: $f(D) = \{f: D \rightarrow \mathscr{C}\}$, such that each d_i is assigned to a class $\mathscr{C}_j$.

F is defined as a finite set of features related to a node and predefined in a dataset structure. Therefore features of a node (b_i) are obtained through the feature function f: $W \rightarrow \{0, 1\}$, where $f_j(b_i) = 1$, it means that it has these features, otherwise $f_j(b_i) = 0$. The d_i indicates the *i-th* node (b_i) data register, each characterized with $f_j(b_i)$ and using to $\mathscr{C}_j$ as class features.

To achieve that an ML model assigns a node to a group, and a layer through its features, it is necessary to find an ideal way to classify it through a function f: $D \rightarrow \mathscr{C}$ such that each $f_j(b_i)$ is assigned to a $\mathscr{C}_j$ class.

f: $D \rightarrow \mathscr{C}$ could be a K-Nearest Neighbor (K-NN), a K-Means, a Neural Network, a Multi-Layer Perceptron (MLP), a Decision Tree (DT), a model based on Discriminant Analysis, or a Gaussian Naive Bayes (GNB) or any other.

Given a collection of registers of D, each register contains a set of variables (features) that define a profile and will be denoted by $X_p = [f_1(b_i), f_2(b_i), f_3(b_i), \ldots, f_k(b_i)]$, with an additional variable (feature) of class that will be denoted by $y = [\mathscr{C}_j]$. Therefore, the register could be rewritten as $d_i = \{(X_1, \mathscr{C}_1), \ldots, (X_i, \mathscr{C}_j), \ldots, (X_n, \mathscr{C}_n)\}$.

In a K-NN classifier, dataset D_X^K, has been classified previously utilizing like training matrix (X_{Tn}, $\mathscr{C}_{Tn}$) through the class labels called workgroup and layer. This classification is based on the distance between the K neighbors with similar features. When a new instance enters the system, it is assigned to a group and an architecture layer. The standard distance for this classifier is Euclidean (d_e), but others can also be used depending on the data type. In the case of boolean values, it is possible to use the Manhattan distance (d_M) and others as Chebyshev distance (d_{Ch}) and Minkowski distance (d_{Mk}).

If a node's features are present in one or more sets, it uses the simple matching coefficient (SMC) or the Jaccard index (I_J). If there is a node at the intersection between sets, its features (functions, services, resources, and capabilities) are shared. SMC encodes 1 and 0 if one feature is present or absent in both sets, while I_J only encodes when the feature is present. For binary data types 1 and 0, SMC is the most appropriate as it obtains a better measure of similarity and more computationally efficient.

Once the system has trained with N cases for D_X^K, the ML can predict how it will classify the new instance depending on the distance of its features between K neighbors.

Figure 19 shows how some nodes of Figure 18 would look organized by groups and layers.

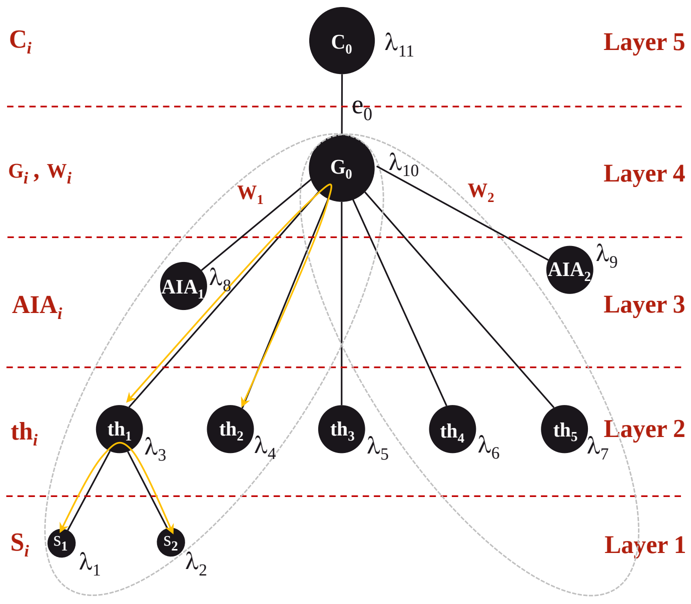


Figure 19. IoT-Network model organized in workgroups and layers

3.7.3. Algorithms implementation

Below are the algorithms implemented in the simulations and the time to compute it. Each algorithm and procedure preexisting in [54], [55], was taken and modified using the previous equations.

The running time or time complexity is estimateed based on the system runs any of these algorithms [55]. This running time is shown in Table 6.

Table 6. ML Techniques Time Complexity

Classifiers	Time complexity
K-NN	O(n·d + n·K)
K-Means	O(n·K·d)
Radial Basis Neural Networks (RBNN)	O(n·K·D)
Support Vector Machines (SVM)	$O(n^2 \cdot \|F\|)$
Decision Tree (DT)	O(|X|·|F| log |F|)
Gaussian Naive Bayes (GNB)	$O(\|X\|\cdot\|F\|) + O(\|\mathscr{C}\|\cdot\|F\|)$

For the proper run of the algorithm, it is necessary to consider that there is a dataset structure with four feature profiles (X_p) in the IoT-Gateway (G_0). In other words, all node features are divided into four profiles. Each profile is delimited by a specific number of features and a classifier class. These are extracted from the node through the

ANNOUNCEMENT or DISCOVERY messages, using the FUNCTION, SERVICES, RESOURCE, and CAPACITY profile sub-messages. The data obtained through each message is organized and stored in the dataset.

The total structure of a dataset register would be as follows, $\mathscr{d}_i = \{(X_1, \mathscr{C}1), (X_2, \mathscr{C}2), (X_3, \mathscr{C}3), (X_4, \mathscr{C}4)\}$. To realize the grouping of the objects and classify a new object in one of these groups, only necessary to work on a dataset made up of the profiles of functions and services. So the dataset for grouping only uses the first two segments defined like this: $\mathscr{d}_i = \{(X_1, \mathscr{C}1), (X_2, \mathscr{C}2)\}$ and for routing the last two segments defined like this $\mathscr{d}_i = \{(X_3, \mathscr{C}3), (X_4, \mathscr{C}4)\}$. The class label for $\mathscr{C}1$ is “workgroups”, and for $\mathscr{C}2$ it is “layer”.

Figure 20 shows the flowchart of the ML classifier. The G_0 establishes an event listener, waiting for the connection of a new object. When there is a “new object” within the network, this must announce its features using the “features number” and the “features sequence” that the AIs have previously exchanged. In turn, it activates a timeout of 10 sg. If the new object is not announced, the G_0 sends a DISCOVERY message. In both cases, the messages extract the object’s features, and then the ML processes them. Once it completes the features’ register, the ML classifies the object into a workgroup and a layer.

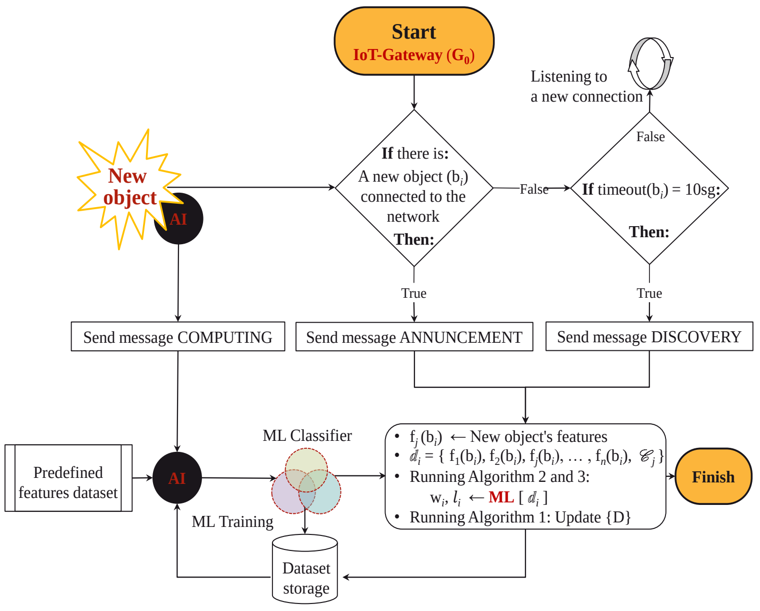

Start
IoT-Gateway (G_0)
Listening to a new connection
New object
AI
If there is: A new object (b_i) connected to the network Then:
False
If timeout(b_i) = 10sg: Then:
True
True
Send message COMPUTING
Send message ANNUNCEMENT
Send message DISCOVERY
Predefined features dataset
AI
ML Classifier
ML Training
Dataset storage
• $f_j(b_i) \leftarrow$ New object's features
• $\mathscr{d}_i = \{ f_1(b_i), f_2(b_i), f_j(b_i), \ldots, f_n(b_i), \mathscr{C}_j \}$
• Running Algorithm 2 and 3:
$w_i, l_i \leftarrow$ **ML** $[\mathscr{d}_i]$
• Running Algorithm 1: Update {D}
Finish

Figure 20. Flowchart of the ML classifier

The following training methodology was used for each ML classifier. It was taken n = 54 different objects and features about their functions and services were extracted. X_1 = 20 for the functions profile and X_2 = 10 for services. Then they were classified into three test groups (m=3) with ($\mathscr{C}_1$) and in an architecture layer ($\mathscr{C}_2$). That is, the input values of the classes function are initially set by default. The values of $\mathscr{C}_1$ are set as {1, 2, 3} representing {w_1, w_2, w_3} respectively. The $\mathscr{C}_2$ values are set to {1, 2, 3, 4, 5} representing the layer {l_1, l_2, l_3, l_4, l_5} respectively. With this dataset previously-stored and predefined, each ML is trained.

Table 7 shows the parameters used to train each of the ML classifiers. Depending on the classifier, the parameters change and are adjusted according to its operating structure. However, some parameters are common to all classifiers, like input nodes, workgroups number and iterations' number.

Table 7. Parameters used to train different classifiers

Classifiers	Parameters n = 54, m = 3, Iterations' number =100
K-NN	K=3, metric=‘chebyshev’, n_jobs=100
SVC	gamma=2, kernel=‘rbf’, probability=True, C=1
GP	1.0 * RBF(1.0)
DT	max_depth=5
RF	max_depth=5, n_estimators=10, max_features=1
MPL	alpha=1, max_iter=1000
AB	n_estimators=50, learning_rate=1.0, algorithm=‘SAMME.R’
GN	var_smoothing=1e-09
QDA	store_covariance=False, tol=0.0001

Data preprocessing and standardization of the dataset are very important since it will get less accurate predictions when using a machine learning estimator. In the K-NN's case, the scaler used was “MinMaxScaler” the rest of the classifiers used “StandardScaler.” Table 8 indicates the parameters' notation be used in the algorithm and its corresponding meaning.

Table 8. List of parameter' notations and its meaning

Notation	Meaning
n_neighbors (K)	Number of neighbors.
metric	The distance metric to use for the tree
n_jobs	The number of parallel jobs to run for neighbors search.
gamma	Kernel coefficient.
kernel	Specifies the kernel type.
probability	Probability estimates.
C	Regularization parameter.

RBF	The kernel specifying the covariance function.
max_depth	The maximum depth of the tree.
n_estimators	The number of trees in the forest.
max_features	The number of features to consider.
alpha	Regularization term.
max_iter	Maximum number of iterations.
learning_rate	Learning rate.
SAMME.R	Real boosting algorithm.
var_smoothing	Variances for calculation stability.
store_covariance	Storage of covariance matrices.
tol	Absolute threshold for a singular value to be considered significant.

As shown in the flowchart, when starting the central AI, this already contains an ML that has been previously trained with a predefined dataset, which was selected for its high percentage of accuracy in the tests. This ML is ready to classify a new object by features in a workgroup (Algorithm 4) and an architecture layer (Algorithm 5). Then, it is updating the new information in the dataset (Algorithm 3) and stores it.

Algorithm 3 updates the dataset's information hosted in G_0 when a new node (b_i) is connected, or due to a change in the network, e.g., a node is turned on or off. Therefore this algorithm, each time a DFSP message arrives, updates the dataset.

Algorithm 3: Updating dataset {D}

```
Input:   DFSP messages over any M2M protocol.
Process:
1.  Update dataset {D}.
2.  Function     ← ANNOUNCEMENT [$X_1$, $\mathscr{C}_1$ ]
3.  Services     ← ANNOUNCEMENT [$X_2$, $\mathscr{C}_2$ ]
4.  D ← { $\mathscr{d}_i = X_1 + X_2$, $\mathscr{C}_1 + \mathscr{C}_2$}.
Output: Updated dataset for two feature profiles {D}.
```

Algorithm 4 classifies a new instance d_i within a group w_i in the network through any classifier from Table 1 or any other with high accuracy. This algorithm begins by reading the predefined table and training the ML. Read the first two profiles from the dataset and use the class feature to start the training. Then the dataset is divided into training and tests, taking 75% and 25%, respectively. The next step is to normalize the data and use an ML model. The model with the highest accuracy will be selected for further training and implementation. It will evaluate the features of b_i and assign it to a workgroup (w_i).

Algorithm 4: Creating collaborative workgroups (AI)

```
Input:   dataset { D }, $d_i$
Process:
```

```
1.  Read and load to { D }.
2.  X   ← [$X_1 + X_2$].
3.  y   ← [ $\mathscr{C}_1$ ].
4.  Splitting the dataset into the Training set and Test set.
5.  x_train, y_train        ← 75% of the D for train.
6.  x_test, y_test← 25% of the D for test.
7.  Normalizer, scaler and transform the data
8.  x_train       ← scaler (x_train)
9.  x_test        ← scaler (x_test)
10. Set fitting training classifiers (Using Table 1).
11. Set the metric according to the selected classifier.
12. Calculate the predictor accuracy and error.
13. Print accuracy of the classifier on training and test.
14. While accuracy ≥ 80% do
15.     For i=1 to m do
16.             $w_i$ ← classifier.predict(X).
17.     End for
18. End while.
Output: New node assigned to a workgroup ($w_i$).
```

Algorithm 5, like Algorithm 4, classifies a new instance d_i in an architecture layer through any classifier in Table 1 or any other with high accuracy. Steps 1 through 14 are similar to Algorithm 4, except that use class $\mathscr{C}_2$. In step 16, the algorithm renames the identifier of the object according to the layer.

Algorithm 5: Allocation in architecture layer (AI)

```
Input:   dataset { D }, $d_i$
Process:
1.  Read and load to { D }.
2.  X   ← [$X_1 + X_2$].
3.  y   ← [ $\mathscr{C}_2$ ].
4.  Splitting the dataset into the Training set and Test set.
5.  x_train, y_train        ← 75% of the D for train.
6.  x_test, y_test← 25% of the D for test.
7.  Normalizer, scaler and transform the data
8.  x_train       ← scaler (x_train)
9.  x_test        ← scaler (x_test)
10. Set fitting training classifiers (Using Table 1).
11. Set the metric according to the selected classifier.
12. Calculate the predictor accuracy and error.
13. Print accuracy of the classifier on training and test.
14. While accuracy ≥ 80% do
15.     l ← classifier.predict(X)
16.     Switch( l )
```

```
17.         case 1:
18.             Rename($b_i$) ← S
19.         case 2:
20.             Rename($b_i$) ← th
21.         case 3:
22.             Rename($b_i$) ← AIA
23.         case 4:
24.             Rename($b_i$) ← G
25.       End Switch
26.   End while.
Output: New node assigned to a layer ($l$).
```

Figure 21 shows the DFSP datagram and the messages used in Algorithm 3 to update the feature dataset.

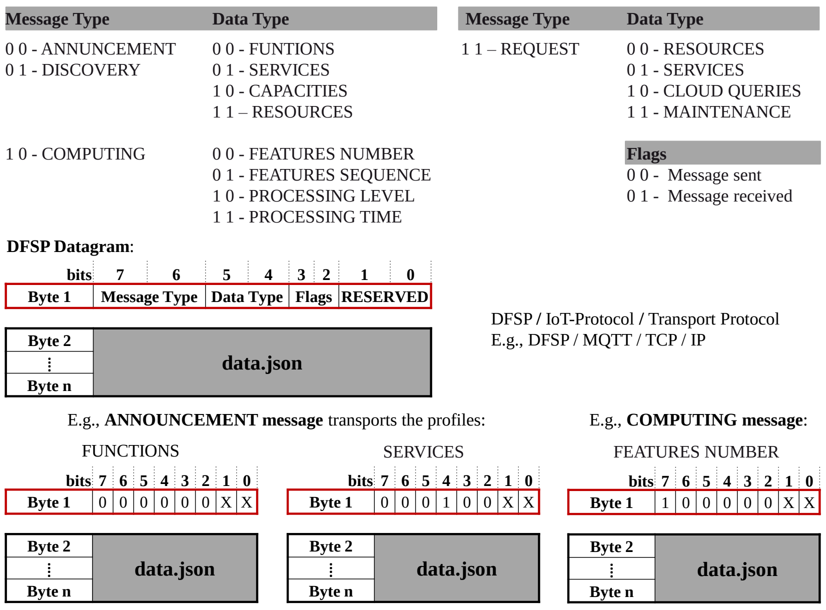


Figure 21. DFSP Messages

3.8. Routing

In the current IoT-Networks, when an object connects to the network, it does so through a standard wireless router. This device is not built to manage internet access; only it shares it. For this reason, each object is managed through the cloud and not locally. When performing a simple ON/OFF operation, the request goes to the cloud and returns, generating delays. Also, this device does not allow reprogramming a host or a broker, so an object with an M2M protocol based on MOM routing must go to the cloud to subscribe or publish. That makes the system's autonomy difficult since one object could not be related to another by itself. In this case, the user must do this task. It means that it is necessary to apply to the network's design a smart architecture with centralized management (IoT-SmartArchitecture) [56], in which the data processing is distributed between the cloud, the IoT-Gateway, and the final objects. Therefore, in this case, the conventional wireless router must be replaced by an IoT-Gateway [45] with the capacity to host a broker, store network information, and host an AI.

Once this architecture is applied to an IoT network, the AI can perform grouping and routing in the network. However, although the architecture already creates and manages workgroups, it also needs to route intelligently. Therefore, it is necessary to modify the current operation of MOM routing and manage it with an AI engine. The data processing starts at the management layer with the IoT-Gateway (Broker + AI), where one of the routing problems appears when using an event-oriented middleware through Pub/Sub since the requester node does not know the route of the subscriber nodes. A node cannot request a resource directly to another node since it does not know its resources (only the broker knows it). Thus, the node must request the resource it needs from the broker, and this it allocates the node with the resource. One of the most useful techniques to do this type of search in a dataset is searching by content (Content-based filtering), but this can give several results. Among these, it may get several nodes that meet the requirement, and only one is needed. In this case, the system must solve which node to choose.

This question is difficult to answer, implementing traditional algorithms since the decision criteria change according to the environment. Therefore, a more advanced system that adapts and learns according to these changes is necessary whereby an AI engine is the most appropriate solution. However, it is first necessary to know other network operation variables to adequately model the AI technique for the system.

3.8.1. Proposal model

The following proposal uses an AI engine to select the most optimal node with the necessary resources to render a service. Therefore, to find it, the system performs a content-based search to find the routes of possible candidates that meet the IA's minimum selection criteria. In this way, the AI decides which node is the most optimal to provide the resources that another requesting node needs without affecting the network's efficiency.

Based on the routing of Pub/Sub systems, the following solution is designed to take advantage of the routing tables created when an object is subscribed. The idea is to combine these tables with the dataset ($\mathfrak{D}$) created from the feature's profiles of each object's functions, services, capacities, and resources in the network. The central node or Gateway stores all types of change when an event occurs in the network (Event

Storage) and associates it with the object that generates it, merging everything in a single dataset.

Figure 22 shows an architecture with centralized management in an IoT-Network [56] with predefined workgroups as $w_i = \{b_1, b_2, b_3, \ldots, b_n\}$. The connected objects are represented as nodes, where b_i represents the ***i-th*** object before assigning it a workgroup and a layer.

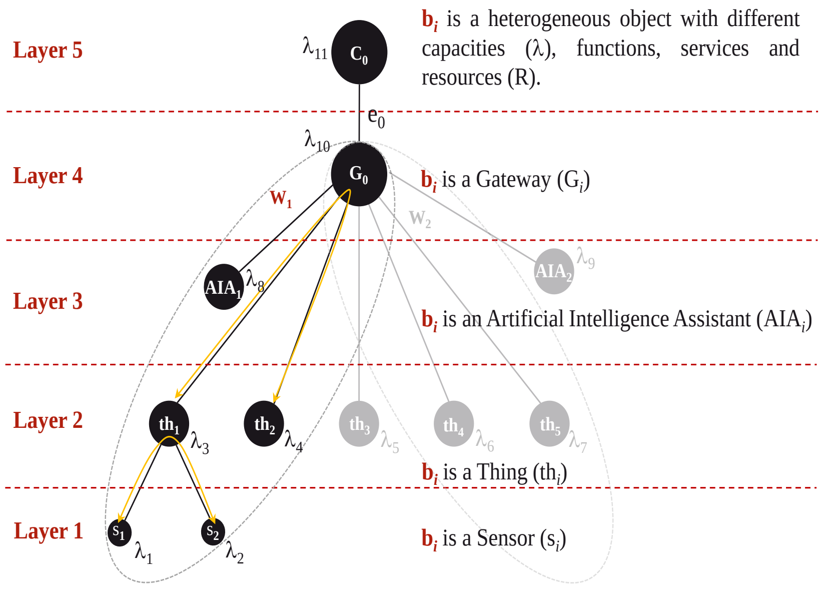


Figure 22. IoT-SmartArchitecture where b_i assumes a role per layer

$W = \{w_1, w_2, w_3, \ldots, w_n\}$, where w_i represents the sub-sets or workgroups, and W is the set of groups where the nodes belong. In turn, these workgroups offer services (one or more), represented like a set $S = \{s_0, s_1, s_2, \ldots, s_n\}$. Each service has a set of resources $R_S = \{r_1, r_2, r_3, \ldots, r_n\}$ necessary to offer that service. Within these workgroups, new nodes will be added, and each node announces what functions, services, capacities, and resources it has. Consequently, each node offers a set of resources R_N. Therefore, it is possible to say that S_i is the set of services offered by group *i*. If a group needs to offer a service, its nodes request the resources from the Gateway (G_0) to attend the service, and it evaluates its capacity (λ_i) to fulfil the service. Then, R_i can be defined as the set of resources that a node requires to collaborate on the service *i*, as shown in Eq. 7, where B_r defines the subset of nodes requesting resources.

$$\forall\, b_i \in B_r \;\; \exists \;\; R_i : \left\{r_i, r_j, r_k\right\} \mid R_i \cap R_N = \emptyset \qquad \text{Eq. 7}$$

In other words, for every node belonging to a workgroup, there is a set of resources that are needed to implement the service. E.g., given S_1 then $R_S(S_1) = \{r_1, r_2, r_3, r_4, r_7, r_8\}$, being b_3 a node that collaborates in the service S_1 and has the resources $R_N(b_3) = \{r_1, r_2, r_{10}, r_{12}\}$, therefore the set of remaining resources so that b_3 can implement S_1 is defined as $R_i(b_3) = \{r_1, r_2\}$. That means, to collaborate with this service, the node needs these resources beforehand. Eq. 8 shows the decision criteria to find a node b_o that belongs to the same workgroup w_i that b_r (the requesting node) so that b_r can implement the service S_j if the selection satisfies a criterion set ($\mathcal{K}$). For example, a criterion used in a conventional routing protocol to select the best route is to do it through the least number of hops or the minimum cost of the link. For this case, the routing algorithm finds the route by selecting the best node using $\mathcal{K}$. Criterion k_1 is based on finding which node has the requested resources but with fewer additional resources. In this way, the total amount of resources that b_i offers to implement other services is the minimum not to waste resources. This criterion is defined in Eq. 9.

$$b_o = b_i \in w_i \mid R_j(b_i) \ni R_N(b_r) \leftrightarrow K \qquad \text{Eq. 8}$$

$$|R_N(b_o)| < |R_N(b_k)| \ \forall \ b_k \in w_i \qquad \text{Eq. 9}$$

With the k_1 criterion of Eq. 9, the AI select the route if the condition is met that the number of resources offered by the selected node (b_o) must be less than the number of resources offered by any other node (b_k) of the same group (w_i).

However, the AI's inclusion allows the routing algorithm to have a set of criteria such as $\mathcal{K} = \{k_1, k_2, k_3, \ldots, k_n\}$. Therefore, the routing algorithm could choose another node as b_o, based on a different criterion that the one shown in Eq. 9.

Another possible criterion is to choose the best node with more energy available (k_2). Depending on the conditions, saving energy in the objective nodes can be a key factor. Therefore, being $\mathcal{E}_i$ the available energy of b_i, this criterion can be defined in Eq. 10.

$$\mathcal{E}_o > \mathcal{E}_k \ \forall \ b_k \in w_i \qquad \text{Eq. 10}$$

Other criteria could take into account network parameters. Criterion (k_3) in Eq. 11 and Eq. 12 consider metrics like packet losses of a node (L_i) and delay in the communication between one node and the IoT-Gateway, b_i and G_0 (D_{i0}), respectively.

$$L_o < L_k \ \forall \ b_k \in w_i \qquad \text{Eq. 11}$$

$$D_{or} < D_{rk} \ \forall \ b_k \in w_i \qquad \text{Eq. 12}$$

Regarding this network point of view, the connectivity and the bandwidth of the nodes can be the key factor to choose the objective node depending on the network status. Consequently, Eq. 13 and Eq. 14 defines this criterion (k_4):

$$|E_o| > |E_k| \ \forall \ b_k \in w_i \qquad \text{Eq. 13}$$

$$|Bw(e_i)| > |Bw(e_k)| \ \forall \ b_k \in w_i \qquad \text{Eq. 14}$$

Where e_i is the i-th connection of the set E= {e_1, e_2, e_3, …, e_n} of one node that have direct communication with G_0, and Bw(e_i) is the bandwidth of e_i. Therefore, the greater the number of connections in a node (e_N), the lower the possibility of being isolated, so the node is a better choice. Eq. 14 shows that greater bandwidth, better is the connection, and so the node is a better choice.

Finally, the criterion (k_5) can be the layer (l_i) where the node belongs. Therefore, the algorithm can take advantage of the architecture shown in Fig. 1 to select one node or another. This allows the routing algorithm to save the nodes with more capabilities if the state of the network requires it. On the other hand, the algorithm could select a node in an upper layer to get better performance. Consequently, Eq. 15 and Eq. 16 defines two different criterions that use this concept:

$$l_o > l_k \ \forall \ b_k \in w_i \qquad \text{Eq. 15}$$

$$l_o < l_k \ \forall \ b_k \in w_i \qquad \text{Eq. 16}$$

x_i represents all the features like resources ($\mathscr{R}$), capacities ($\mathcal{C}$), network parameters ($\aleph$) and architecture ($\mathcal{A}$) that a node has within the dataset ($\mathfrak{D}$) stored in the IoT-Gateway. Let $\mathfrak{D}$ be the structured dataset storage within IoT-Gateway. Therefore, each register in $\mathfrak{D}$ is a b_i node that comprises a series of x_i features segmented into profiles. E.g., the resource features profile $\mathscr{R} = \{x_0, x_1, x_2, \ldots, x_r\}$. Where each data has a value of 1 if there is a resource otherwise 0.

Function $f_1(x)$ in Eq. 19 is defined from the criterion of Eq. 9 that uses the matrix $\mathscr{R}$ to search for the b_o node with the requested resources. This could generate several results. However, the following functions allow choosing the most optimal one.

Eq. 17 and Eq. 18 define the cost equation and the capacity of a node. These equations are based on the objective functions, defined in Eq. 19 to Eq. 22.

Function $f_2(x)$ in Eq. 20 is defined from the criterion of Eq. 10 that uses the matrix $\mathcal{C} = \{x_{r+1}, x_{r+2}, \ldots, x_c\}$ to know which node has better energy availability. $x_g = x_{r+1}$ informs if the node can connect to the power grid and $x_a = x_{r+2}$ to an alternative energy source such as a solar panel. $x_b = x_{r+3}$ reports if the node has a battery and reports the amount of energy remaining in percentage between 0 and 100%.

Function $f_3(x)$ in Eq. 21 is defined as the link's cost between each b_i node and G_0, stored within IoT-Gateway. These data are stored like matrix $\aleph = \{x_{c+1}, x_{c+2}, \ldots, x_{\aleph}\}$ but only are used the Delay and Packet loss parameters. In the criteria of Eq. 11, Eq. 12, Eq. 13, and Eq. 14 are described its use and its relation in the cost's calculation. In conventional routing protocols such as EIGRP [57], the cost equation is used to choose the best route. In this case, the cost calculation is proposed similar to [58] to choose the best node b_o, as shown in Eq. 17.

$$Cost = integer\left(\frac{K_1}{\lambda \cdot l} + K_2 \cdot Delay + K_3 \cdot Loss + 1\right) \qquad \text{Eq. 17}$$

Where λ is the capacity of bi as λ(bi) and is defined in Eq. 18.

$$\lambda = 4\left(\frac{\sigma}{\sigma_{max}}\right) + 3\left(\frac{mP}{mP_{max}}\right) + 2\left(\frac{BW}{BW_{max}}\right) + \frac{e_N}{e_{N\,max}} \qquad \text{Eq. 18}$$

In Eq. 18, λ is defined as the node capacity and related to the features profile $\mathcal{C}$ and $\aleph$. The parameter $\sigma = x_{r+6}$ measures processor performance in Millions of Instructions Per Second (MIPS). Other parameters used for its calculation are the Bandwidth in Kbps as Bw = x_{c+1}, the number of connections as $e_N = x_{r+5}$, and the RAM defined as the processor's memory in GBytes as mP = x_{r+7}. Each of these values is normalized to its maximum value in order to handle orders of similar magnitudes. Each term has been given different weight according to the importance it is considered to have. An advantage of defining λ in this way is that it will never be 0, so the cost will not be infinite.

The presence of l_i in Eq. 17 stored in matrix $\mathcal{A} = \{x_{\aleph+1}, x_{\aleph+2}, \ldots, x_n\}$ allows major priority to nodes in higher layers making the cost decrease. The cost also depends linearly on the delay as $x_D = x_{c+2}$ and packet loss as $x_L = x_{c+3}$. K1, K2 and K3 have been chosen so that the three terms of Eq. 17 have similar weights. Finally, the value 1 has been added so that the cost is an integer greater than or equal to 1.

Function $f_4(x)$ in Eq. 22 is defined to select the node that consumes the minimum power. $x_P = x_{r+4}$ is the variable in the features profile $\mathcal{C}$ that contains the power values.

Table 9 shows the notation used in this model.

Table 9. Notations

Notation	Description
b_i	Node that represents the connected object.
b_r	Node requesting resources.
b_o	Node that selects the Gateway to provide the resources to render the service *i*. (objective node)
R_S	Required resources of a service.
R_N	Resources that a node has.
R_i	Resources required by a node b_r to provide the service *i*.
$\mathcal{E}_i$	Node's energy availability b_i.
e_N	Number of connections in a node b_i.
L_i	Average packet losses of b_i.

D_i	Average delay between b_i and G_0.
E_i	Set of b_i connections directly connected to G_0.
l_i	Layer where the node b_i is located.
$\mathcal{L}_i$	AI's interface processing level.
λ_i	Node's capacity b_i.

3.8.2. AI Technique

After defining the criteria, one needs to use an AI technique. In this case, a search engine is needed to find the best node (b_o) that satisfies the criteria of the requesting node (b_r). The IoT-Gateway knows the route and features of its subscriber nodes (b_i), whereby the AI engine of this proposal is based on a multi-objective optimization model using an evolutionary genetic algorithm. This algorithm used for the objective node's choice is called the Non-dominated sorting genetic algorithm (NSGA-II)[59]. It is a technique characterized by being fast and elitist. That is, it only considers the best solutions found during the search process. It has a time complexity of $(m\mathcal{Z}_{\mathfrak{D}}^2)$, where $\mathcal{Z}_{\mathfrak{D}}$ is the size of the dataset to be classified, and m is the number of objectives.

First, defining the problem must define which criteria will be optimized and then define them as the objective functions. The multi-objective optimization problem [60] can be formulated from several objective functions $f_m(x)$, which will depend constraints functions of $g_j(x) \leq 0$ and equality $h_k(x) = 0$. Therefore, the objective functions set is given by, $\mathcal{F} = \{f_1(x), f_2(x), f_3(x), \ldots, f_m(x)\}$ defined from $\mathfrak{D}$. Each deduced from the criteria of equations Eq. 9 to Eq. 16.

Let $g = \{g_1(x), g_2(x), g_3(x), \ldots, g_m(x)\}$ be the constraints function set.

Problem:

$$\min f_1(x) = \prod_{r=0}^{m} y_r \sum_{r=0}^{n} x_r \qquad \text{Eq. 19}$$

$$\max f_2(x) = x_g + x_a + x_b \qquad \text{Eq. 20}$$

$$\min f_3(x) = \mathrm{Int}\left(\frac{\mathrm{K}_1}{\lambda \cdot l} + \mathrm{K}_2 \cdot x_D + \mathrm{K}_3 \cdot x_L + 1\right) \qquad \text{Eq. 21}$$

$$\min f_4(x) = x_P \qquad \text{Eq. 22}$$

Subject to:

$g_1(x) = \prod_{r=0}^{m} y_r = 1$	Guarantee that the selected node has the required resource (R_i).
$g_2(x) = x_r \in \{0, 1\} \; \forall \, r$	Guarantee that (x_r) is binary.

Lower and upper variable boundaries:

$0 \leq x_r \leq 1$	$0 \leq x_g \leq 1$	$0 \leq x_D \leq 100$
$0 \leq y_r \leq 1$	$0 \leq x_a \leq 1$	$0 \leq x_L \leq 16$
	$0 \leq x_b \leq 1$	$0 \leq x_P \leq 4$

Such as:

x_r:	Node's resource features matrix.
y_r:	Matrix containing R$_i$.
x_g, x_a, x_b :	Available energy for node b$_i$.
x_L:	Average packet losses of node b$_i$.
x_D:	Average delay.

3.8.3. Algorithms implementation

The IoT-Gateway's AI administers $\mathfrak{D}$ by applying the NSGA-II algorithm from the search model of this proposal using the designed functions. Algorithm 6 shows how the data is loaded, and the most relevant variables are chosen since all the features sent from each node are not used, only those related to routing.

Algorithm 6: AI search engine using NSGA-II with Pymoo

Input: dataset $\{\mathfrak{D}\}$

Process:

```
1.  **Class** NSGA-II Multi-Objective Optimizer with Pymoo
2.  Read and load to $\{\mathfrak{D}\}$.
3.  $X \leftarrow [\mathscr{R} + \mathcal{C} + \aleph + \mathcal{A}]$.
4.  y  $\leftarrow$ [R$_i$].
5.  K1= 5, K2 = 1, K3 = 6
6.  **Class** Class RoutingM2M(Problem):
7.      **super**(n_var, n_obj, n_constr, xl, xu, type_var)
8.      **evaluate** ($X$)
9.      y = $x_0 \times x_1$
10.     σ = $x_{r+6}$/10000
11.     mP = $x_{r+7}$/32
12.     Bw = $x_{c+1}$/1000000000
13.     e$_N$ = $x_{r+5}$
14.     $l = x_{\aleph+5}$
15.     $x_D = x_{c+2}$
```

```
16.       $x_L = x_{c+3}$
17.       Calculate λ
18.       Minimize $f_1(x)$
19.       Maximize $f_2(x)$
20.       Minimize $f_3(x)$
21.       Minimize $f_4(x)$
22.       Constraints: $g_1(x)$
23.  problem = RoutingM2M()
24.  F, G = problem.evaluate($x$)
25.  pf = problem.pareto_front($x$)
26.  algorithm = NSGA2(pop_size, n_offsprings, sampling)
27.  res = minimize(problem, algorithm, (‘n_gen’, 10))
28.  opt = res.opt[0]
29.  X, F, CV = opt.get
30.  Print res.X, res.F, res.G, res.CV
Output: optimal $b_o$ = X
```

Algorithm 6 has several stages or steps to find the best node from a multi-objective optimiser, which selects the best node by calculating the optimum called X (capital letter) and which corresponds to the node b_o of this IoT-Network.

The first step is implementing the problem defined in the AI Technique through the RoutingM2M(Problem) class, where each of the functions and constraints is programmed using the previously structured dataset. In the super method, the number of variables, the number of objectives, the number of constraints, and the limits of the variables are defined. From each segment of the dataset, the variables used to calculate the functions according to equations Eq. 19, Eq. 20, Eq. 21, Eq. 22 of the model are obtained. These functions are evaluated with the x (lowercase) values through the evaluate (x) method.

The second step is the initialisation of an Algorithm (in this case NSGA2), where mainly the population size, descent, and sampling are defined. This method may contain other types of algorithms such as R-NSGA-II, NSGA-III among others [61].

In the third step, the minimise method is used to return the Result object (res), which contains the results when are executing the three Algorithm 6 methods: the problem, the algorithm, and the definition of a termination criterion.

Finally, the result (res) contains the best values found in the corresponding spaces. These include the feasible solution space, the objective space and the optimal (X).

3.9. Quality of Experience (QoE)

The Quality of Service (QoS) of the architecture is defined according to each IoT-Protocol used in communication. However, the QoE is analyzed using an Artificial Intelligence System for Multimedia services in Smart Home Environments [62].

This proposed system works with the user to reduce the intrusion that an automatized service management can introduce to the user’s experience. Moreover, the system is oriented to especially reduce the impact of bad predictions on multimedia services, trying to guarantee a good QoE from the user’s point of view. Finally, to overcome the possible difficulties that a deep learning classifying method may provide to the system, we introduce a reinforcement learning (RL) adapted method into the Smart Home.

This architecture is used to design Smart IoT-Networks, which contain interconnected objects with integrated AI (Smart Things). Some of these networks are Smart Home, Smart Office, Smart City, Smart Factory, among others. For this case study, the scenario is based on the connected objects in a Smart Home. The IoT-Gateway's AI classifies the objects connected in the network into workgroups and roles by layer [56], and then, when an object requires resources, the IoT-Gateway routes it, selecting the best node to provide them. The AI creates these groups to provide an automatic service to a user based on their features of functions, resources, and capacities. The following is a case of service within the Smart Home, e, g., when a visitor arrives at the house, the AI selects the objects with nearby features to provide this service. The system searches inside the house if there are users or not and automatically attends the door's visit. The necessary resources for this service are multimedia such as video and sound necessary to show the image of the visit on the objects that have this resource and that are close to the user inside the house. If there are no users inside the house, it will send them to objects or mobile devices in any location. One function that would be activated would be facial recognition and identity verification and then sent to be processed in the cloud. The first object to interact in the service would be the smart main door at the front of the house. This would be the requesting node for the resource and would execute the recognition and verification function. The IoT-Gateway's AI would be in charge of distributing and transmitting the video and voice to the objective objects that have this resource and that meet the condition of being close to a user and with the capacity to reproduce it. Figure 23 shows a case study for a workgroup that attends the door's visit service in a Smart Home through this architecture. This workgroup is organized according to the layers of the architecture in Figure 8. When the smart door attends a visit, it activates a request to send video and image processing data. The image processing data (facial recognition) is sent through the AI interfaces until the visitor's identification is obtained. The video is sent to each connected object with multimedia playback functions that is close to the user and routed through the IoT-Gateway. Therefore, the smart home announces a visit and the identification of a person located in the main door. If the user is away from home, the IoT-Gateway will send the information through the cloud to the closest objects with multimedia functions to the user (E.g., Smart Car, Smartphone, Tablet).

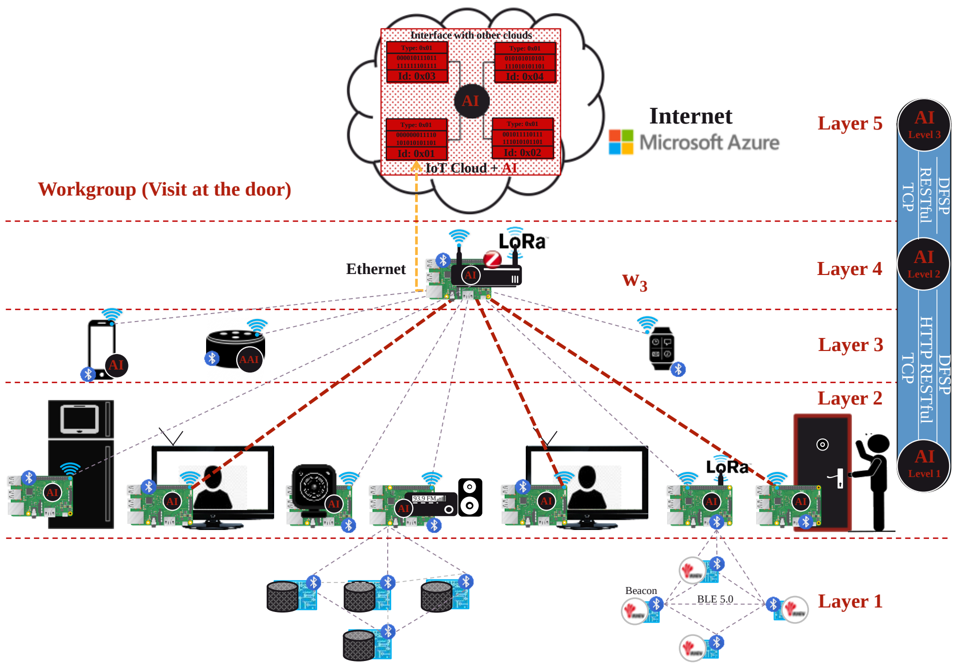


Figure 23. Case study with a single workgroup

In Smart Home, with this architecture, the IoT-Gateway, through the IoT nodes, can provide multimedia services. Through user requests given to the smart assistant placed in the IoT-Gateway, either by voice commands or either by a smartphone application, the user can play their favorite music in the audio system on the distributed speakers in the house, watch movies-on-demand, or automatic record surveillance videos of the house.

3.9.1. Intelligent System architecture

Once the network architecture has been explained, the system architecture must be discussed. The aim of the system is to improve the QoE of the services in the Smart Home, focusing on multimedia services. In order to achieve that, the intelligent system must enable or ask the user to enable the services the user may want to use after other services and learn how to disrupt the user's activity the least number of times.

The intelligent system is composed of several modules, located in the IoT-Gateway, that are interconnected to provide the desired functionality. The architecture of the system is depicted in Figure 24.

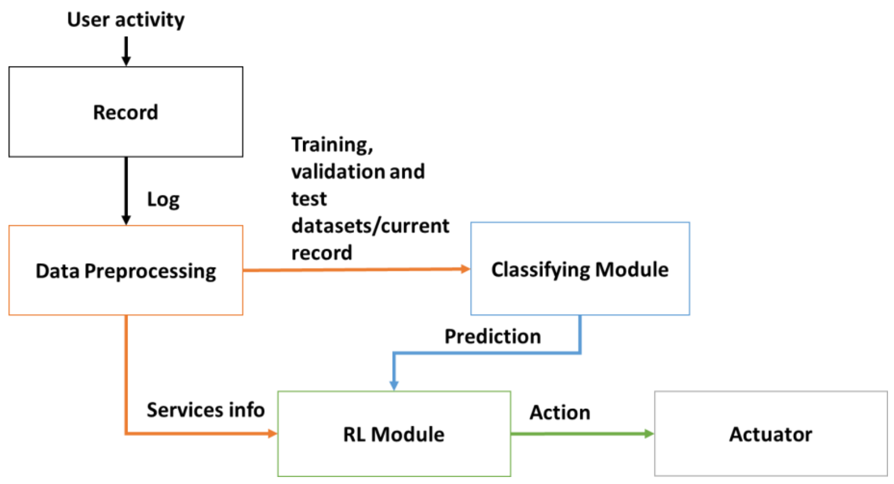


Figure 24. Intelligent system architecture

The first module in the system is the recording module. This module is not an intelligent system, but it is the necessary first step to get the data the system needs. Each time the user consumes a service, the IoT-Gateway records a data array with the structure shown in Table 10. Due to space constraints in the table, the user id column has been omitted and fields like the time, the options or the duration are shown as unique columns. However, in the dataset, they are split. For instance, the time and duration fields are split into day (only regarding the time column), hour, minute and second. As regards options, the entry is a set of columns, from option 1 to option 5, represented by binary values. The meaning of the options is meaningless for the system. Only the service finds this meaning useful. In Table 10, the first row means that the user enabled the heating service the second day of the week at 5:39 PM for one hour 42 minutes. Moreover, that day was a working day (the type of day field shows us this) and that service is not a multimedia service. The options here show us that the system was enabled in heat mode. The IoT-Gateway manages equivalency tables to transform the meaning of these fields into their values. Consequently, the smart system works with integers and that makes easier the processing of the data. Table 11 shows an example of an equivalency table. The options used in this record can be non-exclusive options.

Table 10. Record data example

Time	Type of Day	Service	Description	Options	Duration	On/ Off	Multimedia
2:17:39:17	1	2	Heat	10000	1:42:24	1	0
2:18:33:51	1	0	Lights	01001	0	0	0
3:9:11:25	0	3	Smart TV	10000	11:2:1	1	1

Table 11. Equivalency table example

Service	Option 1	Option 2	Option 3	Option 4	Option 5
0	Bedroom	Bathroom	Kitchen	Dining Room	Others
1	Main Door	Garage	Other Doors	Window 1	Window 2
2	Heat	Cool	Auto	Fan	Sleep Mode

The second module of the system's architecture is the data preprocessing module. This is a software module that computes the data to transform it into the input of the next modules.

After the data has been processed, the datasets extracted from the logs provided by the record module are sent to the classifying module. Here we have to distinguish between the training phase and the prediction phase. In the training phase, the dataset extracted is used to train the classifying. Therefore, a training, validator and test dataset are extracted. Once the classifying system has been trained, the classifying module is used to predict the next service to be consumed. Consequently, the data sent by the data preprocessing module are the next inputs for the classification. In that case, the classifying module returns the predicted service.

The next module in the system is the RL module. The RL module receives information about the services from the data preprocessing module. This information is used to build the initial states and to calculate the required metrics. For instance, the RL module needs to know if a specific service is a multimedia service. When the classifying module predicts a service consumption, that prediction is an input for the RL module. The RL module chooses the best action, and that action is the output of the RL module.

Finally, the IoT-Gateway has an actuator module that performs the action chosen by the RL algorithm.

In the next section, the data preprocessing and the classifying modules are described.

3.9.2. Preprocessing and Classifying Algorithms

In this section, the data preprocessing and the classifying modules are explained. First, the data preprocessing process is described. Its process and algorithms are detailed. Then, the classifying model chosen is described.

3.9.2.1 Data Preprocessing Process

Once the logs are provided to the preprocessing module, this module starts extracting some features from them. Firstly, the RL module uses some data features that are outputs of the preprocessing module. For the RL module, the statistical probability of changing from one state to another and the mean of the timestamp when it does are important data. Consequently, the preprocessing module transforms the data extracting these statistics. In order to achieve this, the module manages Markov chains. The definition of these chains adapted to the problem is described in Eq. 23:

$$\mathrm{P}[X_{n+1} = x_{n+1} | X_0 = x_0, X_1 = x_1, \dots, X_n = X_n] = \mathrm{P}[X_{n+1} = x_{n+1} \mid X_n = X_n] \quad \text{Eq. 23}$$

Where X_{n+1} is the next service consumed, X_n, the service consumed in the iteration number n and x_n is the service number n in the services set S.

In this system, the data will be processed regardless the time. That means, regarding the preprocessing, the time does not change the probability of the transition between services. That is depicted in Eq. 24:

$$\mathrm{P}[X_{n+1} = j | X_n = \mathrm{i} = \mathrm{P}[X_1 = \mathrm{j} \mid X_0 = \mathrm{i}] \; \forall i, j \in S \quad \text{Eq. 24}$$

The fact that the Markov chain does not consider time to set the probability does not mean that in the system the time is not considered an important input. However, the RL algorithm will use time in a different manner.

Once the records have been read, the preprocessing module turns them into matrices so that the RL algorithm can operate efficiently with them. The first data the RL will need is the probability of consuming a service. To set this probability, depending on the last service consumed, the preprocessing module builds a transition matrix. This matrix, given a consumed state i and a possible next consumed state j, defines the probability p of demanding j. The preprocessing module must satisfy the constraints defined in Eq. 25 and Eq. 26:

$$\mathrm{p(i,j)} \geq 0 \; \forall i, j \in S \quad \text{Eq. 25}$$

$$\textstyle\sum_{j \in S} p(i,j) = 1 \;\; \forall i \in S \quad \text{Eq. 26}$$

Where $\mathrm{p(i,j)}$ is the probability of consuming the service j after consuming the service i.

In this system, the probability of consuming a service is not enough. Another important statistical data is the time between transitions. If the time is not considered, the system could ask for the right service hours before the user wants to enable it. This, although will not be treated as a feature of the definitions in the RL algorithm, will be necessary information to implement some of the actions of the RL system. For these calculations, the preprocessing module uses the time and duration of the records for each service. As regards the means of time and duration, and to take advantage of the incremental processing of records, the calculation will use a recursive formula. In order to calculate the mean, the equation defined in Eq. 27 is used.

$$\mu_n = \frac{(n-1) * \mu_{n-1+} \, t_n}{n} \quad \text{Eq. 27}$$

Where μ_n is the media with n records, n is the number of records and $t_n$is the time or duration of the record number n. From this equation, we need to define the basic case as in Eq. 28:

$$\mu_1 = t_0 \qquad \text{Eq. 28}$$

The variance is calculated in a similar way. The recursive formula described in Eq. 28 is used to avoid iterating through each past record when a new service consumption is ended.

$$s_n{}^2 = s_{n-1}{}^2 + \frac{(t_n - \mu_{n-1})^2}{n} - \frac{s_{n-1}{}^2}{n-1} \qquad \text{Eq. 29}$$

With these definitions, we can set the algorithm of the data preprocessing module. This algorithm is shown in Figure 25, in a flow diagram that describes the algorithm. First, the data needed is initialized. Then, all the records are processed until there is no more records left. For each day of the week, that is why the next condition is compared with Eq. 29, the Markov and the stats are calculated. The pseudocode of the algorithm is described in Algorithm 7. This algorithm defines the main procedures of the data preprocessing module. Given a set of records from the IoT-Gateway, the module processes the records to assign them to users and days (User_Records). Then, the Markov transition matrix (Markov) is calculated. This is done in an iterative way. The algorithm of the Markov data building is shown in Algorithm 8 and explained later. Then, the other statistics needed are extracted from the time and duration data. This subroutine is explained in Algorithm 9. Finally, the datasets for training and validating the classification module are extracted.

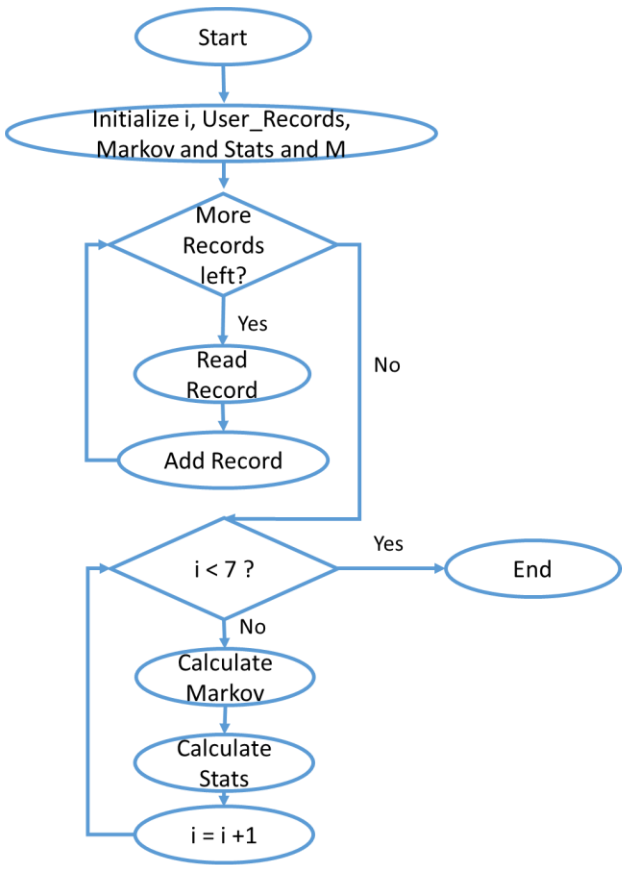


Figure 25. Data preprocessing algorithm

Algorithm 7: Data Preprocessing

```
1.  Given: Records //Ununstructed Records from the Gateway
2.  Var User_Records, var Markov, var Stats
3.  For each record in Records do
4.      If Get_User(record) not in User_Records
5.              Add_User_Record(User_Records, Get_User(record))
6.      End If
7.      Add_Record(User_Records, record)
8.  End For
9.  For each record in User_Recods
10.     For each day in range(1:7)
11.     daily_records = Get_Records_From_Day(day)
12.             For each d_record in daily_records
13.                     Calculate_Markov(Markov, day, d_record)
14.                     Calculate_Statistics(Stats, day, d_record)
```

```
15.             End For
16.        End For
17.  End For
```

As regards the Markov structure, it is calculated as it is shown in Figure 26, that depicts the flow diagram of this algorithm. In order to calculate the transition matrix, we need to know how many times a certain service is consumed after another one. We need to store the data in the Markov structure. In Figure 26, it is shown how this info is read from the record. The structure is indexed based on the day, the first service and the second service, which is consumed after the first one. Moreover, for each first service, we need the total amount of transitions, totalCases. If we find a record and the structure is not created, we need to create it. And then, the totalCases is set to 1, as shown in Figure 26. If that service is the first time that is consumed, the following service consumed has a 100% of transitions. Otherwise, we need to iterate through all the previous transitions from that service to calculate the probability for each second service. That is shown in the last loop in Figure 26. Algorithm 8 describes the same process with more detail.

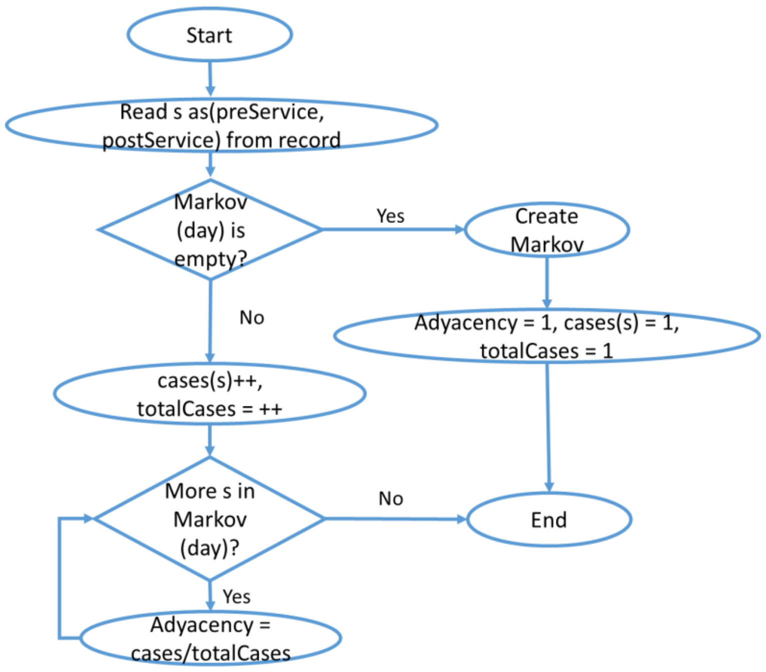

Start
Read s as(preService, postService) from record
Markov (day) is empty?
Yes
Create Markov
No
Adyacency = 1, cases(s) = 1, totalCases = 1
cases(s)++, totalCases = ++
More s in Markov (day)?
No
End
Yes
Adyacency = cases/totalCases

Figure 26. Markov transition matrix calculation algorithm

Algorithm 8: Markov transition matrix calculation

```
1.  Given: Markov, day, d_record
```

```
2.  Var preService = d:record.service
3.  Var postService = d.record.nextService
4.  If Markov[day, preService, postService] is empty
          a.  mark = Create_Markov(); //Create the structure
          b.  mark.adyacency(preService, postService) = 1
          c.  mark.service_cases(preService) = 1
          d.  mark.totalCases = 1
5.  Else
          a.  mark = Markov[day, preService]
          b.  mark.totalCases++
          c.  mark[postService].cases++
          d.  For each service in mark
                  i.  tempMark = mark[service]
                 ii.  tempMark.adyacency = tempMark.cases/mark.totalCases
          e.  End For
6.  End If
```

Finally, we find the statistics calculation in Figure 27. Figure 27 shows the flow diagram that corresponds to Algorithm 9. It depicts the two different ways of calculating the statistics. If there was no previous statistics for a specific day when the record is read, the data structure is created and initialized Otherwise, the statistics must be recalculated. By using equations Eq. 27, Eq. 28 and Eq. 29 these statistics can be recalculated each time a new service is consumed after another one in the same day or each time the new service ends its consumption (for calculating the duration).

In the following subsection, the classifying module is explained. That module uses some of the data provided by this module. In the training phase, the records are split to create the datasets. In the prediction phase, each new record is sent to the classifying module.

3.9.2.2 Classifying Module

The classifying module is based on neural networks. Then, we have a deep learning method to predict the next service consumed by the user. The entries of the system will be the different measurements of the record (time, day, type of day, service consumed, duration and so on). In the records, we have 20 different features. As output, the different services provided. We define 7 different services, shown in Table 12. A weak point of having a classifying system as the single method to predict service consumptions is that if a new service is added to the system, the classifying model has to be redefined and trained again. However, with an RL as a supervisor, that process can be delayed, and a provisional action can be added to avoid a QoE reduction until the classifying model is trained again.

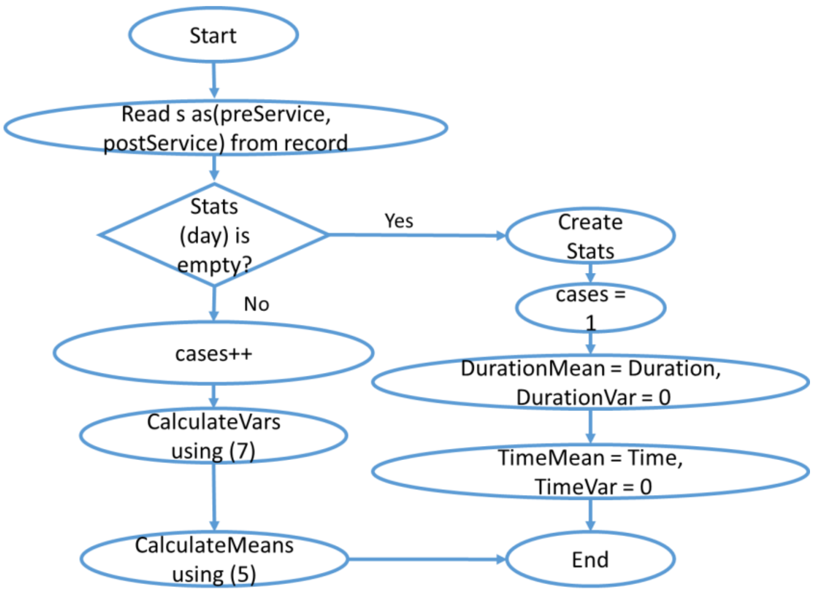

Start
Read s as(preService, postService) from record
Stats (day) is empty?
Yes
No
Create Stats
cases = 1
DurationMean = Duration, DurationVar = 0
TimeMean = Time, TimeVar = 0
cases++
CalculateVars using (7)
CalculateMeans using (5)
End

Figure 27. Statistics calculation algorithm

Algorithm 9: Statistics calculation

```
1.  Given: Stats, day, d_record
2.  Var preService = d:record.service
3.  Var postService = d.record.nextService
4.  If Stats[day, preService, postService] is empty
5.      stat = Create_Stats(); //Create the structure
6.      stat.meanDelta = postService.time-preService.time // (6)
7.      stat.varDelta = 0
8.      stat.meanDuration = preService.duration // (6)
9.      stat.varDuration = 0
10.     stat.totalCases = 1
11. Else
12.     stat = Stats[day, preService, postService]
13.     stat.totalCases++
14.     stat.varDelta += pow(postService.time - stat.meanDelta, 2)/stat.totalCases -
        stat.varDelta/(stat.totalCases-1) // (7)
15.     stat.meanDelta = (stat.totalCases-1)*stat.meanDelta + postService.time/ stat.total
        Cases // (5)
16.     stat.varDuration += pow(postService.duration - stat.meanDuration, 2)/stat.total
        Cases -  stat.varDuration/(stat.totalCases-1) // (7)
17.     stat.meanDuration = (stat.totalCases-1)*stat.meanDuration + postService.duration/
        stat.totalCases // (5)
18. End If
```

Table 12. Services description

Service 0	Service 1	Service 2	Service 3	Service 4	Service 5	Service 6
Lights	Doors	Heat	Smart Leisure	Music Player	Door Surveillance	Video Streaming

The classifying model will be based on a neural network whose architecture is depicted in Figure 28. Classifying neural network architecture. We define a number of neurons in the input layer equals to the number of features in the data. In our case, we will have 20 features, so 20 different neurons in the input layer. The number of hidden layers, n_h, and the number of neurons in each hidden layer, m_h, will be parameters of the model. After testing the model varying these parameters, the model used will be the one with the highest accuracy.

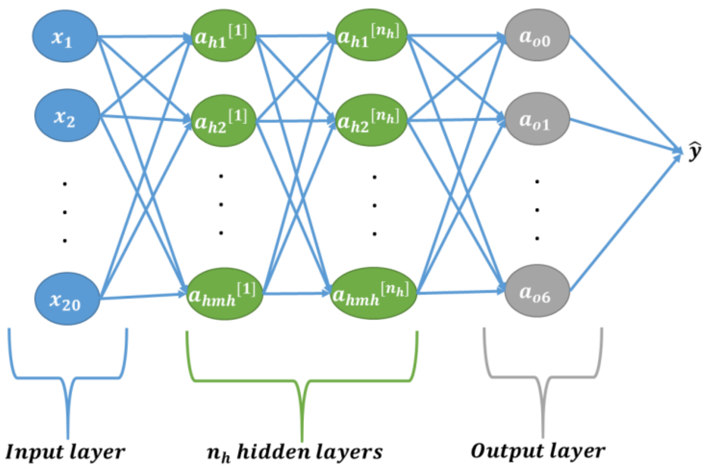


Figure 28. Classifying neural network architecture

As regards the output layer, it will be composed of 7 neurons, one for each class to be detected by the system. If the number of different services in the Smart Home increases, the system would need to extract more relevant features and the model would need to be adapted. However, this adaptation is not going to be considered in this paper, and it will be considered as future work.

The loss function will be cross-entropy and the optimization method will be Root Mean Square Propagation (RMSprop), which achieves good results in multi-layer neural networks [63].

Another aspect of the model that has to be chosen is the activation function. We will use the ReLU function as the activation function of the hidden layer, a_h [64]. The softmax function will be used as the activation function of the output layer, a_o, to get the probabilities of belonging to each class [65].

With the model presented, the next service consumed by the user can be predicted. In the next subsection, we define the RL module, that will choose the best action to perform based on this prediction.

3.9.3. Reinforcement Learning Module

This section describes the RL algorithm that will be used to reduce the impact of bad predictions. First, the environment, the states and the actions are defined. Then, the data structures that the algorithm needs to work on are described. Finally, the rewards calculation and the policy of the system are detailed.

3.9.3.1 Environment, States and Actions

In this subsection, the environment, the states and the actions that the algorithm will use to implement the reinforcement learning will be described.

Firstly, we have to define the environment. The reinforcement learning will act in the Smart Home environment. Exactly, it will notify the IoT-Gateway which command must perform. Therefore, the agent of the reinforcement learning algorithm will be the IoT-Gateway. Initially, it could be thought that the user would be the agent, but the user will be only the source of information of the actions performed by the agent. For instance, when the user consumes a service and the algorithm decides to start another service after a certain amount of time, the user may not need that service and give the order to turn it off. The user, then, is an input for the algorithm. In this case, the user is saying that the action performed was not chosen correctly. This fact would affect the reward of the action. We will discuss that in the next subsections. For now, it is important to notice that the user will be an input, not the agent that modifies the environment. Despite this fact, the user is the one who starts enabling the services. Consequently, if the definition of the states would be only based on the current service running in the Smart Home, the definition of the environment and the agent would be more complicated. Furthermore, the algorithm would be incomplete, due to the fact that we need to know which service is supposed to be necessary to activate. It is there when we need the classifying module. Moreover, due to the user patterns, the day and the current time are also relevant data.

From the previous discussion, the following definition of the states is obtained. The RL algorithm will decide which action needs to be performed based on the current service consumed, the next service predicted and the current day. This presents a problem that will not be addressed in this paper: how new states are generated through the use of the system. We will consider in this paper that the states are statics and are generated from the transition matrix obtained from the preprocessing module. However, new patterns can be adopted by the user and this will be discussed in future works.

Usually, the states of an RL algorithm can be depicted in a state diagram. However, in this scenario, the state diagram can be composed of a high number of states, depending on the number of services. This can make the algorithm not scalable. Nevertheless, the Smart Home environments do not have a high number of services. Furthermore, not all the services can be important enough to define a state. In this paper, we will use all of them but, in future works, a categorization of services can be proposed to reduce the space of states.

Figure 29 shows an example of a diagram of states with three different services to enable. In this case, there is no probability of transitioning from state 2 (S2) to state 3 (S3). Therefore, there is no transition between those states in the graph. Furthermore, depending on whether the user actually consumes the predicted service, the next state might be with the same service being consumed. That may happen when the classifying module does not predict the next service accurately or when the prediction was to consume again the same service. The state diagram is not, then, a direct representation of the Markov chain derived from the transition matrix. There are transitions to states with the same current service that do not represent the same service being consumed twice. In Figure 29, in order to make easier the readability of the diagram, the states are composed only of the current service. However, the diagram is more complex because the algorithm defines the state as a pair of states. The first state is the current state and the second one is the predicted state. Figure 30 depicts the diagram of states of the state S2 of Figure 29. Within each state of Figure 29, there would be a subdiagram with similar transitions. Although this can add some difficulty to understand the algorithm, the number of states is finite and not big enough to present a deep-learning algorithm for the policy function as in [66]. However, it could be a future work to study.

For each state defined, a set of actions can be performed. The actions, however, must be classified based on how much they interrupt the user's activity. This helps the algorithm to not reduce the user's QoE as much if the predictor fails to predict the next service. For this proposal, six different actions are going to be defined and classified depending on this intrusion level. The actions are shown in Table 13, where the level of intrusion and the description are detailed. The simplest action is to wait, without executing any service. This has the lowest impact on the user because they do not have to do anything. However, this does not mean it is always the best action, because sometimes enabling a service or turning on a node can reduce the waiting time for the user or implying other advantages (for instance, saving energy or heating up the house before the user arrives). An alternative with low impact would be to ask if the user wants to enable the predicted service or another one from a list of similar alternatives (services from the same group). This is not too different from a manual service selection. If the system only asks to enable a certain service or it turns on a node it will have a higher intrusion from the user's point of view. We have to take into account that certain services are more intrusive than others. This is reflected in the actions. The system can automatically start a predicted service, with some options or subfunctionalities that are not too intrusive, such as perform a search on Internet or increase or decrease the temperature a few degrees. However, other services, like opening a door, turning on the TV or start playing a song or video have a bigger impact on the user. Finally, stopping services that the user is using or turning off nodes automatically has the biggest impact on the user, so transitions that require disable services will have the biggest level.

Table 13. Actions

Level	Action
0	Wait
1	Ask with alternatives
2	Enable a node or ask to start a service
3	Start a low-impact service
4	Start a high-impact service
5	Stop services or turn off a node

The definitions of the actions are not simple actions as they could be found in other RL algorithms, but there are actions that will depend on the predicted service. That means, that enabling a node will enable a node that provides the predicted service, starting a service will start the service predicted or a module of the service and so on.

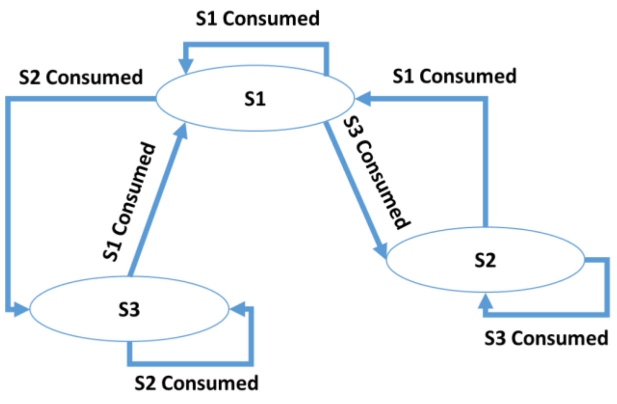


Figure 29. State diagram with three services

These actions provoke changes in the network performance, and, depending on the next values obtained from the Statistic Analyzer, the reward value of the action taken will be updated.

The way the rewards are assigned and calculated and how the actions are performed are described in the next subsections. However, we first need to know the data structures and concepts.

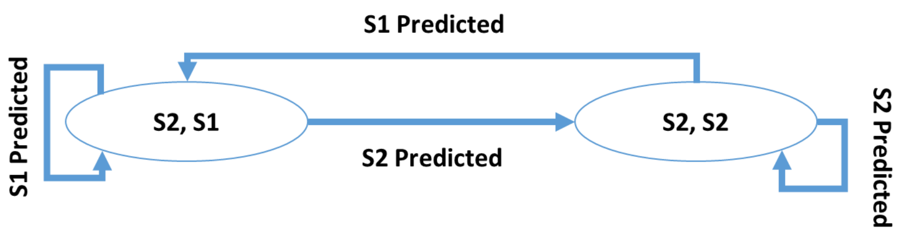


Figure 30. Subdiagram of state S2

3.9.3.2 Data Structure

In this subsection, the data structures that the RL algorithm uses are described.

First of all, the RL algorithm needs a structure where the reward of each action for each state is stored. This data structure will be a matrix, where the rows will be the states and the columns will be the different actions of the algorithm. In this case, the states are a pair of current service and predicted service values. Table 14 shows an example of the data structure for three different services, following the same state diagram that is shown in Figure 29.

Table 14. Reward-State Matrix

State	Action 1	Action 2	Action 3
$S1, S1$	1.3	7.5	2.21
S1, S2	0.4	1.2	2.05
$S1, S3$	1	2	0.75
$S2, S1$	1	2	0.75
$S2, S2$	1	2	0.75
$S3, S1$	1	2	0.75
$S3, S3$	1	2	0.75

Secondly, the algorithm will need information about the services. This information is given by the data preprocessing module. For each service, the following data is required: the group where the service belongs, the impact that the service has on the user and if the service is multimedia. The group of the service is data that the IoT-Gateway knows because it is that agent who categorizes the services attending to the architecture presented in Figure 8. If the service is multimedia it also comes from the IoT-Gateway. Like the group of the service (workgroup), this is information that is presented in the dataset. However, the last data that is needed, the impact of the service on the user, must be set by the RL algorithm. A simpler classifying module could be added in the system architecture just before the RL module to determine which services have a bigger impact on the user. However, to make the proposal simpler, the IoT-Gateway provides the RL module with this information, obtained statically from the group of the service. Table 15 shows an example of this data structure, which we will name as infoServices. The impact field is a positive integer. The bigger the impact value, the more intrusive the service is.

Table 15. InfoServices Matrix Example

Service	Workgroup	Multimedia	Impact
1	1	0	5
3	2	1	4
4	3	1	4

The last data structure needed to implement this RL algorithm is the Input User Matrix. This data structure will store how much a user input is needed for each action. That means, that the algorithm will be able to know the impact of having a certain input for the user for each action chosen. This will be useful for knowing how much a certain action was appropriate for a state, that is, how the RL algorithm should be rewarded. For instance, if a set of services is given to the user to choose which one should be started, if the user selects the predicted, the reward will be different from the one obtained if the user discards all the possibilities. This will be discussed in more detail in the next subsection, where the policy of the RL algorithm is detailed. Table 16 shows an example of this matrix.

Table 16. Input User Matrix Example

Action	User Option 1	User Option 2	User Option 3
a_1	0.2	0.25	0.1
a_2	0.4	0.8	0.1
a_3	1	0	0

3.9.4. Rewards, Policy and Objective Function

In order to define the reward properly, we have to analyze the problem we are dealing with, because it is the problem, and the scenario, the one that defines the appropriate method to obtain and calculate the reward. The reward will indicate to the system whether the action chosen was effective in the same state or, otherwise, if there are better options. If reinforcement learning is applied to a game, the effects that a performed action over the player determines the reward. If the game provides an actual reward such as finishing a level or defeating an enemy, the reward will be positive. If other actions usually drive to losing a life, losing objects and so on, the reward will be lower and will decrease if we choose that action, even being able to contain a negative value.

In the environment previously defined, the goal of the system is to avoid the user from enabling services or nodes. However, it is also important to reduce the intrusion of the system, especially regarding multimedia services. Therefore, the input of the user will be quite important to know if the action chosen by the system was appropriate.

Applying the RL system to the result of a classifying algorithm modifies the way the reward is calculated. In this case, the reward will be a measure of the number of times the classifying module has predicted correctly which service would be consumed. Therefore, if the module provides high accuracy with certain services, the actions with a high level of intrusion can work well. On the other hand, when it has low accuracy, performing an action with a low level of intrusion could be a better option.

The algorithm has a set of actions, J. We are going to describe the general case where several actions can belong to the same level classification. However, this definition will also be valid when there can be only one action per level. Consequently, we can define the reward obtained after choosing an action j as in Eq. 30.

$$\mathrm{r}_j = \mathrm{u} \quad \text{, where} \qquad \text{Eq. 30}$$

$$\mathrm{u} = 1 \leftrightarrow \mathrm{Input(User)} < \beta \qquad \text{Eq. 31}$$

$$\mathrm{u} = -1 \leftrightarrow \mathrm{Input(User)} \geq \beta \qquad \text{Eq. 32}$$

where Input(User) is a parameter that returns the IoT-Gateway and its value depends on the action chosen and the action the user does after. The possible values were defined previously, as an example, in Table 16. Moreover, β is a threshold defined to classify the action performed by the user as corrective or not. Then, if the action was corrective, the reward should be decreased.

Nonetheless, if we only consider the reward like that, we can drive the system to a point where the actions of less level, due to the fact that they are less intrusive, present a higher reward. In order to solve this, the reward should be incremented regarding the level of the action chosen. Consequently, the reward given to an action j in the iteration i will be defined by Eq. 33.

$$\mathrm{r}_{i,j} = \mathrm{u}_{i,j} * \frac{level(j)}{Max(level)} \qquad \text{Eq. 33}$$

Where level(j) is the level of the action j chosen.

The last adjustment that needs to be done is to give extra importance to multimedia services. That means, for those actions coming from a state whose current service is multimedia, the reward should increase or decrease at a higher rate. In Eq. 34 we can see this adjustment.

$$\mathrm{r}_{i,j} = (1 + \mathrm{multimedia}(s)) * \mathrm{u}_i * \frac{level}{Max(level)} \qquad \text{Eq. 34}$$

Where $mulimedia(s)$ is the flag in the dataset that identifies the service s as a multimedia service.

We can define, then, the total reward of an action j as Eq. 35.

$$\mathrm{R}_j = \sum_{i=0}^{n} \mathrm{r}_i \qquad \text{where} \quad \mathrm{R}_j \geq 0 \; \forall \, j \; \in J \qquad \text{Eq. 35}$$

Initially, the rewards are calculated depending on the level and the probability of transitioning from the initial state S_a to S_b as is defined in (14):

$$\mathrm{r}_{0,j} = (1 + \mathrm{multimedia}) * (\mathrm{P}(\mathrm{S}_a, \mathrm{S}_b) * \frac{level(j)}{Max(level)} + (1 - \mathrm{P}(\mathrm{S}_a, \mathrm{S}_b)) * \frac{Max(level) - level(j)}{Max(level)}) \qquad \text{Eq. 36}$$

We need to define then the policy of the system. In subsection 5.2 we defined a matrix that contains the rewards for each pair of state and action. Usually, RL algorithms choose the action with the highest reward. However, in this case, the policy will have a component of exploration. This component will force the algorithm to try actions that had not been tried for a high number of iterations. Eq. 37 defines the function ρ that represents the probability of choosing and action α that has less reward than the action with the maximum reward, n, for the same state.

$$\mathrm{\rho}_\alpha = \frac{R_n - \mathrm{R}_\alpha}{|level(n) - level(\alpha)|} + \omega * i \qquad \text{Eq. 37}$$

Where i is the number of iterations that have passed since the last time action α was chosen and ω is a parameter that sets a weight to the exploration.

The policy function can be described then by Eq. 38:

$$\pi(s) = a_j | \, \rho_j \geq \rho_n \; \forall j, n \in J \qquad \text{Eq. 38}$$

3.10. Conclusion

A new algorithm has proposed for allocating and sharing the common resources among the heterogeneous devices in the scenarios of IoT.

Although, the MQTT protocol is redesigned to consume a low bandwidth of QoS. It was demonstrated through simulation that with a simple protocol like DFSP, the length of the payload of the PUBLISH message was sufficient to transport the information about the functions and services of the Things.

Notwithstanding each training matrix of each thing requires many iterations of the user, the simple matrix used that proved that the algorithm AI works ideally in the system.

While there is still no simulator that completely emulates an SBC like the RPi3, the Packet Tracer simulator can be a good option to see the behavior of the algorithms designed within the network.

After successfully modifying the MQTT protocol of the python code example that is in the Packet Tracer simulator, it is likely that the code for the CoAP and HTTP-Rest protocols can also be created and simulated in the network.

In a conventional WLAN, a WiFi router can not intelligently manage Internet access, storing datasets, host an IoT-Broker, or support AI. For this reason, it is proposed the use of low-cost SBC and freely developed boards and convert them into programmable IoT-Gateways to let them use the algorithms proposed in this paper. In this way, the objects connected can send information to AI through IoT-Protocols using this architecture.

The IoT environment provides a set of several services whose constraints and requirements vary from each other. In addition, the different features of the IoT nodes turns the problem of selecting which nodes must implement which service into a complex question. M2M communications and publisher-subscriber models can be used to route the information and create bonds between consumers and publishers. However, the IoT environments and their services may present different scenarios. Therefore, the routing protocol that interconnects the nodes should manage different criteria and select the most appropriate. In this paper, a new routing protocol for M2M communications in IoT environments has been proposed. The problem has been formulated, the architecture has been discussed, and both the model and the algorithm have been detailed. Moreover, some criteria that may be relevant in IoT environments have been defined. The proposed protocol uses an AI algorithm that selects the most relevant criteria to establish a cost function. The AI algorithm that has been chosen is the NSGA2 algorithm. Then, the protocol has been simulated to test its functionality.

IoT has provided new ways of networking, communicating and sensor development. In addition, with its introduction, several new applications have been designed in several fields. Smart Home is a new way of understanding services at home. With the communication that this technology offers, users can access new services at home. In addition, AI can help automatizing tasks at home using these services and architectures.

In this paper, we have introduced an intelligent system to automatize Smart Home services management. The aim of the intelligent system was to avoid user interruptions to guarantee a good QoE, prioritizing the multimedia services. In the system, we have designed the data preprocessing process, the classifying algorithm and the RL system that improves the performance, defining all the related concepts needed to describe the scenario and so that the system can interact and provide functionality.

Chapter 4. Test and evaluation of the proposed architecture

4.1. Introduction

Next, the DFSP protocol is tested in an IoT-Gateway by measuring the data packets with an IoT-Protocol as MQTT. Each object connected to the experiments' IoT-Networks is organised accord proposes architecture. With the dataset extracted from each object using DFSP, it is analyzing the intelligent clustering algorithm (based on machine learning classifier), the intelligent routing (based on the NSGAII optimiser) and the quality of experience (QoE) based on deep learning and reinforcement learning.

4.2. Test on IoT-Gateway

This proposal's tests are carried out to demonstrate that the algorithm can encapsulate the information extracted by the DFSP protocol and send it over MQTT to the IoT-Gateway.

4.2.1. Implementation of the devices

The development of this proposal involves the use of single-board computers (SBC) such as the Raspberry Pi 3 Model B+ (RPi3) [67]. Each actor in this proposal, it has implemented with an RPi3, except for the cloud platform. It has tested on two different platforms for IoT: ThingsBoard [68] and ThingSpeak [69].

The idea is that each thing is controlled by an RPi3 including the IoT-Gateway, under the Android Things operating system [70]. Using the Java programming language and the Android Studio development environment [71].

4.2.2. Testbed

The technical implementation of this proposal is possible through the implementation of the RPi3, three of it emulating three Smart Things and one emulating the IoT-Gateway. Another way to verify the operation of this proposal can be done through a simulation. Several simulators were studied, including Cooja [72], [73], NS-2 [74] (Network Simulator 2), GNS3 [75] and Cisco Packet Tracer [76]. Most of the simulators are oriented to IoT sensor networks (Motes, mostly 8 and 16 bit Microcontroller based). Some of these simulators do not support the use of the RPi3 because it falls into the SBC category and cannot run a full Operating System (OS). Sometimes also, the OS of the simulators are not compatible with OS of RPi3. Currently, there is an RPi3 simulator [77], [78] that can emulate the operating system (OS-RPi3), but cannot simulate it in a network.

Cisco Packet Tracer simulator was the best option in this case, although it has limitations when emulating the speed of processing of the packets and the time of decision in the AI algorithm. This simulator allows illustrating the behavior of the RPi3 in a network environment and seeing the operation of the MQTT protocol. It also allows you to modify the pre-installed sample programs and adjust them to the design of this proposal.

The modified program was the MQTT Client and MQTT Broker protocol in Python code into each RPi3. An algorithm was added to both to calculate the length of each of the 16 messages of the MQTTv3.1.1 protocol [79], called Remaining Length (RL). Using the model described in the previous session, the algorithm adjusts to the size of the IoT-Protocol payload, in this case MQTT.

The MQTT Client and MQTT Broker initially worked manually, but now due to this modification, it is the control algorithm in gateway whose controls them automatically. In the payload of the PUBLISH message the algorithm sends the protocol DFSP announcing and sharing information about the services and functions of the Things. So a simple network design was proposed, starting with a group of Things where a specific task is assumed. The network of this test supposes an IoT-Gateway based on a centralized management architecture following a star topology.

Figure 31 shows the simulation performed with the Cisco Packet Tracer simulator version 7.2.1.

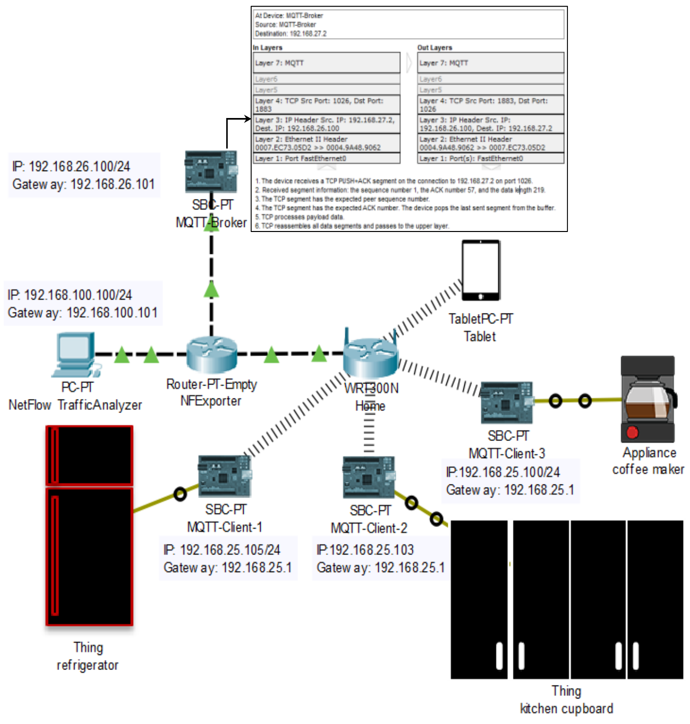
The IoT-Gateway of the simulation is represented as the next set of elements, a wireless router WiFi, a Cisco 2911 router (NFExplorer) and SBC-RPi3 (inside is content the MQTT broker and AI control algorithm). A real implementation would be done only with an SBC-RPi3, who would be integrated by MQTT Broker and the AI control algorithm.

Smart things are also integrated by an SBC-RPi3 and host the MQTT client and the AI algorithm, which is responsible for announcing its functions and services. To simulate the operation of each Smart Things, a component called "Thing" was programmed and connected to an SBC-RPi3. Then in this way, it is possible to simulate the set (Thing + SBC-RPi3) as a refrigerator, a cupboard and a coffee machine.

All these devices are connected with WiFi technology following a star topology that supports the transport of IoT-Protocols [80]. In this case, the MQTT protocol selected and modified for all the reasons explained above.

With the target of evaluating the system, it was used to capture the packets in the network, a special configuration with the Cisco 2911 router and a personal computer (PC) with NetFlow Traffic Analyzer [81].

The DFSP protocol is transported over MQTT, which in turn is transported throughout the network over TCP sessions, established at the beginning of the connection. For this reason, all TCP packets are analyzed, as their traffic and bandwidth changes depending on the other protocols.



At Device: MQTT-Broker
Source: MQTT-Broker
Destination: 192.168.27.2

In Layers	Out Layers
Layer 7: MQTT	Layer 7: MQTT
Layer6	Layer6
Layer5	Layer5
Layer 4: TCP Src Port: 1026, Dst Port: 1883	Layer 4: TCP Src Port: 1883, Dst Port: 1026
Layer 3: IP Header Src. IP: 192.168.27.2, Dest. IP: 192.168.26.100	Layer 3: IP Header Src. IP: 192.168.26.100, Dest. IP: 192.168.27.2
Layer 2: Ethernet II Header 0007.EC73.05D2 >> 0004.9A48.9062	Layer 2: Ethernet II Header 0004.9A48.9062 >> 0007.EC73.05D2
Layer 1: Port FastEthernet0	Layer 1: Port(s): FastEthernet0

1. The device receives a TCP PUSH+ACK segment on the connection to 192.168.27.2 on port 1026.
2. Received segment information: the sequence number 1, the ACK number 57, and the data length 219.
3. The TCP segment has the expected peer sequence number.
4. The TCP segment has the expected ACK number. The device pops the last sent segment from the buffer.
5. TCP processes payload data.
6. TCP reassembles all data segments and passes to the upper layer.

Figure 31. Simulation of the IoT-Network

The capture of the data was achieved with NetFlow Traffic Analyzer through the PC and then was crossing it with the information obtained by the router NFExplorer and the information provided by Eq. 1 and algorithm RL.

Once all the elements of the network have been configured, is measures the system with two types of tests. Both tests have the same initialization process autonomously through the AI. This test consists in seeing the behavior of the network with or without DFSP protocol [82].

The captured data are analyzed and graphed to interpret it through four graphics.

The three first is composed of three operation mechanisms, connection with the broker (CONNECT message), PUBLICATION/SUBSCRIPTION message and DISCONNECT message. The fourth describes these mechanisms separately in five operation segments.

Figure 32 and Figure 33, the behaviour in the length of the packets are observed when the network is started without carrying the DFSP protocol, and when it is started carrying the DFSP protocol, in both cases, it is realized the same process.

In both figures, the lines that describe the length of the packets corresponding to the connection and disconnection mechanism remain the same.

However, in the mechanism corresponding to subscription and publication the length changes in relation to the data of the DFSP protocol.

The data sent in the DFSP tests were small, only functions corresponding to "on and off", and short services such as "Make Coffee". Therefore, package sizes are not of the order of Mega Bytes, but only of some Bytes.

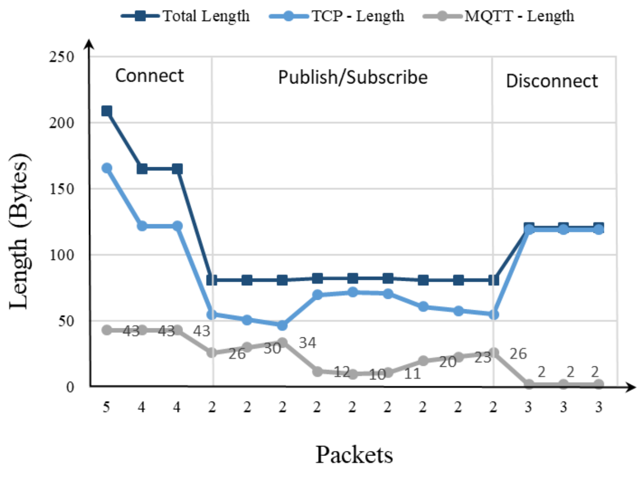


Figure 32. Length of the MQTT/TCP packets

As can be seen in Figure 33. Length of the DFSP/MQTT/TCP packets, the length of the packets shows that the DFSP protocol has been introduced in the payload of MQTT compared to what was seen in Figure 32. Length of the MQTT/TCP packets.

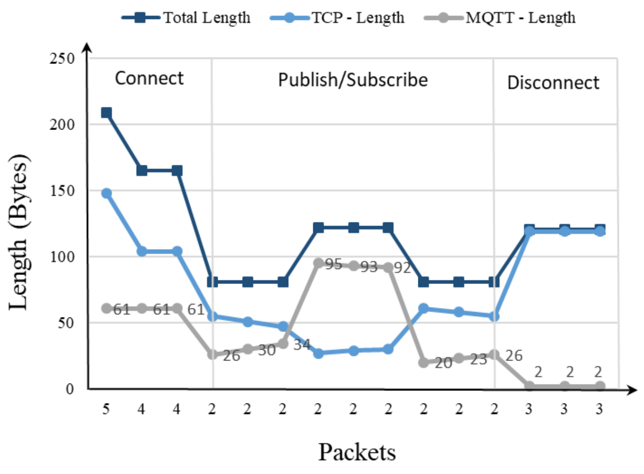


Figure 33. Length of the DFSP/MQTT/TCP packets

The length of the TCP packets given during the test is shown in Figure 33. Length of the DFSP/MQTT/TCP packets. The maximum values are peaks of 209 bytes, while the minimum value has been 81 bytes, which belongs to an UNSUBSCRIBE message. The average number of bytes transferred has been 116 Bytes.

Figure 34 shows the relation between numbers of packets is transmitted per seconds along the tests. The captured time is the real time of duration each packet sent in the network, which is measured in milliseconds (ms). The maximum time transmission of packets was at 0.1 ms, with 5 packets. The rest of the packets remain constant after the connection is established through the TCP socket.

In this time interval, PUBLISH/SUBSCRIBE messages are exchanged. The average time of the TCP packets transporting them is 0.070 ms.

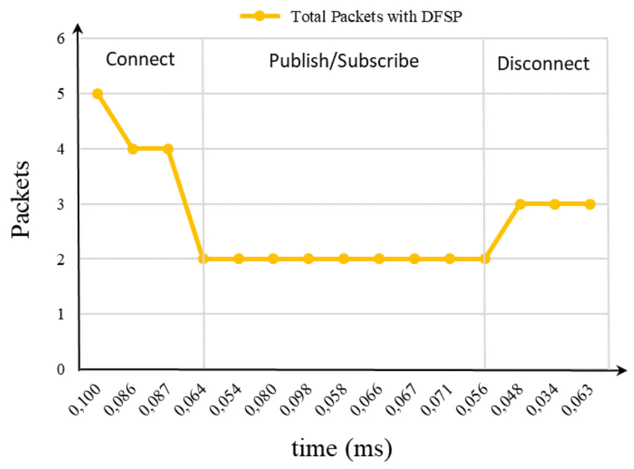


Figure 34. Number of packets transmitted

In Figure 35, it is can see the data transfer rate (DTR) in Bytes per seconds between each Smart Thing and the IoT-Gateway. In the simulation, the AI makes a boot of the smart things automatically and start to send the messages of MQTT protocol to the Gateway.

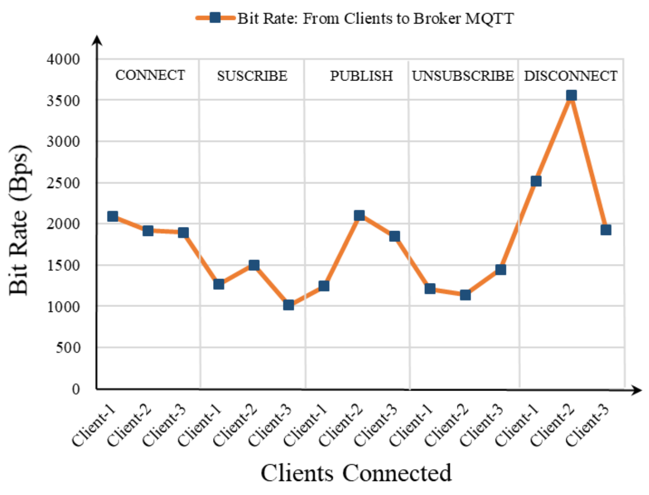


Figure 35. Data transmission rate

In Figure 35, the transmission of a message different from the MQTT protocol is observed in a sequential test process performed by the AI. This sequence begins with the CONNECT message and ends with the DISCONNECT message. The bit rate of each transmission from the customers is on average between 1000 Bps and 2100 Bps. In the disconnection segment, a higher speed is observed, on average, between 2100 Bps and 3600 Bps.

In the simulation seen in Figure 36 has added to each Thing (fridge, cupboard and coffee maker) a training matrix based on statistics of usage and user preferences.

These data are assumed through a text file, in order to observe the operation of the system and thus be able to capture the packages evidencing the transport of information autonomously to other machines.

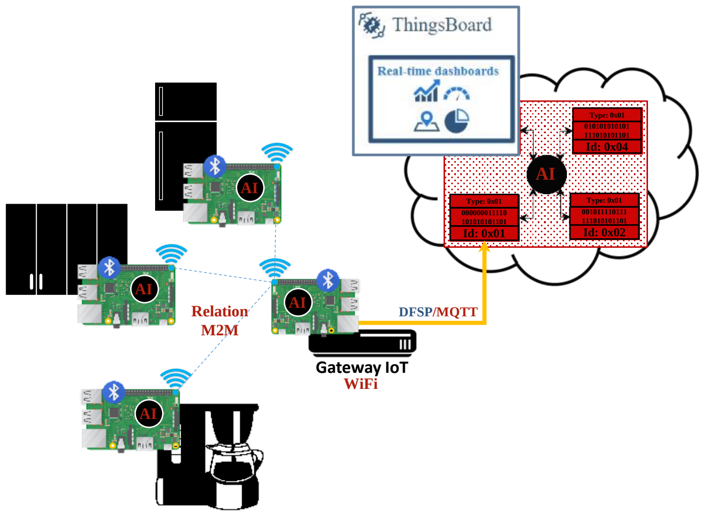


Figure 36. Implementation of the proposal

The simulation also allows us to observe the functioning of Things by controlling other things with basic functions such as on/off (on, red color), selection of type coffee preparation and the ingredients available for its preparation.

In a real implementation within a smart home, everything will be smart [83], [84]. Each of the things in this proposal, including the gateway, is built with an RPi3 and programmed with AI. As shown in Figure 34, it also connects to the cloud where there is also a platform with AI. As in the simulation, this proposal is expected to use other types of connections such as Bluetooth 5.0, WiFi and Ethernet. The idea is that protocols M2M, in this case, MQTT can be transported throughout the network, regardless of the type of the underlying connection.

4.1. Test on clustering

This section provides the performance test. The tests aim to observe how the objects behave under this architecture using the algorithms and the proposed model.

The idea is that the conventional router in a home will be replaced by an IoT-Gateway, with the ability to support AI, routing, grouping, storage, management, and hosting of services. These include hosting a Broker for M2M connections and allowing reprogramming to accommodate a ML classifier. However, the scope of the tests is only limited to grouping based on classification-oriented algorithms.

4.1.1. Implementation of the devices

The development of this proposal involves the use of single-board computers (SBC) such as the Raspberry Pi 3 Model B+ (RPi3) [67]. Each object in this proposal between layers 2 and 4 have been implemented with an RPi3 device, except for the Cloud and Sensor layers. In layer 1, the sensor devices are implemented on a Programmable System-on-Chip (PSoC) as the ESP8266-01, which is integrated with a microcontroller and a Wi-Fi network module or is also used a WIFI LoRa 868 (V2) board [85], to implement a sensor.

4.1.2. Considerations for the simulations

Since it is not possible to modify a wireless router in a house to include network management capabilities with this architecture, it is used an RPi3 to replace it. In this way, it is possible to program all the algorithms, modify them, and adjust them as often as necessary to achieve good tests. Therefore the RPi3 will be the IoT-Gateway, implemented as a Wi-Fi router with expanded capabilities, and adjusted to this architecture's operation. Among these capacities are the different IoT-Technologies of interconnection it can handle and the different IoT-Protocols with which it can communicate. However, we will only use Wi-Fi over TCP protocol in the simulation connection in order to facilitate the tests. The RPi3 devices of layer 3 are things and they connect with the Layer 1 sensors via Wi-Fi and Bluetooth.

The first part of these tests show how the dataset os established in the IoT-Gateway. It is preconfigured with several features, organized through feature profiles, and select the features that will act as classifying classes for each profile. Once the dataset is structured, the data extracted from all the objects connected in the network is stored following the number and order that the AI requires through the interface.

In the second part, the capacity λ of the devices on which objects are developed (Layer 2 to Layer 4) are considered the same because all the objects are based on an RPi3, and therefore, the capacities are the same. In the case of the connections from Layer 2 to the boards of the devices in Layer 1, the capacities are different and lower, so in this case, it is considered that λ equals the lowest.

The third part tests the grouping algorithm, based on the fact that the IoT-Gateway establishes routing over TCP. Then the case is considered that the network is previously established and that the IoT-Gateway is the one who discovers objects or waits for them to be announced.

This simulation scenario uses a fixed number of objects and groups. It is designed assuming each object's participation within a group when attending a service type. In this way, each object reports the data of its features under these conditions. This data is stored in the IoT-Gateway's dataset. Then, it is analyzed using different classification techniques, testing each one separately until the classifier is found with the highest precision in the prediction.

The data obtained from the objects in the simulation are transported under the architecture's AI Interface policies. Whereby, our system uses the DFSP protocol over IoT-Protocols such as MQTT (layer 2 to 4) and MQTT-SN (layer 1) [86] over TCP for transport. The data is pre-processed in level 1 according to the centralized AI requirements and finally processed in level 2; for this case, only level 3 is used when it is necessary to process very dense information such as sound and images.

Once the entire network converges and is stable, with a previously pre-established dataset and the selected classifiers and trained, the network is finally ready to receive a new object. Therefore, if a "New object" enters the network, the ANNOUNCEMENT message is activated and sent through DFSP/MQTT protocol to the IoT-Gateway, or the opposite, after some time the IoT-Gateway discovers it using the DISCOVERY message. In this case, the network is evaluated when the user places the New Thing observing how the AI assigns it, according to its features, to a workgroup and the architecture layer.

4.1.3. Simulators

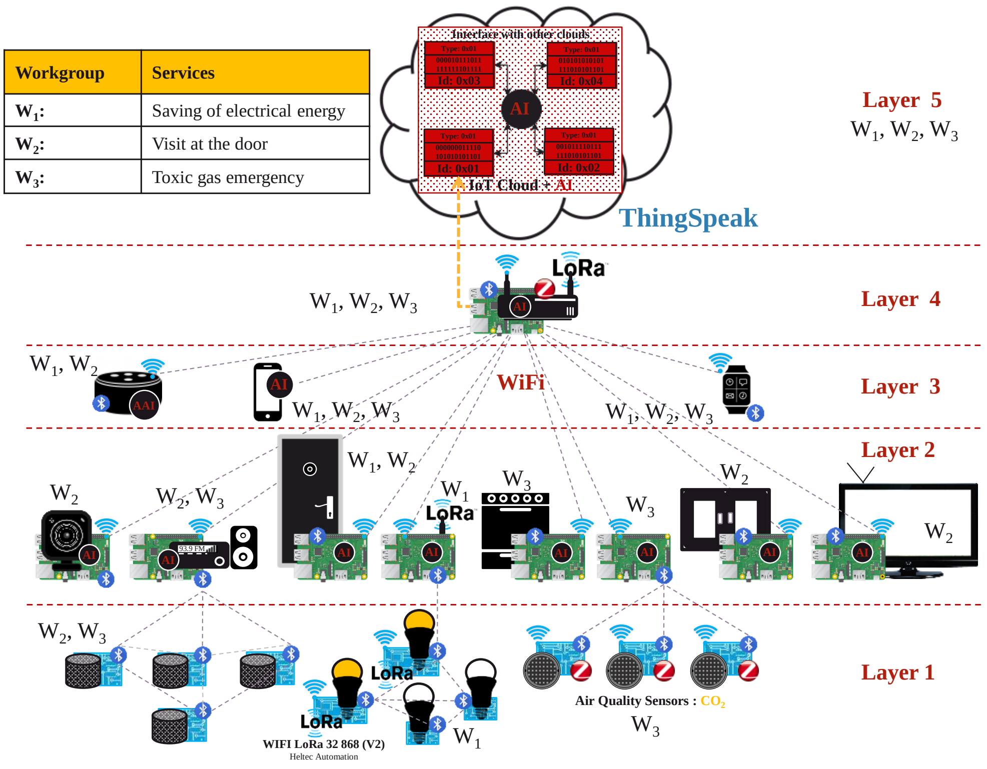
The simulation was carried out with several simulators such as Cisco Packet Tracer 7.3, Jupyter [54], and iFogSim [87], [88]. Each of them complements the other.

With Cisco Packet Tracer, the proposed architecture and model was emulated over an entire network on. This simulator does not have pre-installed network algorithms in RPi3 as in real implementations, but it allows programming in Python and Java. It is a great advantage because it allows modifying and programming the proposed algorithms and putting the network into operation according to the architecture. It can simulate a real connection to the cloud (IoT-Platform: Thingspeak) and capture packets using Wireshark.

With Jupiter, the ML of Table 6 and Table 8 is simulated in Python language, testing each classifier. The goal is to predict where will be assigned a new object according to its features of a group given and previously characterized, depending on whether its features are closest enough to the group or not. The library used was scikit-learn 0.23.2 [55], together with other important libraries for data preprocessing.

4.1.1. Testbed

The simulation scenario is organized in groups and layers, and each group is arranged so that they can collaboratively provide a service. As it was observed in the network model presented in the previous section, an object can participate in providing a service in one or more groups. Therefore, each object in Figure 37 is labeled with W_i, representing its participation in each group. With this scenario, the initial data for ML training is collected.



Workgroup	Services
W_1:	Saving of electrical energy
W_2:	Visit at the door
W_3:	Toxic gas emergency

Figure 37. Simulation scenario organized by groups and layers

The first test is assumed that the IoT-Gateway connects with Wi-Fi to all network objects sending data over the DFSP/MQTT protocol. While the network converges, it storing data in IoT-Gateway, updating the features dataset pre-established. Then, the number of features versus the number of objects in the dataset network is previously analyzed. If there are too many features that are not relevant, this can affect the result. Therefore, a correlation map is made between all the feature profiles that define the connected objects' functions and services. The objective was to evaluate which features were relevant (they had an average absolute correlation coefficient greater than 0.68) to evaluate the workgroup and layer. With this information, the most representative variables were selected to perform the classification.

It caused a change and a minimal reduction in the dimensions of the dataset. The parameters that varied were $X_1 = 20$ decreased to $X_1 = 14$ and $X_2 = 10$ decreased to $X_2 = 7$. It was also observed that the results are affected by increasing the number of objects in the network. When the number of objects was increased while maintaining the same number of feactures, the results improved. Because of a new object added, the number of registers in the dataset increased from n = 54 to n = 59. In other words, five new objects were used in the test.

Figure 38 shows the ML using a classification method based on discriminant analysis to categorize the three groups. The workgroup label was used as the classifier class of the discriminant function to make the prediction. The two discriminant functions with P-values lower than 0.05 are statistically significant with a confidence level of 95.0%.

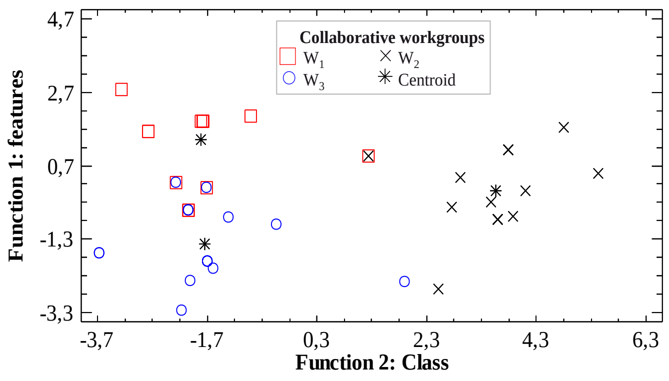


Figure 38. Discriminant Analysis

Figure 38 shows these workgroups as the collaboration between different types of objects joined to attend a service by interpreting its features; in this case, there are three services.

The red squares, blue circles, Xs, and asterisks observed in Figure 38 shows the objects that belong to different groups. Some of these elements are very close to each other, which represents their nearness in their features. It is also observed how some of these features intervene in one or more workgroups. For this reason, it is seen that in some cases, objects overlap one over the other, indicating that there is an interception between the groups.

The results obtained through the discriminant analysis suggest that it is possible to obtain better results by applying a classifier based on the nearness of its features. For this reason, it is initially tested with a K-NN classifier, and the dataset is evaluated to know what precision will be when a new object is classified after it reaches the network.

The best choice of K depends fundamentally on the data; generally, large values of K reduce the effect of noise on the classification but create boundaries between similar classes. Figure 39 shows that with values K = 3, greater precision is achieved, so it is selected for all tests.

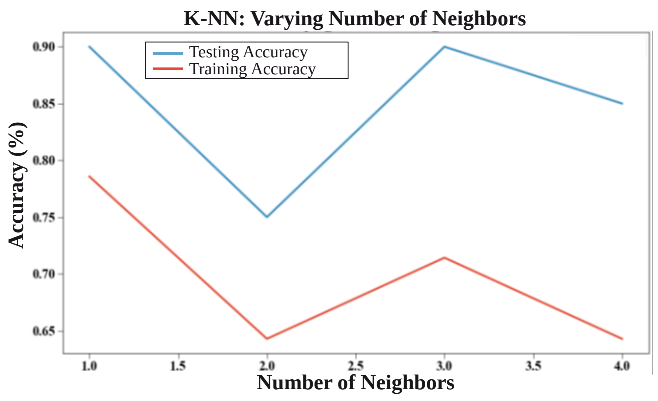


Figure 39. Selection the best value of K

The four distance metrics used to test the K-NN model are defined below: d_e, d_M, d_{Ch}, d_{Mk}. The dataset was divided into 75% for the training model and 25% for the testing model. It is then applied data preprocessing and standardization of the dataset with "MinMaxScaler" and "StandardScaler."

To select the best configuration of the K-NN classifier, we have realized different tests.

First, it is tested with "StandardScaler" and all the distances in metric parameter, and then with "MinMaxScaler." It is observed that with "StandardScaler," most results were high. However, using "MinMaxScaler" with metric = d_{Ch} gives the best result for the "workgroups" class. On the other hand, with the "layer" class, each parameter's best result give using "StandardScaler."

Figure 40 shows the best results for the different tests using K-NN for two classes. The experiment is performed 100 times for each combination of parameters.

The running time that the algorithm takes to learn the parameter in the two best results in Figure 40 is 0,47 seconds for the "workgroup" class and 0,44 seconds for the "layer" class. It is run all experiments using a Python program on a DELL PC with a Core i7 microprocessor and 4 GB RAM.

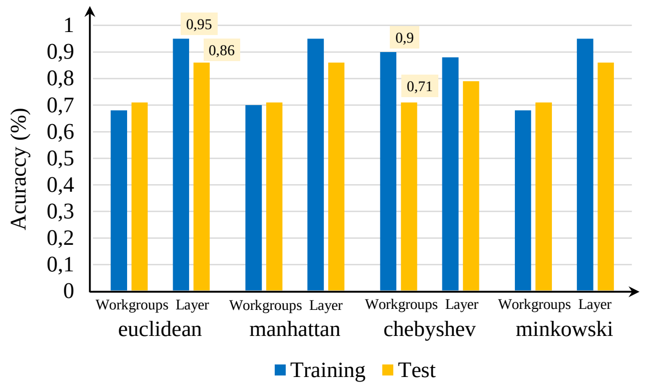


Figure 40. Comparison of distance metric in a KNN classifier (K=3)

Figure 40 compares the results of different distance metrics for each classification label. It is observed in the results that the accuracy ≥ 68% in all the tests, being d_e and d_{Ch} the best. With d_e is obtain an accuracy = 95% to predict an architecture layer and with d_{Ch} an accuracy = 90% to predict the workgroup.

Figure 41 shows the error in the previous tests. Results show very low error for most of the distances evaluated.

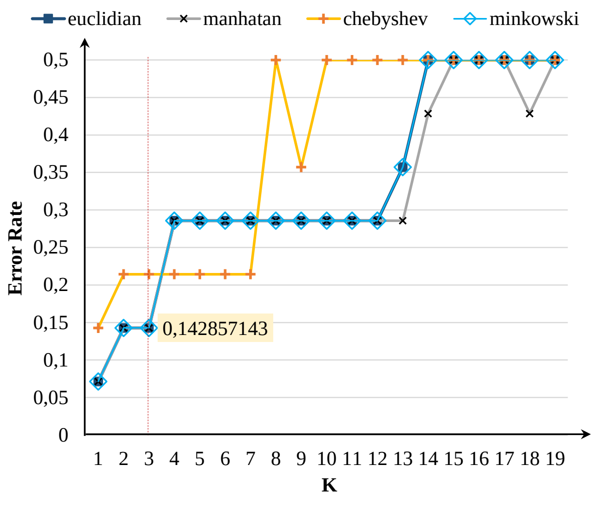


Figure 41. Comparison of Error Rate in a KNN classifier (K=3)

As it is observed in Figure 41 the error rate at K = 3 was 0.14 for most of the distances, while d_{Ch}, which previously showed the best precision for the workgroups, was 0.23.

The dataset obtained from the network test of Figure 5 is small in the number of samples or registers per object. This dataset may have more features (variables) than objects because they are necessary to describe functions, services, relations, resources, and each device's location in the house. However, it can happen that after a threashold, the performance of the model will decrease due to the number of features. If features are continuously added without increasing the number of samples, then the space between the features increases and it becomes more dispersed. Therefore, to adequately process the data and reduce random variables, it is only considered the main variables using dimensional reduction methods. That allows it to remove redundant variables that do not add new information to the dataset and make it easier to view it.

In order to improve the results of the K-NN classifier, it is applied it several dimensionality reduction techniques. The reduction tests were performed for the "workgroup" class with K = 3, metric = d_e, and 100 iterations.

Figure 42 shows the Principal Component Analysis (PCA). It identifies the combination of features and helps examine relations between groups through the main or most relevant features of the dataset. The orange, gray, and red points represent the features of each workgroup. Although it applies a large dimensionality reduction to the dataset, still it is being observed more features than in the other two figures.

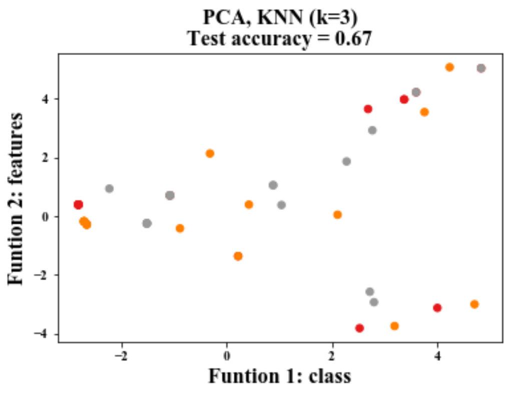


Figure 42. PCA, K-NN (K=3) Test accuracy = 0,67

Figure 43 shows a Linear Discriminant Analysis (LDA). This technique is used to observe the differences between the groups since classifying can cause overlapping and the shared features are not appreciated.

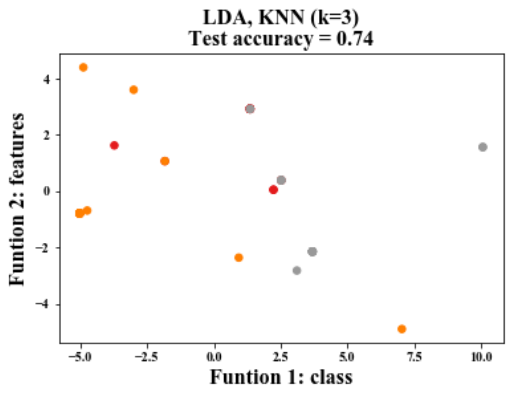


Figure 43. LDA, K-NN (K=3) Test accuracy = 0,74

Figure 44 shows a Neighborhood Components Analysis (NCA). It tries to find a feature space to improve accuracy.

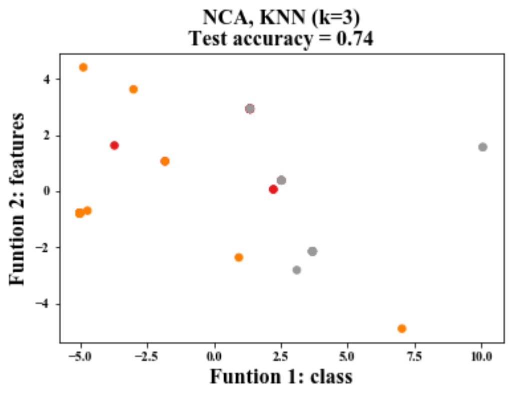


Figure 44. NCA, K-NN (K=3) Test accuracy = 0,74

In any case, none of the three figures shows a clustering of the data that is visually meaningful, like in Figure 38.

The results of these reduction techniques PCA, LDA, and NCA applied to the K-NN model are compared with those obtained in Figure 10. It is observed that using PCA and NCA, the accuracy is improved from 0.71 to 0.74. However, the PCA technique even lowered from 0.71 to 0.68.

After realizing the tests, the K-NN classifier with the best results is selected. We obtained a precision of 0.9 in training and 0.71 for tests with the workgroup class using d_{Ch}. For the layer class, the best results were obtained using d_e with a precision of 0.95 in the training and 0.86 in the tests.

K-NN results are now compared with other types of classifiers, which use different techniques and metrics. Figure 45 shows a comparative graph with the other classifiers.

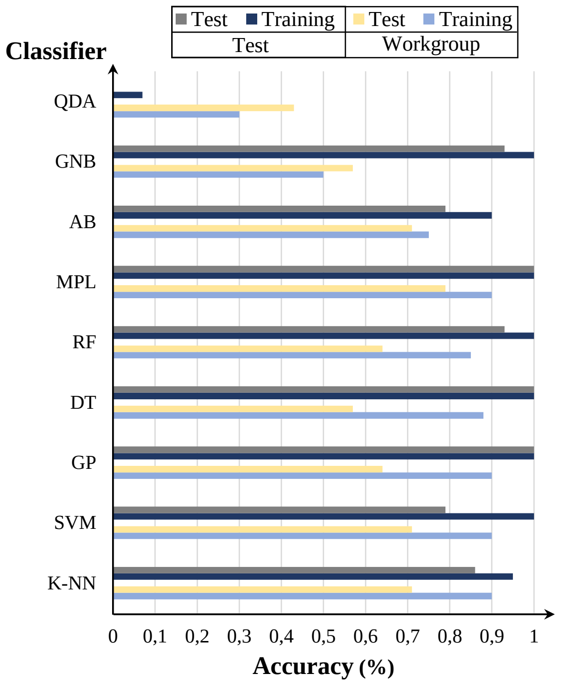


Figure 45. Comparison of accuracy with different classifier

Figure 45 compares the accuracy obtained between the classifiers K-NN, SVM, Gaussian Process (GP), Decision Tree (DT), Random Forest (RF), MLP, AdaBoost (AB), Gaussian Naive Bayes (GNB), and Quadratic Discriminant Analysis (QDA). In the results of the figure, it is observed that the classifier with the lowest accuracy was QDA and the highest was MPL. However, it is also appreciated that the K-NN classifier hasan average value like the other classifiers. The results obtained for the workgroup class with Neuronal Net MPL were 100% accurate in the prediction and 90% for the layer class.

Furthermore, the training time for KNN = 0,44 seconds, SVC = 0,043 seconds, GP = 0,675 seconds, DT= 0,049 seconds, RF= 0,7 seconds, MPL= 0,5 seconds, AB= 0,8 seconds, GNB= 0,5 seconds, QDA= 0,8 seconds.

Figure 46 shows each classifier's error rate, and it can be seen that the MPL classifier has the lowest error of all (workgroup class) compared to the rest. In addition, the class layer for the same classifier has no value.

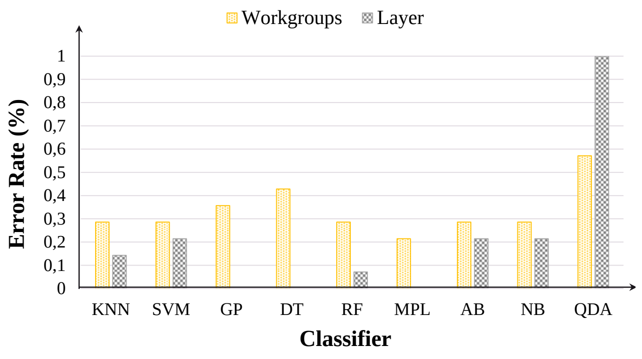


Figure 46. Error Rate's Comparison each classifier

Figure 47 shows the cross-validation technique applied to the models used. In this case, it is using a cross-validation process with ten interactions. This technique just was applied to workgroup class for all classifiers.

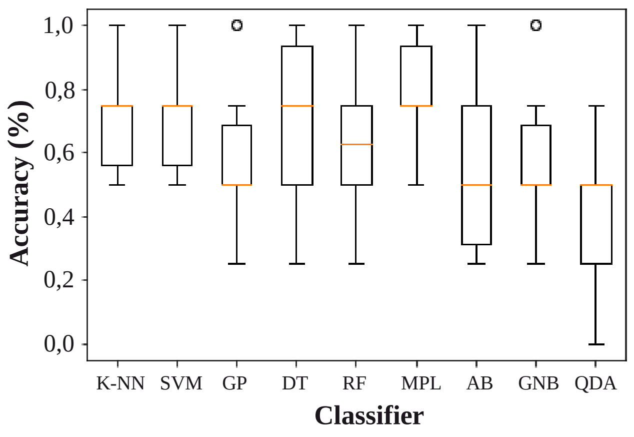


Figure 47. Comparison of models with the cross-validation technique

The figure shows that the best classifier is MPL since the accuracy value (orange line) is the highest, and the upper and lower error margins are lower than the rest. However, the precision depends on the test and training datasets, which may be biased, so cross-validation is a better approximation. The difference between some values compared with those obtained in figure 15 probably is due to inadequate data randomization. Therefore, rather than just measuring accuracy, efforts should be focused on improving the algorithm. If the algorithm is improved, the accuracy will also improve compared to previous approaches.

4.2. Test on Routing

This section presents the experimental part of this paper in order to test the proposed model. For this reason, several experiments are carried out using simulation and different types of tests. Figure 48 shows the objects connected inside a house to an IoT-Gateway. Users use these objects at different times of the day according to their habits. The data stored in the objects and pre-processed are sent to the IoT-Gateway, where they are stored together with data of features and network parameters. From there, the centralized dataset of the entire network is obtained. The objects can be from different manufacturers and have different types of interconnection technology depending on the board capabilities of each object. However, in this scenario, only WiFi and Bluetooth connections are observed for ease of simulation.

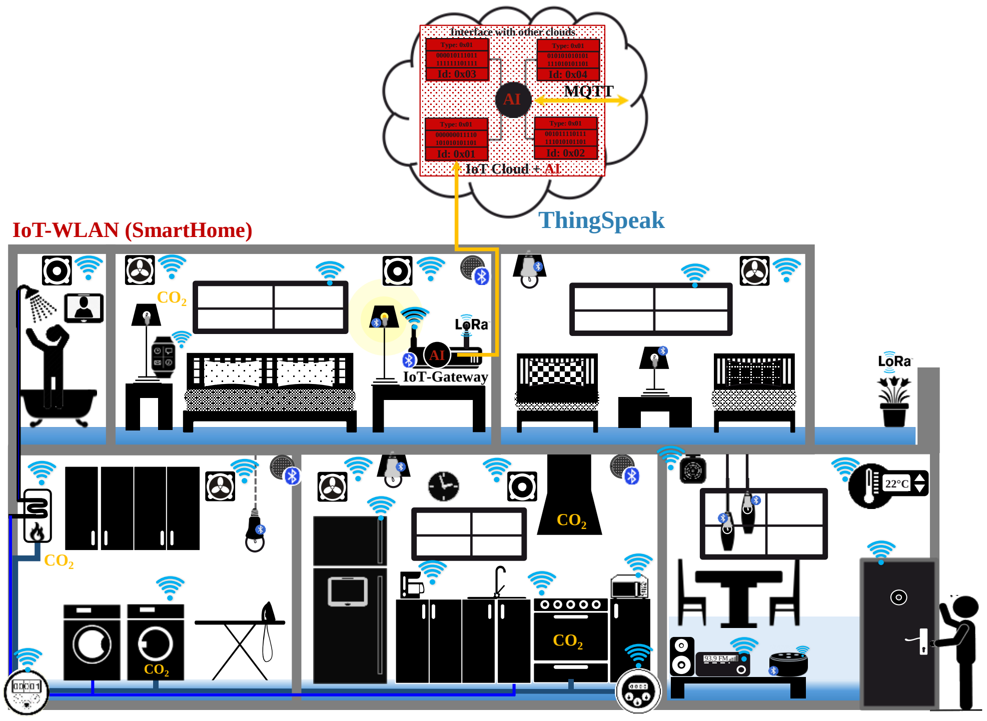


Figure 48. Smart Network Scenery: An Smart Home IoT-WLAN

4.2.1. Implementation of the devices

Most networked objects are appliances, and the others are smart assistants, mobile devices, actuators such as light bulbs and sensors such as gas meters, air quality, humidity, water, and electricity. In the implementation in the simulator, the single-board computers (SBC) such as the Raspberry Pi 3 Model B + (RPi3) [67] were used with which the objects between layer 2 and 4 were simulated. However, the system is designed to be used with a WisGate Developer D4 + (EG95-NA) / US915 [89] as the IoT-Gateway. In layer 1, the bulbs use Bluetooth Low Energy (BLE) technology in Mesh Topology [90] using board nRF51822 [91], and the meters are implemented on a Programmable System-on-Chip (PSoC) as the ESP8266-01.

4.2.2. Considerations for the simulations

It starts with a structured dataset previously created by the IoT-Gateway's AI. The objects were simulated by sending their features and pre-processed data to the IoT-Gateway through the DFSP/MQTT/TCP Protocol [45]. The IoT-Gateway unifies them with the network parameters and the Pub/Sub tables of the MQTT Broker under a single identifier. This identifier is bi which refers to the connected object.

4.2.3. Simulators

A combination of simulators was used to achieve the main dataset, such as Cisco Packet Tracer 8.0 and MatLab, to simulate the scenario in Figure 48. Algorithm 1 with the model of this proposal was simulated using Jupyter Notebook (anaconda3).

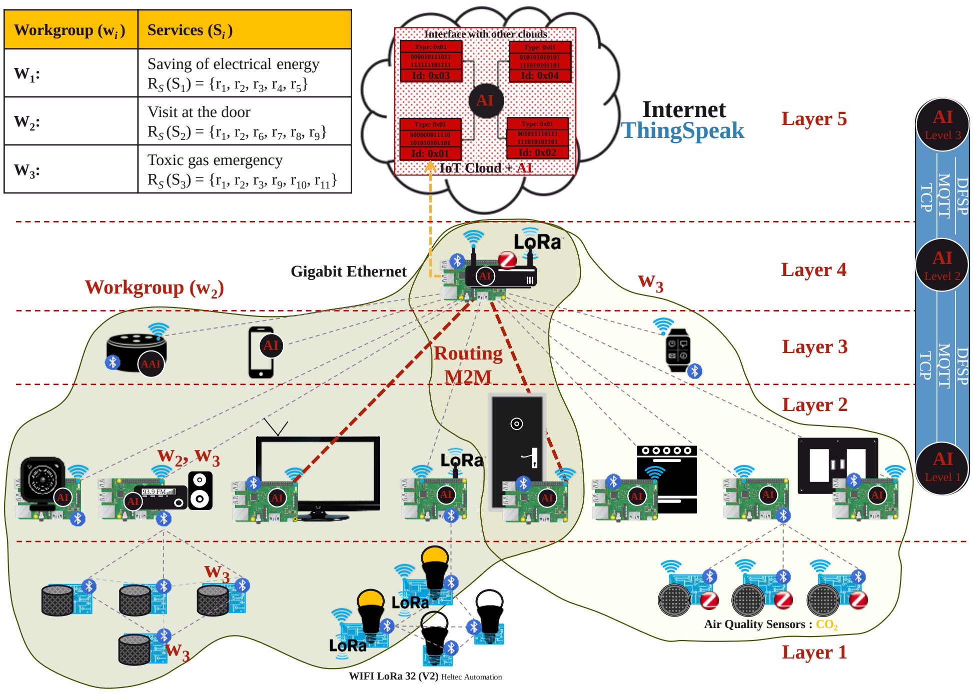
4.2.4. Clustering and routing

From the scenario in Figure 48, only three workgroups are represented and explained through Figure 49. In this figure, the objects are organized following the structure of the IoT-SmartArchitecture to observe the role of each object according to the model. However, also when comparing it with Figure 1, it can be seen how each of these objects is represented as nodes and their meaning in the model.

The IoT-Gateway's AI creates each workgroup to fulfil a service. This grouping was already carried out in a previous paper [56], where the objects were classified in workgroups according to its features of Functions and Services through ML. In this case, the dataset already has the scenario objects classified in groups and an architecture layer. Therefore, in this simulation, an initial state is assumed where the workgroups were previously created.

An object within a workgroup has resources available to collaborate in a service. However, when this resource is exhausted, another object must perform the work or replenish this resource. The simulation shows how the proposed model algorithm can find a route based on the best node with the requested resources.

From now on, the tests are focused on making requests for the necessary resources to the IoT-Gateway's AI to fulfil a service. Previously, the requests in an M2M connection to the Broker were made by the user through topics. With the modification and introduction of this model, the object autonomously requests resources from the Broker.



Workgroup (w_i)	**Services (S_i)**
W_1:	Saving of electrical energy $R_S(S_1) = \{r_1, r_2, r_3, r_4, r_5\}$
W_2:	Visit at the door $R_S(S_2) = \{r_1, r_2, r_6, r_7, r_8, r_9\}$
W_3:	Toxic gas emergency $R_S(S_3) = \{r_1, r_2, r_3, r_9, r_{10}, r_{11}\}$

Figure 49. Case study of M2M routing within a workgroup.

4.2.5. Testbed

Although the scenario in Figure 8 or Figure 22 supports multiple users and multiple services, these tests assume a single user and a single service on the network.

This scenario could have several services such as Saving electrical energy (w_1), Visit at the door (w_2), Toxic gas emergency (w_3), among others. However, it was chosen to do the first tests starting with w_2 as a reference experiment.

Figure 49 shows the objects connected in the network through the IoT-Gateway, which has created collaborative workgroups to fulfil the services $R_S(S_1)$, $R_S(S_2)$ and $R_S(S_3)$. However, the constraints of the model indicate that routing must is done between objects in the same workgroup to fulfil the service. Therefore, each test only uses the data from the same workgroup.

According to the proposed model, Algorithm 1 is used as a search engine to find the best object that provides the requested resources.

The simulation was carried out in Python language on Jupyter. The device's features used was an HP Pavilion personal computer with an AMD Ryzen 7 processor and 16 GB of RAM. The Pymoo [92] library was used to program the NSGAII multi-objective optimizer [93] using the proposed model. With the parameters in Table 17, Algorithm 6 is set and executed.

Table 17. Parameters for the execution of the Algorithm 6

Parameters used in NSGA-II with Pymoo	
n_var = 21	number of variables uses for w_2
n_obj=4	number of objective functions
n_constr=1	number of constraints
xl = 0	variables' lower boundaries as a NumPy array
xu = 1	variables' upper boundaries as a NumPy array
pop_size= 40	population size
n_offsprings=10	number of offspring
sampling= *x*	array *x* with n=18 rows and m=21 columns of the w_2
crossover	get_crossover("real_sbx", prob=0.9, eta=15)
mutation	get_mutation("real_pm", eta=20)
'n_gen', 10	generations number

In this first test, an object b_r from workgroup (w_2) requests from the IoT-Gateway (G_0) the $R_i = \{r_1, r_2\}$ to collaborate on the S_2 service. The test is performed with Algorithm 6, keeping the default configuration defined by Pymoo.

The problem method is configured with the dataset, and the sets $\mathscr{F}$ and g obtained from the analyzed model in this proposal and using table 2. The NSGA2 algorithm method is initially configured with a population size of 40 (pop_size = 40) instead of generating the same number of descendants (n_offsprings = 40), whereby only creating 10 (n_offsprings = 10). In order to implement an alternative that improves optimizer convergence. Additionally, duplicate checking is enabled (eliminate_duplicates = True) so that the mating produces different offspring of themselves and the existing population relative to their design space values [93]. It is also necessary to define a termination criterion to run the algorithm for several generations through the parameter ("n_gen", 10).

Finally, Algorithm 6 calls the minimize method when it solves the problem and algorithm methods.

For this, the functional interface of Pymoo is selected to solve the optimization problem, which provides the minimize method. This method uses the problem, algorithm and termination parameter to find the optimum through the X and F values.

The algorithm was executed for 100 iterations with 4 objectives and 1 constraint, so 400 function calls are necessary for evaluation. Although Pymoo allows using parallelization, in these tests, it was not used. Therefore, the time required for running was t = 0.073 seconds.

It obtains a design space with 18 solutions from which are obtained 5 sets of feasible solutions, including the optimal solution. Figure 50 shows how it represents.

F: [2. -6. 15. 10.]
X = [1 1 0 0 0 0 0 1 1 1 10 3 9 8000 8 72000 1 2 2 1 1]

Figure 50. Representation of a solution X

Figure 51 shows the result of the algorithm evaluated for three functions $f_1(x)$, $f_2(x)$, $f_3(x)$. The red point is the optimal solution, the blue points represent the feasible solutions space and the black points the design space. The optimal solution is found at the coordinate F: [2. -6. 15.].

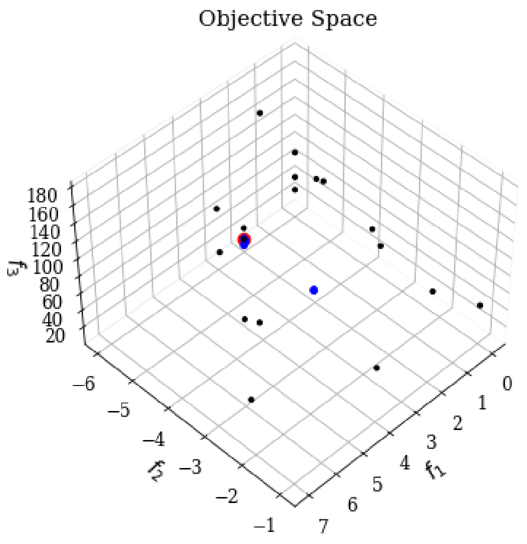


Figure 51. Objective Space for three functions from test 1

Figure 52 shows the four objective functions of this proposal $f_1(x)$, $f_2(x)$, $f_3(x)$, $f_4(x)$, and Figure 4 shows the result obtained of this test, in the coordinates F: [2. -6. 15. 10.].

This is the best node or object b_o within the workgroup (w_2) that can provide the resources with fewer additional resources without affecting network performance. From the data contained in the array X, it can be deduced that the node has the requested resources r_1 and r_2 as [1 1], it does not have additional resources [0 0 0 0 0], it has the highest energy availability [1 1 1], it has the lowest cost and power, calculated from the rest of the X parameters.

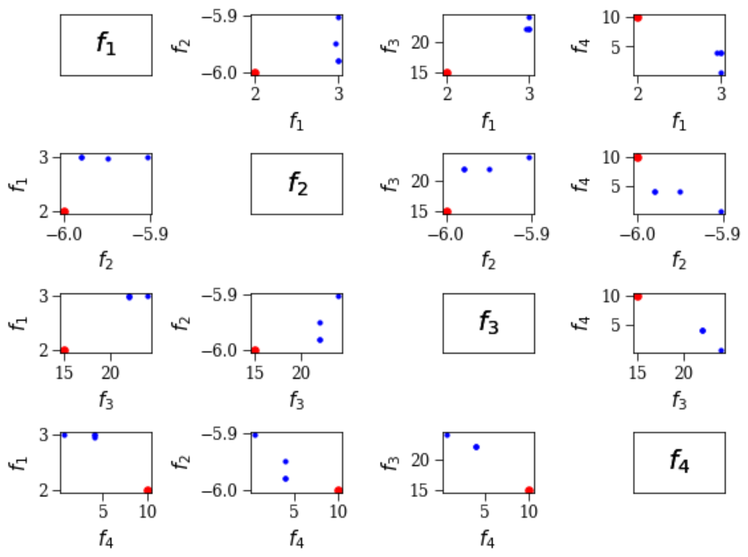


Figure 52. Multi-objective function space from test 1

Figure 52 shows between the first row of graphs and the second column that by following the location of the red point, it is possible to see the sequential composition of the optimal solution. In all three graphs on the $f_1(x)$ axis, the red point is located on the position [2.]. In this same row, in the first graph, it finds on [-6.], and then in the second on [15.] and the last one on [10.]. Finally, it obtains the optimal solution F: [2. -6. 15. 10.]. In the same way, the sequences of the rest of the rows and columns can be deduced.

With the history of the algorithm, the number of each function evaluations and its values each iteration are extracted. Figure 53 shows that the algorithm managed to converge, and a set of feasible solutions and an optimum was obtained.

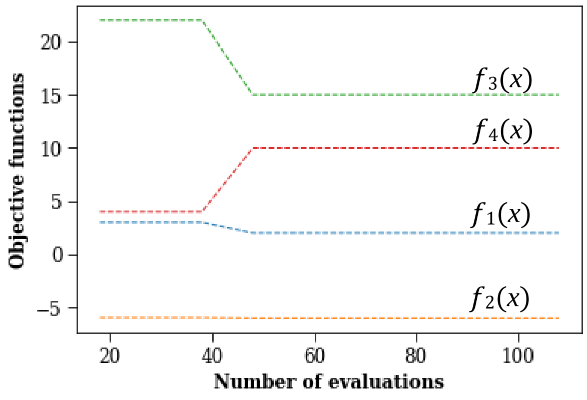


Figure 53. Convergence of objective functions from test 1

The behavior of the objective functions as the algorithm iterates can be seen in Figure 53. In all four cases, it is observed that the functions converge. This occurs approximately after 50 iterations. Function $f_1(x)$ (dashed blue line) converges to a value of 2 and function $f_2(x)$ (dashed orange line) to a value of -6. Otherwise, $f_3(x)$ (dashed green line) converges to a value of 15, and finally, $f_4(x)$ (dashed red line) reaches a value of 10. Once convergence is achieved, the values of functions are taken as optimal.

Figure 54 shows if the constraint is violated or not. The graph shows that the first feasible solution found after 18 evaluations presents an interception of 0.00 CV. Therefore, there is no violation of the restrictions.

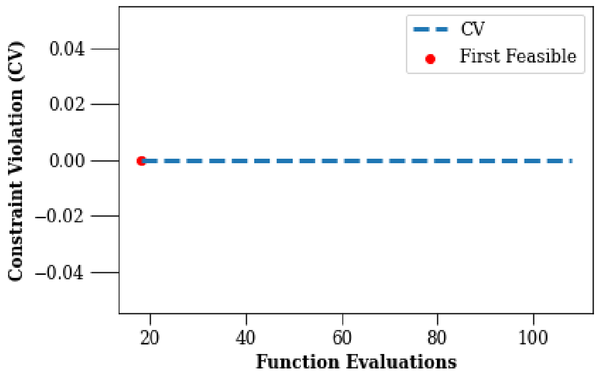


Figure 54. Constraint Violation (CV) from test 1

These results show that, in the first generation, a feasible solution was already found and that the constraints of the problem are satisfied in the initial population.

The second test is performed on the workgroup dataset (w_3), which contains different objects with different features. In this case, the object b_r requests the resources $R_i = \{r_1, r_2, r_3\}$ from the IoT-Gateway (G_0) to collaborate on the S_3 service.

Algorithm 6 is configured the same as the first test with the difference that three resources are requested in this case.

Figure 55 shows the result of the algorithm, the red point being the optimal value in the coordinate F: [3. -6. 11. 25.] and X = [1 1 1 0 0 0 0 1 1 1 25 3 5 8000 4 72000 3 1 2 1 1]. This algorithm took time to execute t = 0.065 seconds and did not violate the restrictions, shown the same results as in Figure 54.

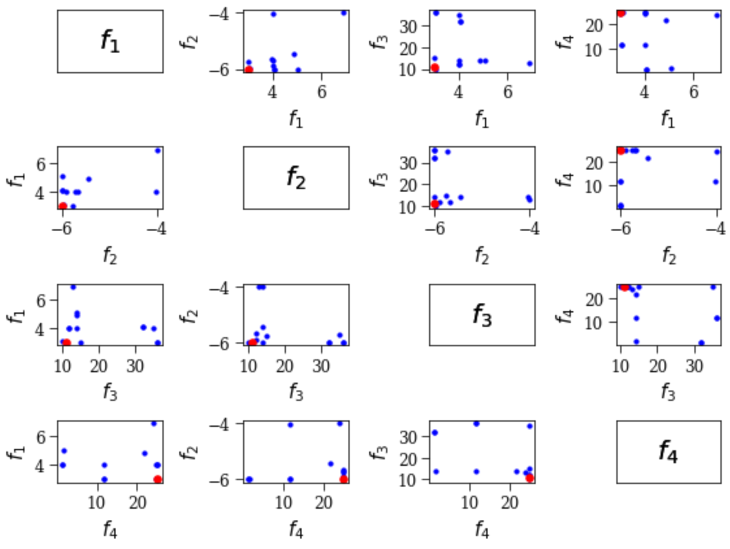


Figure 55. Multi-objective function space from test 2

Following the same interpretation of Figure 52, the result of X is divided into parts to obtain the values of the functions. Whereby the first part corresponds to $f_1(x)$, indicating that the node contains the three resources requested as [1 1 1] that is to say r_1, r_2 and r_3. This is the best node or b_o object chosen by the IoT-Gateway's AI.

Figure 56 shows the convergence of the objective functions in this second test.

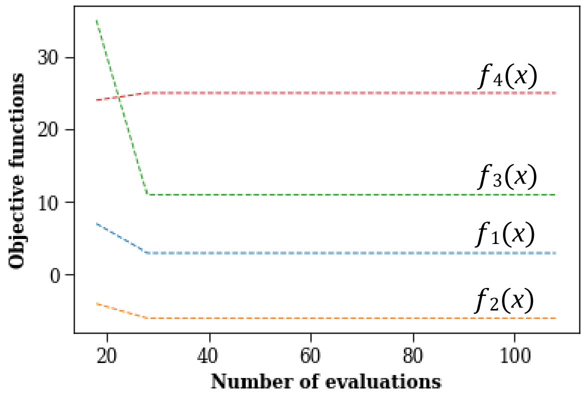


Figure 56. Convergence of objective functions from test 2

It is observed in Figure 56 that for all functions, the algorithm converges approximately after 30 iterations.

In the third test, test 1 is repeated, but in this case, the object b_r requests the resources R_i = $\{r_1\}$ from the IoT-Gateway (G_0) to collaborate on the S_2 service. It is possible that the network conditions have changed and, therefore, the dataset as well. It means that different results can be obtained. For example, the battery of some sensor has gone down or that some resource of the collaborating nodes in the group has been exhausted. Moreover, in most cases, requests are made for a single resource.

The simulation is performed using the same parameter settings of test 1 in algorithm 1. The result obtained is F: [2. -6. 32. 3.] and X: [1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 3 1 1 600 1 1 72000 4 4 4 1 1 1 3]. This algorithm took time to execute t = 0.052 seconds and did not violate the restrictions.

Figure 57 shows the result of the optimiser for the single resource request to the workgroup (w_2).

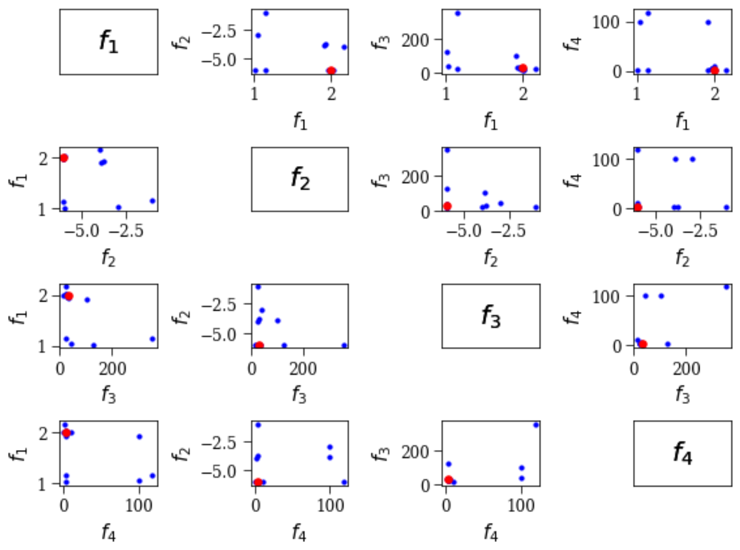


Figure 57. Multi-objective function space for request one resource

The figure above shows that there are more feasible solutions than in Figure 52 of the first test. It may indicate that by requesting a single resource, the algorithm must choose the optimal one from a larger number of candidates. However, the execution time was shorter than in the first test.

Figure 58 shows the new convergence of all functions when requesting a single resource.

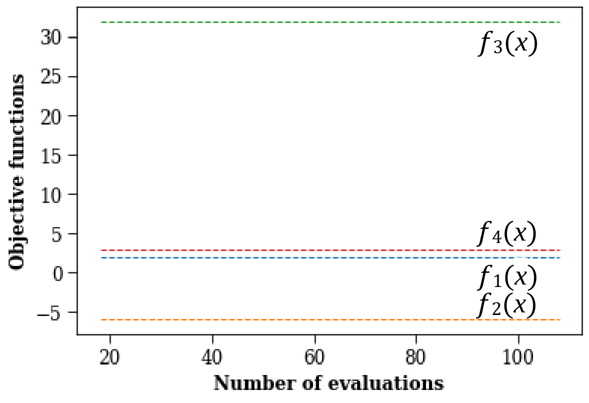


Figure 58. Convergence of objective functions for request one resource

In this case, the algorithm converges approximately after 18 iterations.

In the tests of the three previous cases, it is observed that the algorithm responds well and finds the target node with different requests, changes in the dataset and maintaining the same problem defined in the model of this proposal.

However, the following test is performed by changing the main algorithm NSGA2 of the algorithm method for other available in Pymoo's library. Among them is RNSGA2, NSGA3, RNSGA3 [61]. According to each algorithm, the test was carried out under the same conditions of reference test 1, modifying only the algorithm method and the default parameters. It is important to clarify that not all the algorithms available in this library can be applied since only those requiring multiple objectives are required. Each works with a different technique, but all three find the optimal value. Figure 59 to Figure 61 shows the convergence of each function using each algorithm. Function $f_1(x)$ (dashed blue line) and function $f_2(x)$ (dashed orange line) are superimposed. On the other hand, $f_3(x)$ (dashed green line) and $f_4(x)$ (dashed red line) fluctuate throughout the process.

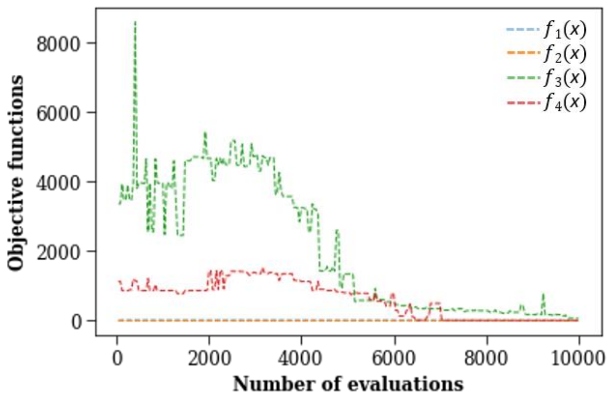


Figure 59. Using RNSGA2 algorithm

In configuring the RNSGA2 algorithm, it was necessary to use the termination criteria with a number of generations equal to 250. Therefore, this algorithm took time to execute t = 5.41 seconds. Analyzing Figure 59 shows that approximately 10000 function calculations were required to reach some level of convergence. However, except $f_1(x)$ (dashed blue line), the values found for the functions did not match those found with the NSGA2 algorithm.

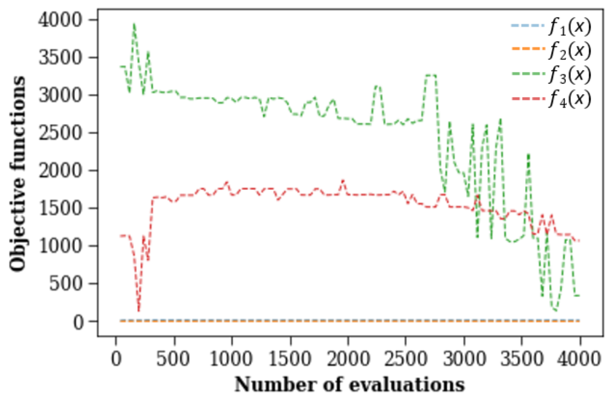


Figure 60. Using NSGA3 algorithm

With NSGA3, it was configured with an offspring equal to 40 and a termination criterion of 100 generations. The time it took to execute the algorithm was 1.57 seconds. In Figure 60, it is observed that after having evaluated the functions 4000 times, $f_3(x)$ and $f_4(x)$ still do not converge. Furthermore, $f_1(x)$ and $f_2(x)$ converged to values that are not optimal.

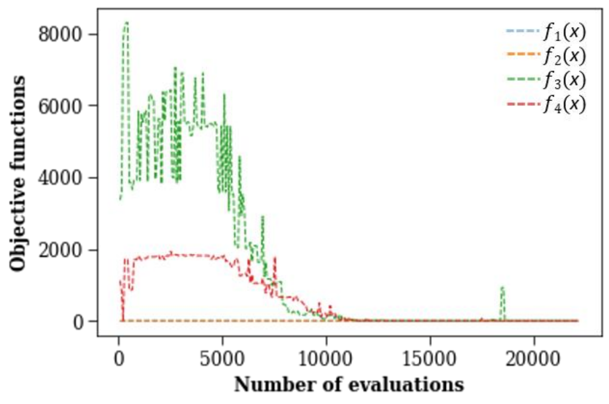


Figure 61. Using RNSGA3 algorithm

In order to obtain the optimum with RNSGA3, it was necessary to configure the termination criteria in 300 generations. The time it took to execute the algorithm was 7.54 seconds. The figure shows that the algorithm needs to calculate all the functions approximately 11000 times before converging to the values. Additionally, the values found are not optimal for this problem.

Comparing Figure 59 to Figure 61 with Figure 53 shows that the NSGA2 algorithm requires much less computation of the functions to reach convergence. Moreover, this algorithm is the only one of the four tested that finds the optimal function values. On the other hand, if we compare the execution time required by the four algorithms, NSGA2 is the most efficient.

4.3. Test on Quality of Experience (QoE)

In this section, the tests performed and the results obtained are discussed. First, the classifying model will be tested in order to obtain goods parameters. Then, the QoE will be measured to check if the introduction of an RL algorithm improves it.

4.3.1. Classification analysis

The classifying model was defined based on two parameters. These parameters, the number of hidden layers n_h, and the number of neurons in each hidden layer m_h, are analyzed in this subsection. In order to get the better values for these parameters, the model will be trained, given the same dataset, for the values of n_h, and m_h shown in Table 18.

Table 18. Parameters for Classification analysis

Parameter	**Val 1**	**Val 2**	**Val 3**	**Val 4**	**Val 5**	**Val 6**	**Val 7**
n_h	1	2	3	4	5	-	-
m_h	1	2	5	10	12	15	20

The results of the evaluation of the model for each pair of parameters values are shown in Figure 62. In the figure, the accuracy for each number of neurons is shown for each model with a different number of hidden layers. The size of the training dataset corresponds to two weeks of user activity. Although the mean of the values of each series could be interesting for other experiments, we only need here the maximum value of all the series to select the values of the parameters with the best accuracy. In this case, the model works better with this kind of data when it has 5 hidden layers and 10 neurons per hidden layer. With those values, the accuracy of the classifying model is 54%. Despite the fact the accuracy might be improved with further feature selection or with larger datasets, this is a good accuracy to prove the hypothesis of this work.

Furthermore, the results of the deep learning model are tested against another classifying algorithm. In this case, we chose the KNN algorithm to implement a classifying algorithm that predicts the next service. In this case, the KNN algorithm is parametrized depending on the number of neighbors. Figure 63 shows the results obtained from the KNN algorithm for a number of neighbors from 1 to 18. The maximum obtained accuracy, with the same dataset that the deep learning model, is 41.21% with 3 neighbors.

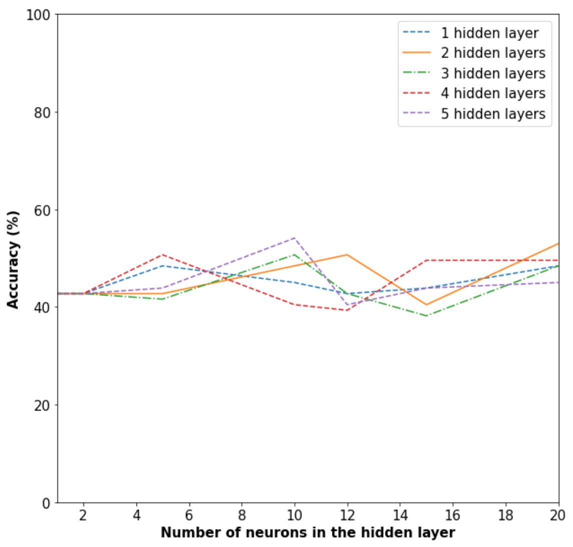


Figure 62. Classification model results

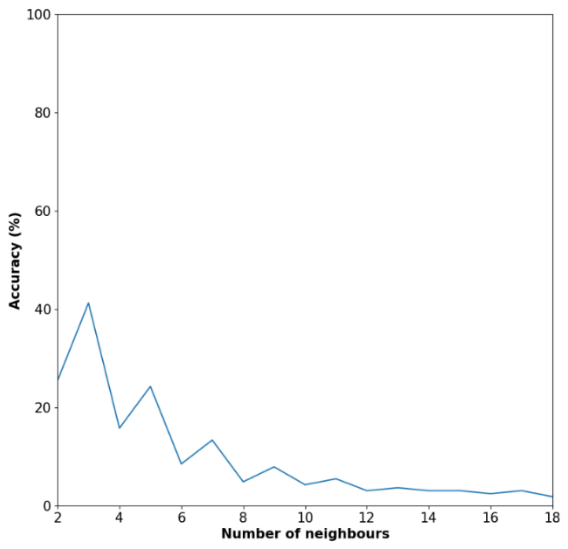


Figure 63. KNN classification results

Figure 64 shows an example of how preparing the data from the Smart Home environment may help to get good levels of accuracy. The figure depicts the violin graph of the feature “type of day”. This feature has two values: 1 if the day is a working day 0 it is not. Since the services with higher ID are the services regarding multimedia and leisure (see Table 12), the probability of having a high value in service is higher when the type of day is 0.

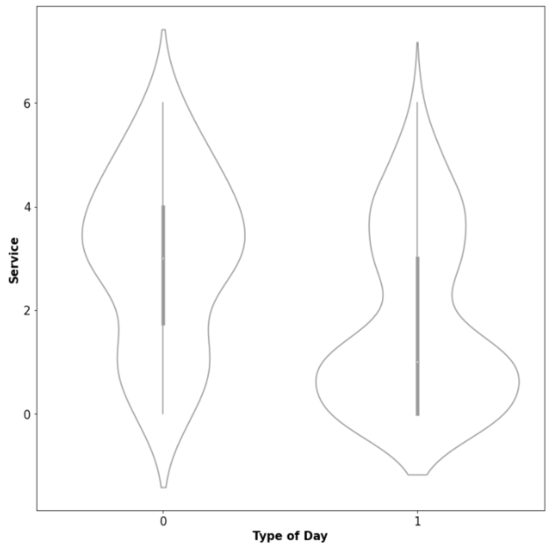


Figure 64. Violin graph of service depending on the type of day

Finally, Table 19 shows the running time of the different parts of the algorithm. We have measured the mean of the running times from the different parts of the proposal. The experiments were carried on an AMD Ryzen 5 5600X with 32GM RAM DDR4. The results show that the times fit the environment restrictions, which are not extremely demanding in terms of running time

Table 19. Running Time

	Data Preprocessing	**Classifficator (KNN)**	**Classifcator (NN)**	**Reinforcement Learning prediction**
Running Time (ms)	24	9	21	1

4.3.2. QoE in Smart Home results

In this section, the most important metric for this work, QoE at Smart Home, will be calculated for a KNN classifying system, the deep learning system and the deep learning system with the RL as a supervisor.

However, we need to define first how we calculate QoE applied for the Smart Home scenario. At Smart Home, unlike in multimedia scenarios, the delay, bandwidth and jitter will not be our concern. We are going to put aside the network resources and focus on the user. Thereby, the QoE decreases when the user is interrupted with consults about services they do not want to use. Moreover, if the system starts a service the user does not want to consume or to stop a service the user wants to keep consuming, the QoE will also be decreased. After each prediction, the QoE will be updated based on the loss using the function λ in Eq. 39.

$$\lambda = \mathrm{L}(\mathrm{y}, \hat{y}) * (1 + \mathrm{multimedia}) \qquad \text{Eq. 39}$$

Where $\mathrm{L}(\mathrm{y}, \hat{y}) = \begin{cases} 0 \; if \; y = \hat{y} \\ 1 \; if \; y \neq \hat{y} \end{cases}$, and multimedia refers to the current service.

Note that the loss defined is only applied to the classifying. For the RL algorithm, the loss is defined by the user input matrix, depending on the action chosen.

Therefore, let α be a parameter to weight the decreases, the total QoE after the n iterations (transitions) of the system is determined by Eq. 40.

$$\mathrm{QoE(n)} = \mathrm{Max(QoE)} - \frac{\sum_{i=0}^{n} \lambda_{i}}{\alpha} \qquad \text{Eq. 40}$$

In order to maintain similarity with the QoE defined in multimedia, the max value of QoE in our experiments will be 10. As regards α, it can be adjusted depending on the length of the measurement. We will show three different experiments where the parameter α is different, but the number of iterations does not vary. The number of iterations will be 88 for each measurement.

Figure 65 depicts the QoE obtained during all the iterations with an α value of 20. With this α value, the RL algorithm obtains a QoE of 8.7. the deep learning algorithm obtained a QoE of 6.56 and the KNN algorithm of 5.89. Although the RL obtains a high QoE, the other two methods are not too far from each other in terms of QoE. Despite the good QoE obtained with the classifying methods without RL, the accuracy of the predictions has been low. This α value only shows a real behavior when the user cares less about interruptions.

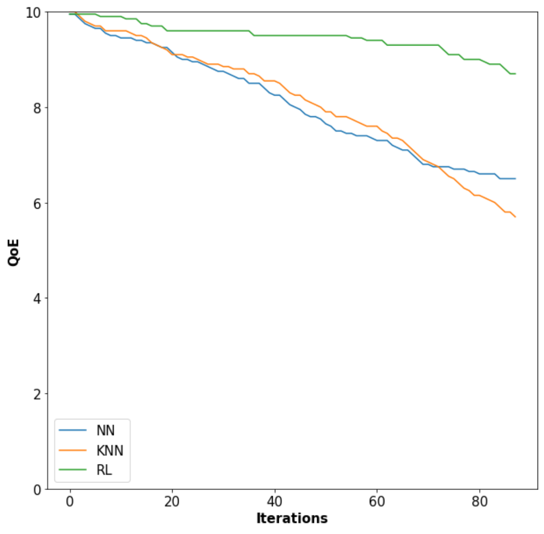


Figure 65. QoE after 88 iterations with α=20

The QoE values obtained with an α value of 15 are shown in Figure 66. Again, the RL algorithm obtains better QoE than the other alternatives, with an 8.26. In this experiment, the deep learning algorithm obtains a 5.53 of QoE. The KNN algorithm is again the one that provides the worst performance with a QoE of 4.26. These results might represent with more accuracy the average user, whose QoE decreases with the interruptions. In addition, the RL presents the least decrease with the change of the α value.

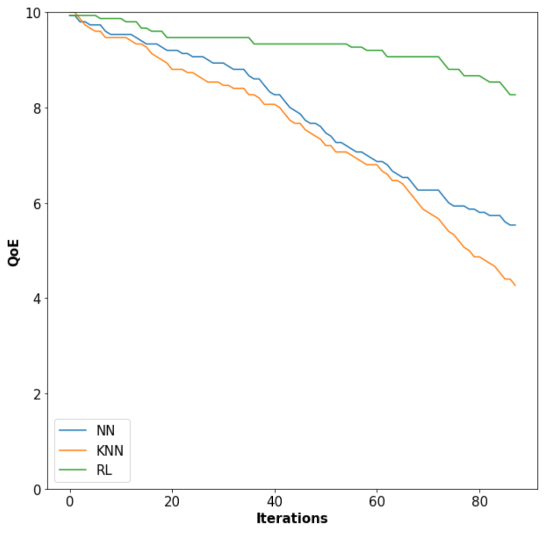


Figure 66. QoE after 88 iterations with α=15

Finally, Figure 67 illustrates the results when the parameter α is reduced to 10. In this case, the bad predictions that have as a consequence to interrupt ongoing services decrease quickly the QoE. The RL supervisor, by choosing actions that prevent interruptions when the classifying algorithm does not give an accurate prediction, presents the best QoE with 7.39. One more time, the deep learning classifying model presents better results than KNN, with 3.59 and 1.53, respectively. The decreased pace of the QoE, in this case, may show an unrealistic situation where the user punishes the system too much. However, it shows how the RL algorithm improves the performance of the system.

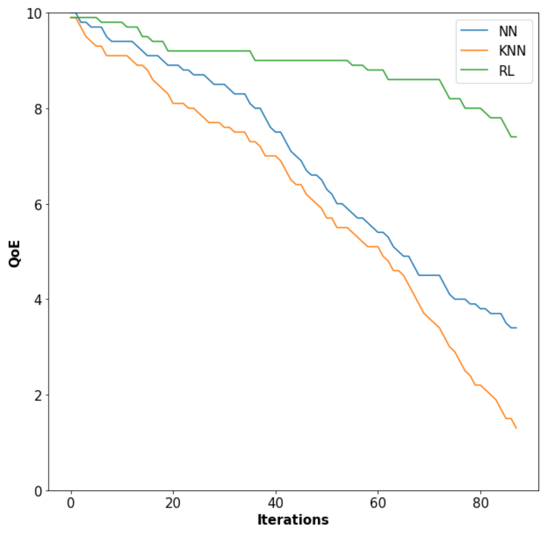


Figure 67. QoE after 88 iterations with α=10

4.4. Conclusion

Smart Things need to exchange much more information than a sensor node, while there is not a standard protocol that carries more information, as showed in the simulation, working with MQTT is better to achieve the goal.

In future work, we plan verifying that the DFSP algorithm over the MQTT protocol by implementing over using real devices and obtain measurements at runtime and obverse the energy consumption.

The following work consists in showing the results of this system implemented in a real environment, with centralized administration through a Multiprotocol IoT-Gateway controlled by AI.

It is expected that in the next results on the real network, it will be possible to observe and measure the designation of M2M relations by the AI and how the AI manages Internet accesses to solve problems.

The K-NN classifier, being a simple method, is ideal for classifying the most similar data points, and it is also easy to implement, although compared to the other classifiers, it was not the best. However, it remained within the average results. The tests showed that the MPL classifier is the best to classify both classes. The obtained results were very high compared to the other classifiers. It is necessary to continue testing other classifiers for the group creation algorithm with ML. However, the tests showed that the architecture design widely allows the use of ML for its operation.

It is expected to perform architecture tests on a real network, testing the clustering algorithm used in this simulation and observing its behavior when a new object joins the network. Thus, larger input datasets could be handled by further improving the classifier's results, e.g., proposing a recurrent learning classifier such as RBNN or a deep learning technique.

The simulation results show that the proposed model algorithm allows the IoT-Gateway's AI to find the best node and establish an M2M connection. Furthermore, it is shown that it allows evaluating several criteria, functions with different values. For the chosen criteria, the values converged being $f3(x)$ the function with the highest value. In addition, the constraint was not violated. Consequently, the proposed model may implement the routing protocol in the IoT scenario. With this proposal, objects will not have to know in advance the network topics to control other objects, it is enough to request a resource, and the AI connects them.

This proposal, however, opens several lines of future works. Firstly, the criteria chosen in this work are generical criteria for every IoT environment. However, this protocol allows us to define functions depending on our needs. Therefore, the study of objective function selection for specific IoT environments such as Smart Home or Industrial IoT will be addressed in the future. Nonetheless, other improvements can be applied to the protocol. The AI can be enhanced with other algorithms. The optimizer proposed in this paper can be complemented with other ML techniques. In future works, we will study the application of reinforcement learning techniques to explore other functions or features to select the best criteria, a solution already studied in other areas like software defined networks (SDN) [94].

We have also defined QoE for this scenario and we have measured it for different parameters and systems.

Results show that the deep learning classifying model proposed achieves better accuracy than other algorithms like KNN, improving their performance around 33%. The QoE of the deep learning algorithm shown with different values of the parameters has always been higher than the KNN algorithm. With high values of the parameter α, the QoE obtained is higher, being in the experiment 5.53 for the deep learning algorithm and 4.26

for the KNN. This difference, based on the difference in accuracy, gets higher when the weights of the prediction increase. That is when α has lower values. This difference raised up until 2.06, with QoE values of 3.59 for deep learning and 1.53 for KNN. Nevertheless, the most remarkable results are the ones obtained from the RL system, which manages the predictions to reduce the impact and obtains QoE values of 8.7 when α has the highest value and 7.39 when it has the lowest. That makes a difference of 3.17 in the first case and 3.8 in the second one. That shows that the inclusion of an RL system improves the QoE in Smart Home environments when the classifying cannot predict the next service with high accuracy.

There are several aspects that can be studied in future works. Firstly, the system can be evaluated against better classifying models. That would include the study of improving the accuracy obtained in this work. The feature selection could be enhanced, reducing dimensionality [95] or an automatized label system could be included in the system [96]. Other parameters, such as the activation function or the optimization parameters could be changed to improve the accuracy. The scalability of the system could be studied. Mechanisms to adapt the system to new services could be defined to efficiently control a bigger number of states. Moreover, the system could be compared to other classifying models, besides KNN. The QoE metric could add an intelligent method to select the α parameter based on the user profile. Thereby, some actions to correct low QoE could be included in the system. Finally, this system could be adapted to face other problems in IoT, such as video surveillance or Industrial IoT environments.

Chapter 5. Implementation of the proposed architecture

5.1. Air quality monitoring in a smart city

The air we breathe is essential to living and is increasingly contaminated by different factors affecting the health of all living beings. In the process of breathing, it is not possible to know through the upper airways (nasal and buccal cavities, pharynx, larynx, and trachea) if the air is clean and without risks to health. Some environments are more polluted than others and with different types of toxic gases and concentrations that, combined with air, are imperceptible to our senses.

Intoxication by continuous inhalation of gas in high quantities and for a long time can lead to the death of a person or family group, especially in homes with poor ventilation. But, also when a person moves from his home through the city to his work can be dangerously exposed to different environments with toxic gases, making their usual route from home to work every day.

Due to the above, it is very difficult to take air quality data in different environments in which a person moves in the day, and thus establish, what was the amount of toxic air to which he was probably exposed, and use them to know how it affects your health.

5.1.1. Proposal

The following proposal consists of a collaborative system of things that communicate with each other within a Wireless Local Area Network (WLAN) extended to other networks through an intelligent platform in the cloud (AI-Cloud), under the concept of the Internet of Things (IoT) [12]. This system is implemented experimentally using IoT protocols and technologies with the aim of solving a case study based on the need for a user to have a wearable intelligent assistant that manages the information regarding the quality of the air that it breathes in three different environments (at home, at work, and through the city). Allowing control and establishing acceptable parameters of air quality to reduce possible risks of respiratory diseases.

This document addresses the study of the problem through the methodology of a case study based on a scenario.

Each scenario is an IoT-WLAN network interconnected through an intelligent platform in the Cloud (AI-Cloud) that monitors air quality.

In the Figure 68 it is observed, the first scenario is located in a Smart Home. For this specific case, it was chosen to study the air quality in this environment, starting from the Machine-to-Machine ratio (M2M) between the following actors: smart gas stove and gas heater, smart ventilation system (which is made up doors, windows, ventilators, and extractors that take actions of opening or closing through requests on the network) and a system of measurement of all types of gases, which are interconnected with each other, with the target provide a safe environment for the user. This implementation is made through the use of computer learning algorithms (Machine Learning) and a network connection using WiFi and Bluetooth technologies [97] with MQTT Information Protocols over a centralized management architecture based on a Gateway with Artificial Intelligence (AI-Gateway).

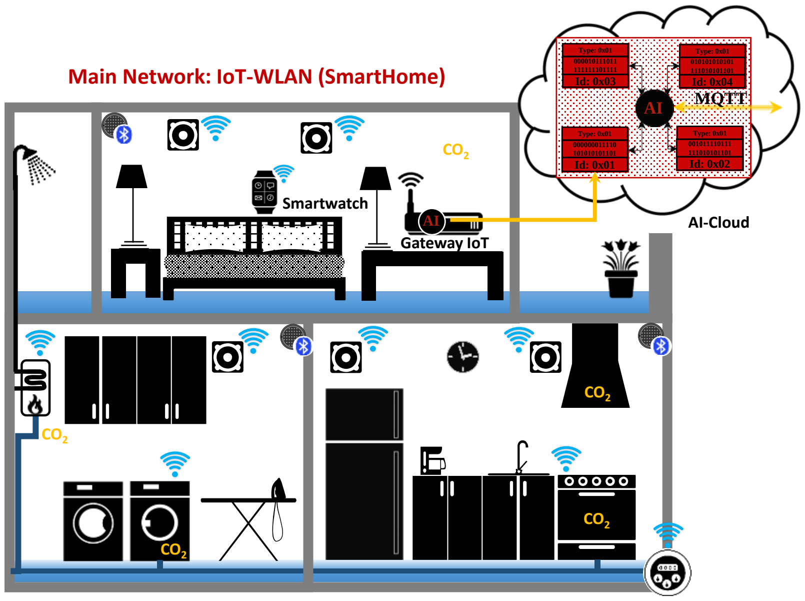


Figure 68. First scenario: SmartHome connected to AI-Cloud

Figure 69 shows the second scenario, which is located in the streets of the city. This IoT network is responsible for monitoring, through a system of sensors, the quality of the air breathed by the user in different areas of the city, when he goes from home to work. This measurement is made through several Gateway with LoRa technology located at strategic points of high pollution in the city. Each Gateway connects to the cloud through its own platform and its own LoRaWAN protocol. In the cloud, the LoRaWAN platform communicates with AI-Cloud through the MQTT protocol to interconnect the networks.

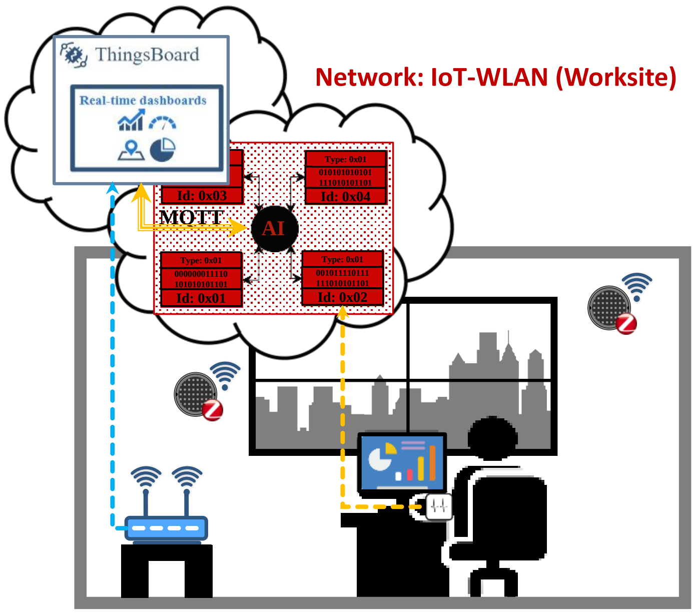


Figure 69. Second scenario: Worksite connected to ThingsBoard Cloud

The third scenario seen in Figure 70, uses the ZigBee [98] and LoRa technology [99] to collect the information of air quality sensors of the central monitoring system of the user's work site, through an open source IoT multiprotocol Gateway called ThingsBoard [68]. The data are published and transmitted to the AI-Cloud through the MQTT protocol, interconnecting the three IoT-WLAN networks.

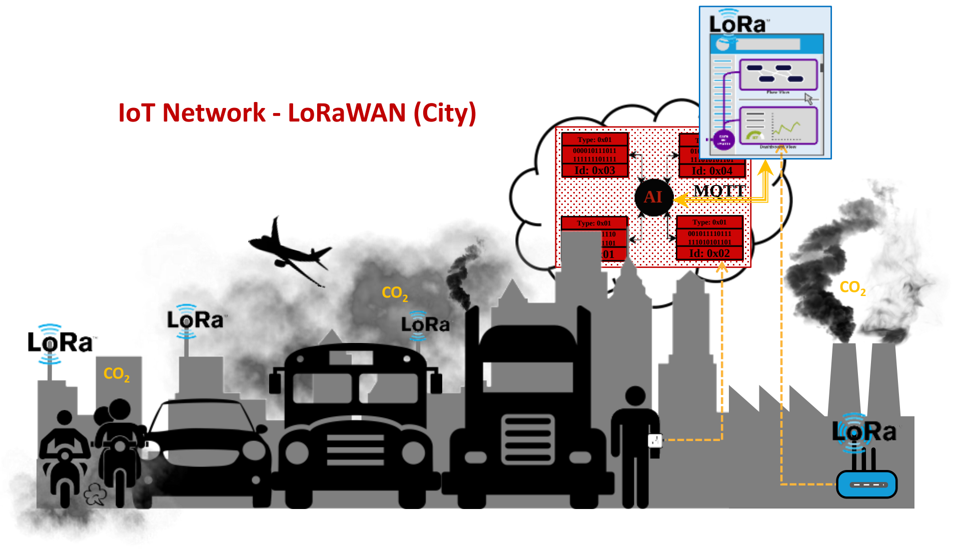


Figure 70. Third scenario: City connected to LoRa Cloud

5.1.2. Interconnected data network (Integration cloud)

The main network (SmartHome), is focused on maintaining a suitable environment, free of toxic gases, for the user. We propose an algorithm to allow the conversation between the machines and to solve by itself a local problem in the network and so make more efficient use of the Internet connection. The Artificial Intelligence (AI) system constantly measures the levels of Carbon dioxide (CO_2) and Volatile Organic Compounds (VOC) in the house and if it finds any risk, it solves the problem. For example, the AI system inside the house can know the location and vital signs of the user through the watch he is wearing, and know if he is asleep or if he has suffered any health alteration according to the air quality system reports (Bluetooth 5: Thingy52).

When an event occurs in which the levels of CO_2 increase inside the house, it consults on the internet if they are acceptable levels for the user and it is establishing risk indicators, if the event is due to a gas leak due to some failure, the system solves the problem autonomously and reports the failures to the user or if the event is more serious it calls the emergency systems through the Internet, safeguarding the user.

In the Figure 71, the connection of the Gateway to the intelligent platform in the cloud is done through the Internet and constantly monitors what happens in the house. If it is realized queries to the cloud, the platform validates the type of request, classifies it and decides if it is necessary to connect to other platforms or if the problem can be resolved internally through the M2M connections.

Another risk is the time of daily exposure of toxic gases in other environments that is outside the controlled environment of the smart home. When the user leaves the house, it is out of reach of the intelligent assistant of his house and enters other environments where there is little air quality information or nothing.

Some cities already have IoT networks to monitor air quality, but they are very new and little known. In addition, they do not have good coverage, but they are still growing, which means that in the future we will have more information available about CO_2 concentrations in some sites with higher incidence. Such is the case of LoRa networks [99], which currently use SmartHuman sensors (Air Quality Monitor) with LoRaWAN protocol [100]. These are connected through Gateways strategically placed throughout the city, forming a network of its own.

Each sensor connects to the nearest Gateway and the Gateway connects to a management platform via the Internet. So when the user leaves his house, during the route to his work the clock he carries connects to the Internet and to the AI platform of the house that is in the cloud. The AI knows the location of the user through the watch and looks to the cloud closest to its location, so it can report information about the pollution indexes of the area in which it is located. When the user arrives at your workplace, where the user spends most of his time on a weekday, he switches to another environment with other pollution indices. Depending on the type of work to which the user is dedicated, the type of contamination also changes. If it is an office job, the contamination is relatively low, but if it works in the industry and if it is dedicated to the manufacture of different types of elements, the pollution could be higher and also with different pollutants in the environment

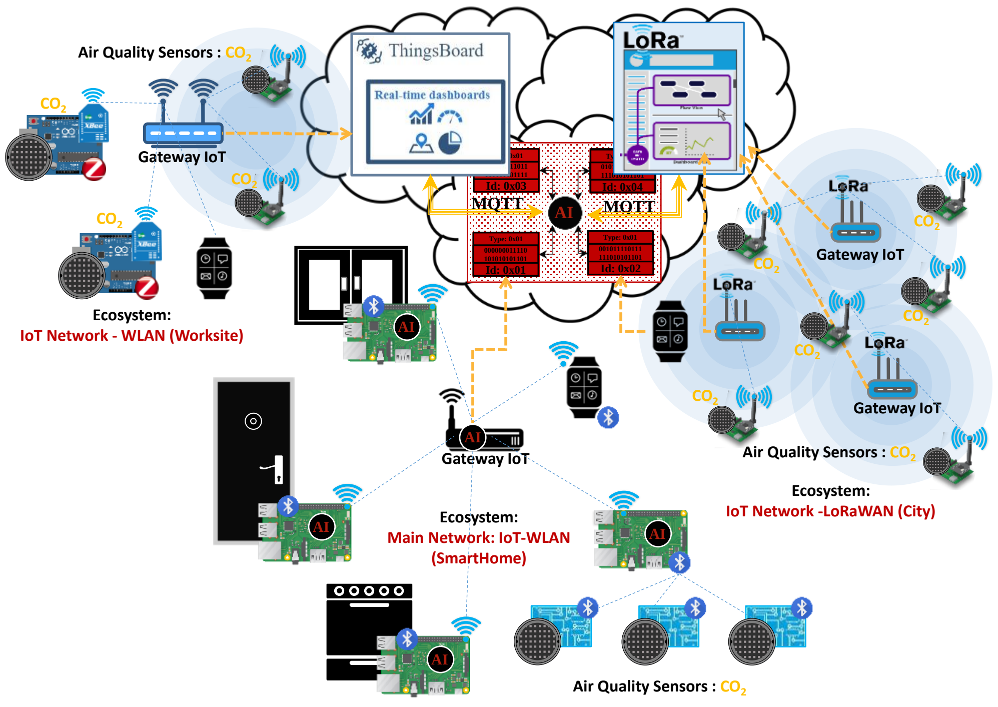

Air Quality Sensors : CO2
ThingsBoard
Real-time dashboards
LoRa
Gateway IoT
MQTT
AI
MQTT
Ecosystem:
IoT Network - WLAN (Worksite)
Gateway IoT
Ecosystem:
Main Network: IoT-WLAN (SmartHome)
Air Quality Sensors : CO2
Air Quality Sensors : CO2
Ecosystem:
IoT Network -LoRaWAN (City)

Figure 71. Interconnected data network (Integration cloud)

However, this work is aimed at measuring relatively low CO_2 values in a work environment, with the aim of measuring the levels of exposure of a user at the end of a day. Upon returning home, the user and the AI system will know the level of contamination to which the user was exposed within 24 hours, during their usual route. The clock also takes the vital signs of the user in the same 24 hours and crosses the data to send them to a health professional who can interpret the information and diagnose if this has influenced in some way to your health.

5.1.3. AI operation

The monitoring network of each environment (Smart Home, City, Worksite) has a multiprotocol gateway that allows it to manage and centralize all of the information regardless of the underlying technology of interconnecting.

The proposed algorithm is shown in Figure 72. Each Thing connected inside the network, makes an exchange of requests and inquiry through the protocol messages. These messages are managed by an AI algorithm in the Gateway, which tags them according to its type of parameter and then forwards them to the destination Things inside the network WLAN or outside the local network through the Internet. The Gateway manages them centrally and forwards them using the AI. It will decide, if the packages are forwarded within the same network or outside the network, depending on the type of relationship and the type of resolution to a problem. In this way, the Gateway forms a group of evidences to verify if there is a clear reason for the requests to be sent out of the network. If not enough, the Gateway will decide that the problem can be controlled locally. Once the data is sent out of the network, they travel over the MQTT protocol making use of another type of message where the information that will be served by the IoT platform in the cloud is packaged. Inside the cloud, the AI algorithm is more complex and requires more resources to operate. For example, the management of large databases (big data) and permissions to give the necessary security to the entire system.

In the cloud, the IoT platform is divided into sections with an identifier (Id) for each connected network, becoming an interface that receives and classifies the information according to its types of parameters. Depending on the type of service, it organizes horizontally in interfaces to give greater flow to the information and more processing capacity to the IA. The data sent from the Things are organized in the platform by the AI in groups of parameters common to the IoT networks that are linked to the cloud. In this case, the group of parameters to process corresponds to a monitoring system on air quality and the service layer could be: Alarms for overexposure to high levels of toxic gases, exposure time and geographical location. In this way, the messages issued by the Gateway of each network, would add to the package a header with information regarding the type of the parameter and the AI would be responsible for deciding which service it belongs to and would also add it in the header. The body of the message, would take the data processed by each protocol depending on its destination. In this sense, the function of the protocol is more relevant when the communication is established between the IoT-Gateway AI and the IoT platform AI in the cloud.

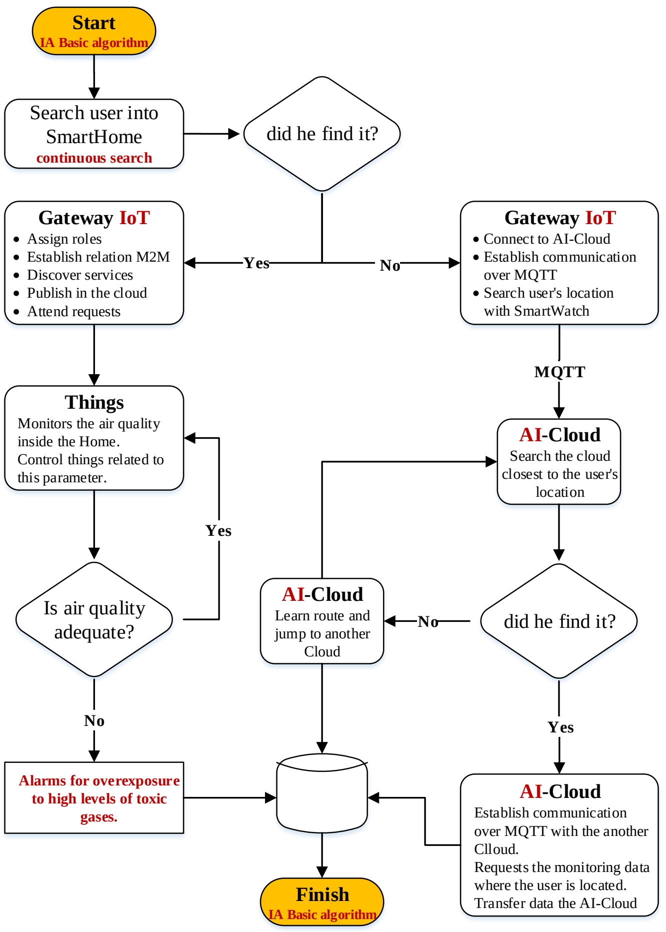

Start
IA Basic algorithm
Search user into SmartHome
continuous search
did he find it?
Yes
No
Gateway IoT
Assign roles
Establish relation M2M
Discover services
Publish in the cloud
Attend requests
Gateway IoT
Connect to AI-Cloud
Establish communication over MQTT
Search user's location with SmartWatch
MQTT
Things
Monitors the air quality inside the Home.
Control things related to this parameter.
AI-Cloud
Search the cloud closest to the user's location
Yes
Is air quality adequate?
AI-Cloud
Learn route and jump to another Cloud
No
did he find it?
No
Yes
Alarms for overexposure to high levels of toxic gases.
AI-Cloud
Establish communication over MQTT with the another Clloud.
Requests the monitoring data where the user is located.
Transfer data the AI-Cloud
Finish
IA Basic algorithm

Figure 72. Flowchart of the Gateway and the Cloud

The operating philosophy of this protocol is based on two main functions; Monitoring of Things and the exchange of "requests" through different types of messages. The requests or requests are processed by the AI depending on the type of relationship that is established (example: M2M). That is, it is not just a remote control over Things; if not that the things decide from the received request, yes it activates or not the function that is required. In the interconnection of clouds, the protocol allows an intelligent communication between each IoT network, if each central node of the network (Gateway IoT) allows to create an MQTT socket and connect to the intelligent manager in the cloud (AI-Cloud) through Internet.

5.1.4. Performance test

The tests were carried out with the objective of establishing the amount of gas disseminated in relation to the distance that the sensors could detect. Then, the system through AI would decide if with this amount of gas, it was necessary to activate the ventilation and closing systems of the gas valves. In this way, the minimum parameters were evaluated in case of an event, to keep the user at home in a controlled and safe environment for their health. However, all the CO_2 values present in the home environment (harmful or not), are stored in the database of the cloud platform, with the aim of carrying an accumulated data of exposure time versus the quantity of gases to which a person could become subject for several years. But, these tests only show data taken at very short time intervals during a day.

5.1.5. Environment sensors

It is used three Nordic Thingy: 52 boards [101] inside the house, connected via Bluetooth 5 low-energy wireless technology (BLE 5.0) in a mesh network. Each module contains an environmental sensor that can measure temperature, humidity, air pressure and air quality (CO_2 and VOC). In the streets of the city, a project [102] is being implemented with SmartHuman sensors with LoRa technology [103], which can measure VOC, CO_2, temperature and humidity. It requires connection to a LoRa network, either with its own Gateway or a contracted service.

In different work sites, low-cost alternatives can be used with open source platforms, implementing CO_2 sensors with Arduino boards [104] and SmartHuman LoRa sensors.

5.1.6. Testbed

To test the need to have networks interconnected through the cloud, a pilot test was carried out with a Nordic Thingy: 52 board, carrying it through the three environments by the daily route of a user (home, city and work site). The experiment was carried out without ventilation, to obtain the maximum values of gas saturation. In the smart house intentionally blocked the requests for ventilation by saturation of toxic gases in the AI system. In the route through the city the vehicle had the windows closed.

The experiment begins in the smart home from 8:45 a.m. until 5:16 p.m. at the user's workplace. During the journey through the streets of the city, the route is made through the main avenues of high traffic.

The first data are collected inside the house in three different areas in the following order: main room, living room, laundry room and kitchen (when it is cooking). When leaving the house, the sensor records the data inside a vehicle until it reaches the user's workplace.

In the Figure 73 it is observed, the high levels of CO_2 inside the house in the kitchen, in the city the data show high values of CO_2 and VOC in the first main avenue of high traffic. In the third part of the route through the streets of the city, it shows average and constant values of CO_2, when arriving at the destination it decreases the CO_2 values similar to those found inside the house when it is cooking. In the Figure 74, is can see the exposure time in minutes to CO_2 and VOC gases, during the route. For example, for values between 2158-2509 ppm of CO_2 and 267-321 ppb, there was an approximate time of 10 minutes of exposure. These are the highest values of the Figure 74, and its location was on a high traffic road in the city. If the user would have been longer, the amount of inhaled gases would be even greater.

According to the world health organization [105] and the world system for measuring global warming [106], they indicate that the average values recorded since 2017 in the atmosphere of the planet are 400 ppm. The above indicates that, with the data shown, this proposal could be of great help to know the amount of toxic gas in which a person is exposed on a daily day in their life.

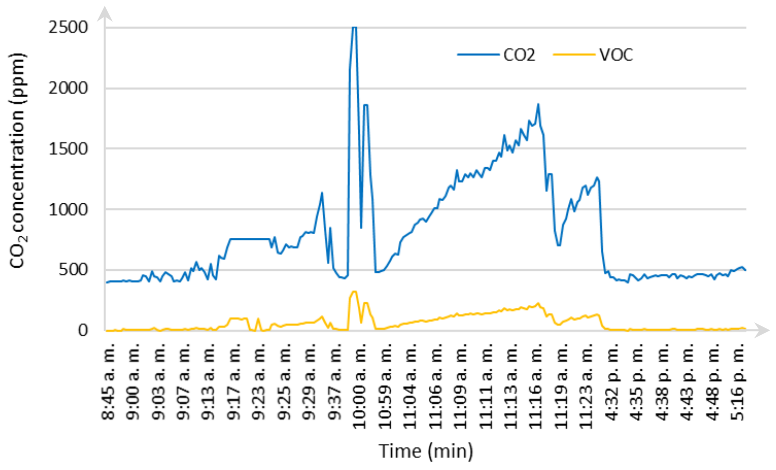


Figure 73. CO2 and VOC values in one day

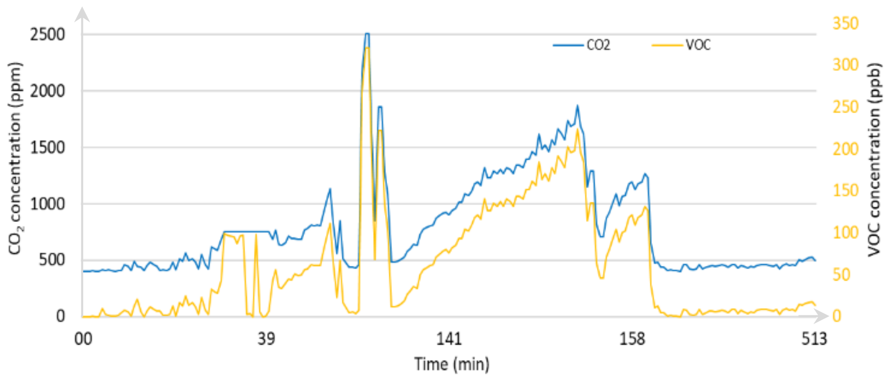


Figure 74. Time concentration profiles in CO2 and VOC measurements with device Nordic Thingy: 52

5.2. IoT-WLAN proximity network for Potentiostats

Potentiostats are devices used to make an electrochemical analysis of different substances and determine its composition. Nevertheless, it requires continuous monitoring and data collection to control the reference voltage and observe the changes. Some of the current systems store the different measurements represented in voltammograms that have already been characterized. However, these results cannot be compared in real-time with another device measuring along at the same time. Although the potentiostats are smaller and more portable, they have appeared recently and are not yet networked. Nevertheless, potentiostats are increasingly been used as biosensors, though it is still not possible to have several measurements at the same time in different locations. Therefore, it is not possible to correlate the distance between them and the external factors that may affect the sensor measurements.

Another problem is that most sensors only send information of its measurements, although the potentiostat requires sending and receiving information to adjust its operation. That is, some potentiostats developments operate in client-server configuration but not in a network, which makes automatic synchronization difficult between several clients. Moreover, these configurations do not offer long-distance communications with the device that processes the information.

5.2.1. Proposal

Taking advantage of IoT technologies and protocols, we propose a wireless local area network for potentiostats (IoT-WLAN) that will be monitored through an App. This potentiostat network design follows a proposed architecture of the Interconnection of an IoT smart network of proximity with centralized management.

To explain this proposal, we will start presenting the mechanics of the interactions between the defined nodes through Figure 75 and explain the architecture on which the network was designed. Finally, we will present a mobile application. The network measures different substances of an area, identifies them, and reports them to the broker. Whenever the system needs to know whether the same substance is present in all the areas or its distribution; the sensor takes data more precisely and synchronizes it one by one with the other sensors.

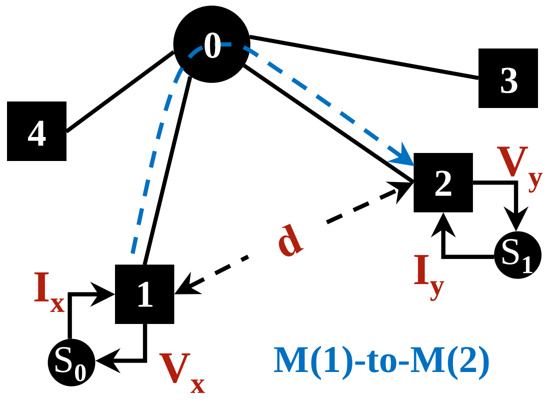


Figure 75. M2M protocol over Potentiostats Network

The Broker M(0) can be an IoT-Gateway or a mobile device, depending on the topology and network configuration. Nodes M(1, 2, 3, 4) of Figure 75 represent wireless transmission modules connected to a sensor (Sn). In Figure 75, the sensors S (0, 1, 2, ..., n) are potentiostats (e.g., S0 and S1 concerning its current (I) and voltage (V) parameters). In this example, M(1) measures the current of S0, because these values are a function of the voltage V(t). This voltage is a reference value that controls the measurements through a V(t) ramp function signal. That is, M(1) injects V(t) into S0 to obtain the values of (I).

Through an M2M protocol, a sensor can establish a relation with another sensor to synchronize the same measurement on both sensors. If M(1) obtains measurements in expected or minimum acceptable ranges with Vx, then it is possible to change the values of other sensors making Vx = Vy. This way, if Ix ≠ Iy we can conclude that some additional element is interfering or altering the measurement. Therefore, Sn{I, V} will be the set of measurements of node M. However, if more parameters are obtained from Sn, then Sn{I, V, P, T, H, L}.

Let S_i and S_j be the set of measurements of two nodes so that S_i and S_j are the measurements of all possible Sn sensor that meets the condition of the precision factor (f_p).

Then (r) in Eq. 41 represents the set of all M2M relations established by the Broker in an ordered pair (Si, Sj).

$$\mathrm{r}(S_i, S_j)\ \forall_{i,j} \mid i,j \geq 0 \wedge i,j < n\ \wedge\ f_p(S_n) \geq 90\% \qquad \text{Eq. 41}$$

Let M be a set of sensor-nodes and R the set of all relations established by Broker M(0). The sensor-node is the union of two disjoint sets between the node in the Things layer and the sensor in the Sensors layer.

$$M(n) = M(n) + Sn\,\{parameters\} \qquad \text{Eq. 42}$$

Then the relation between M(i)-to-M(j) will be shown as in Eq. 43.

$$R\big(M(i), M(j)\big) \qquad \text{Eq. 43}$$

This proposal is an application of a network model based on workgroup theory [45]. In this case, each Sn is a directed graph that maintains a reference to its parent node M(n), according to the graph theory [107] and data structure [108], merging both architecture layers into one.

5.2.2. Implemented this architecture

This article presents the proposed architecture in [43] and [52], where the IoT network is organized in five layers called: Internet, Management, Assistants, Things, and Sensors in an ascending hierarchy and tree structure. The network model of this architecture allows grouping things based on M2M protocols, at the level of the management layer and by groups of parameters in the Internet layer.

Moreover, this architecture is flexible and was designed to allow interaction with artificial intelligence (AI) algorithms. The broker device is the node of the management layer that hosts the AI and manages Internet access.

In this network design, the client nodes M(n) are the transmitting devices in the Things layer of the architecture, while the broker node M(0) is in the Management layer. The potentiostats (Sn) are located in the Sensors layer, and its parent node, the wireless modules M(n) are in the Things layer. In any network configuration, any device can assume the role of the broker (a smartphone or a PC or an IoT-Gateway). It is important to state that the connection is performed in a star topology and routed through the device of layer 3. Therefore, the ideal way to fulfill this condition is to do it on an IoT-Gateway device located on layer 3 of this architecture.

In this case, the supervision and control of the network are done locally. However, it is possible to do it from the Internet by adding an IoT platform to the Internet layer.

5.2.3. Mobile Application (App)

This application was developed in Android Studio so that it can be installed on an android mobile device (smartphone or tablet). The app has two functions. Its main function is to monitor the footprints based on voltammograms sent from the sensors to the broker and visualize them. Its second function is to act as a backup whenever the broker in layer 3, running in the background is not available. Although the M2M protocol passes through the broker with the collected data from a client node to another client node, the algorithm checks the payload every time it passes through. With this information, the modified algorithm of the broker intervenes and decides the most appropriate network sensor.

Once selected, it is matched with the source sensor and synchronized with the same parameters (Vx=Vy and Ix=Iy).

The flowchart in Figure 76 shows how the sensors are synchronized to assess whether the measurements are the same or not. Each time a sensor finds a measurement that corresponds to the previous characterization of some substances or materials, it sends the information of its Voltage parameters to the entire network. The broker compares the voltammograms of the measurements of each sensor and evaluates its precision percentage (fp). If any of these percentages is fp $\geq$ 90%, it saves the location of the sensor and evaluates other variables. This way, the algorithm can establish the distance (d) between the nodes and create a map, which could be a contamination map if that was the case.

The algorithms developed for the Broker App or IoT-Gateway (Broker) [45], are based on Eq. 41, Eq. 42, and Eq. 43. The algorithm comparing the previously stored typical voltammograms with those obtained from the potentiostats has been developed with Eq. 41. This comparator is shown in the flowchart and evaluates the factor of precision of each sensor-node as fp(Sn). The algorithm responsible for collecting the parameters sent from each sensor-node was based on Eq. 42. This algorithm reviews the payloads of the protocol and identifies its source address. The Broker's pairing algorithm based on Eq. 43 establishes machine-to-machine relations for sensor pairs to synchronize. Once the acceptable fp values of each sensor are established, this algorithm performs synchronization by sending the parameters (V and I) to the selected destination sensor.

The Broker App has a user interface from which the different voltammograms are been, monitor. Each voltammogram is built with (V) and (I) parameters using a cyclic voltammetry technique [109]. That is, the node applies a ramp voltage signal V(t) between two voltage values to the sensor and reads its current (I) answer. Therefore, the voltammogram will be the footprint of the substance or material that has been analyzed. The sensor-node sends this information through the network on an M2M protocol for the App Broker to process it.

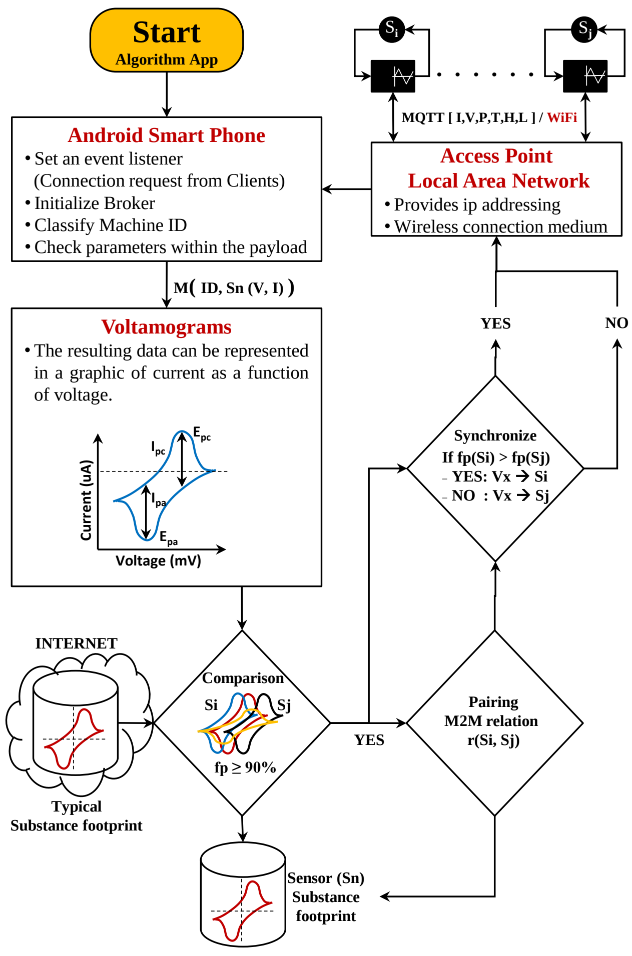


Figure 76. Flowchart of the App

The typical substance footprint (TSF) is stored in the cloud and the Broker App or IoT-Gateway, and the query via the Internet whenever it needs to compare it with the one obtained by a sensor. The footprint is also temporarily stored in the internal memory of the wireless module (node).

5.2.4. Performance test

In this section, the tests of the proposed system are explained. These tests seek to demonstrate that the broker algorithm can create new M2M relations so that the sensors synchronize with the most precise measurement given by another sensor. Following the relational condition explained in Figure 75 and the flowchart of Figure 76, an IoT-WLAN network was implemented. This network has three parts. The first part is the set of hardware devices used on the implementation. The second one is the design of the wireless network on Wi-Fi technology in a star topology and with MQTT protocol. The third one is the resulting analysis of the operating network.

Although the architecture supports any M2M protocol to perform these tests, MQTT was selected because of its adaptability and extensive documentation.

5.2.5. Implementation of the devices

The development of this proposal involves the use of ESP8266-01 board [110] with a Programmable System-on-Chip (PSoC), integrated with a microcontroller and a Wi-Fi network module. The transmitter module of the Things layer sends the parameters (V) and (I) from the sensor to the broker. These parameters are obtained from the LMP91000 potentiostat [111] and its 3-Lead Electrochemical Cell. The rest of the parameters (P, T, H, L) were added to the test and demonstrated the versatility and scalability of the system. The program in the ESP8266-01 microcontroller provides a ramp signal in voltage to the potentiostat while recording the currents produced during the reaction. The resulting data is sent to the mobile application and is represented by a plot current vs. voltage called voltammogram.

The ADS1115 module is an ADC with four channels of 15-bits + 1-bit of precision. Therefore, each port can be used to collect more parameters. The port A0 of the ADC measures (I) on the potentiostat and the module MCP4725 12-Bits DAC applies V(t). The modules BME280 measure (P, T, H), and the Ultimate GPS Breakout v3 measures (L). Figure 77 shows the sensor node circuit and the set of parameters of each module as a single sensor.

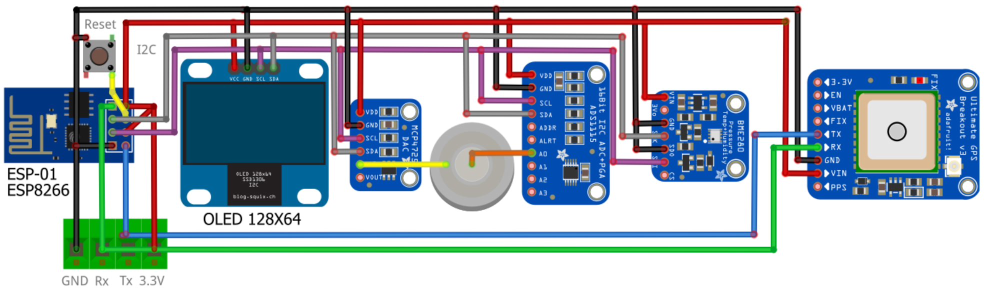


Figure 77. Sensor-Node circuit (M(1) + S_0)

The Wi-Fi transmission module is the node M(n), and the potentiostat and the other measuring circuits are the sensors Sn{I, V, P, T, H, L}. Both sets are a sensor-node. The entire circuit is shown in Figure 77 and was performed with the fritzing tool [112].

The ESP8266 was programmed to collect all sensor information using the Inter-Integrated Circuit (I2C) protocol. Then the ESP8266 puts this data on a PUBLISH message of the MQTT-Client protocol and sends it to the network with the Broker's destination address.

5.2.6. Testbed

The technical implementation of this proposal was possible through the mentioned modules. The target was using a low cost, fast and easy implementation to focus the tests on the communication system.

The network design was adapted to this architecture to convert a conventional network into an IoT network. The idea with this architecture is to take advantage of the access points of a home network and reuse it. Therefore, Figure 78 presents an IoT network design using a conventional Wi-Fi Access Point (AP) and a mobile device such as a Broker.

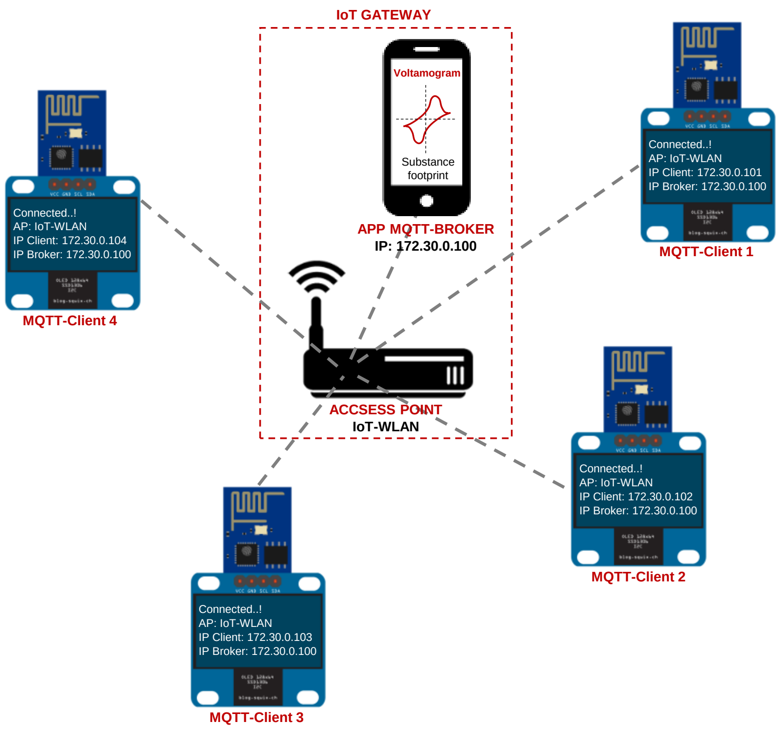


Figure 78. IoT-WLAN proximity network design

For the MQTT-Broker protocol to work on the Mobile App, it was necessary to install Mosquitto on the smartphone [113]. The Arduino code of the ESP8266 uses a library called <PubSubClient.h> to implement the MQTT-Client protocol. The Arduino code of the ESP8266 uses a library called <PubSubClient.h> to implement the MQTT-Client protocol. In both cases, the Broker and the Client are connected to the network through the AP with Service Set Identifier (SSID: IoT-WLAN). MQTT is an open-source M2M protocol [79], and therefore, it is possible to modify its operation mechanics. This advantage allows new proposals by modifying the already developed code that has been shared through communities of programmers.

The tests were performed following the design of Figure 78. Additionally, a personal computer (PC) connected by Ethernet cable to the AP was included. On the PC, the Wireshark tool was used to capture the sent data from the sensor nodes to the Broker.

The captured data is used to analyze the Broker's performance after modifying the protocol algorithm of its usual operation. That is, the speed of the data may be affected by the intervention of the Broker in the processing of MQTT payloads. Therefore, the following measures were made: first with standard MQTT packages to have a reference and observe the system changes, and then, the MQTT packets were measured with the algorithm proposed one-to-one to know how the transmission speed was affected when the number of sensor-nodes increased.

Figure 79 represents a reference information flow of a M2M synchronization. The test was performed by initially transmitting a standard MQTT establishment process, including TCP sessions of connection initiation and termination (filter: tcp.stream eq 0). The transmission was made from the source address 172.30.0.101 with the messages Connect, Subscribe, Publish, Unsubscribe, and Disconnect with destination 172.30.0.102. Where the Connect Message is IP-Broker: 172.30.0.100, the Subscribe Message is Topic:/S1, and Publish Message is Topic:/S1, Payload: V, I. Figure 79 shows the number of packets per second TCP between a M2M synchronization after the broker pairs it. The filter used in the test to capture the packets was as follows: (ip.addr eq 172.30.0.101 and ip.addr eq 172.30.0.102) and (tcp.port eq 52620 and tcp.port eq 1883). The maximum peak value was 83 Bytes, while the minimum value was 54 Bytes, which belongs to an Unsubscribe Message. The average number of Bytes transferred was 63 Bytes.

In Figure 80, the broker intervention process is more evident. The values sent of V and I are longer data strings. This Figure shows the relation between the numbers of packets transmitted per second. The average duration of each packet sent through the network was 77s.

In Figure 81, the maximum values are peaks of 1392 Bytes, while the minimum values were 42 Bytes and the average values were 266 Bytes.

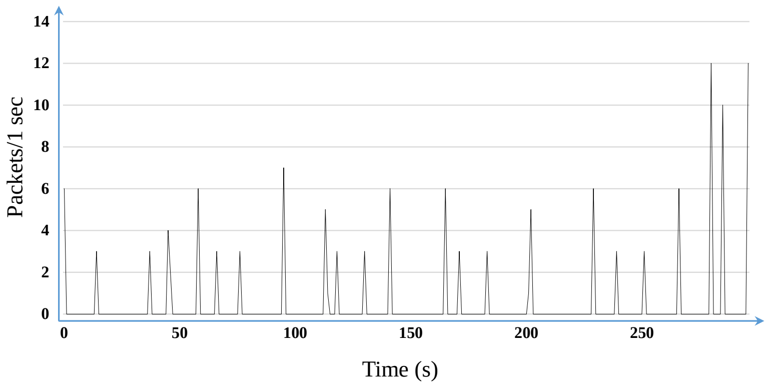


Figure 79. Machine-to-Machine synchronization M(S0)-M(S1)

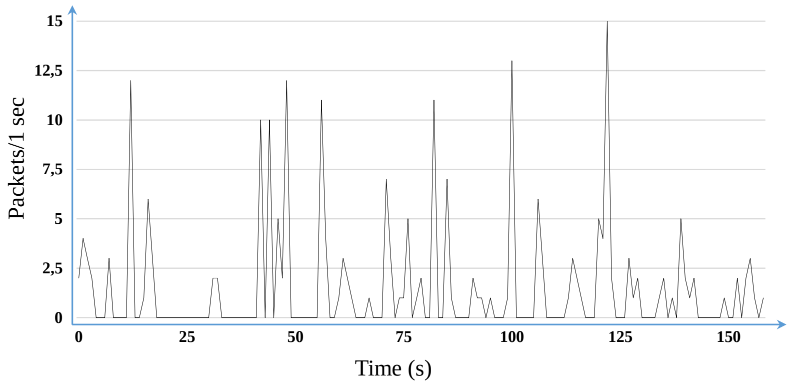


Figure 80. Datastream M(S0)-M(S1) with values payload of V and I

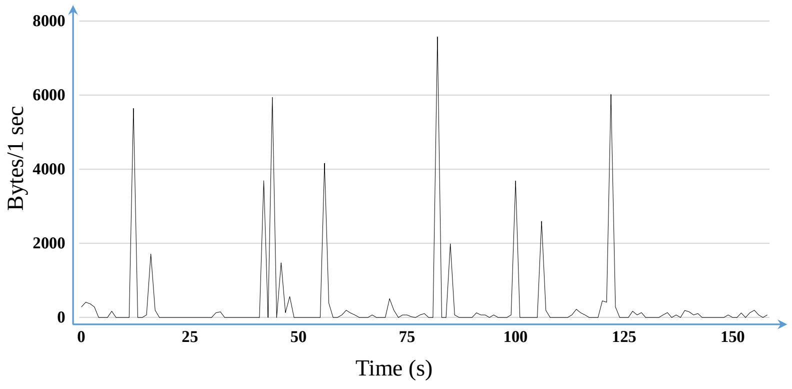


Figure 81. Pair M(S0)-M(S1)

5.3. Conclusion

The data collected during this route, showed that this type of connection architectures in the cloud is necessary to maintain an updated information system with accurate data. By having a platform with intelligent management and a mobile application in a smart watch, one could cross the information of the user's vital signs at the same moment of time and location with the data of environmental contamination to which a person is exposed.

The importance of having installed air quality IoT networks constantly monitoring the contamination of different environments, allow to have more stable and reliable values. Although the tests performed were made with a sensor in motion, these data allow us to see the need to monitor these environmental variables and extend this initiative in all cities with pollution problems.

The integration of different technologies, protocols and architectures through a centralized management in a Gateway on a level of the Platforms in the cloud, a complex of tools to manage, due to the large amount of information that all things, for What is evident is the use of decision methods through artificial intelligence (AI) algorithms.

Most things are programmable and have the ability to connect to the Internet, which means the possibility of integrating intelligent artificial intelligence systems that make users, this facilitates communication between things, Gateway and Platform are even more efficient, because they would all be controlled by the AI. The AI can use the main Internet connection channel, if it first evaluates and decides that Things (M2M) can solve a problem locally.

Currently, there are some experimental implementations of potentiostats on small boards that are very efficient in comparison to the traditional desktop stations used in laboratories. However, its connectivity is limited to the monitoring of information. With this contribution, the possibility of using the IoT concept to configure proximity wireless networks and transmit its information in a collaborative environment is feasible.

The architecture on which the network was designed allows flexibility in adapting existing networks to the IoT. Therefore, it can be used in any application such as the electrochemical measurement and analysis system (potentiostats) of this work.

The algorithms of the App Broker, built from the deduced equations from the architecture, improve the information processing. As a result, the tests showed that the MQTT packages are not affected in its speed due to the intervention of the Broker in the payloads.

In the network tests a conventional Wi-Fi AP was used and, although outdoor distance reaches an approximate coverage of 100 to 150 meters, it is possible to use GPS as an optional parameter (L). There are many applications and where this solution may be necessary such as, for example, whenever the coverage is extended with repeaters or a low-power wide-area network (LPWAN) is used.

We expect to do the same experiment with more client sessions (at least 10 to 15 clients) to see the performance. In the same way, we would like to try other M2M protocols such as CoAP and HTTP Restful. Moreover, it would be interesting to implement the Broker on an ESP8266 or replace the AP with an IoT-Gateway (Broker) based on a single-board computer (SBC) such as the Raspberry Pi 3 Model B + (RPi3).

In future works, we will perform more tests that measure the bandwidth based on the size of the parameters, including not only transmitted parameters such as (V, I), but also the rest of parameters (P, T, H, L).

Chapter 6. Tests on real implementations

6.1. Introduction

This chapter will see two ways to obtain the data: the first, through Cisco Packet Tracer using an external connection to the cloud, and the second on real devices. In both cases, Wireshark was used to capture the packets of the MQTT and REST protocols. As explained in previous chapters, achieving a complete scenario in real life is very difficult because there is no interoperability between different manufacturers, nor are they reprogrammable. Therefore, it was necessary to acquire several appliances with these characteristics to operate under the proposed architecture to carry out these tests, but it was not possible for the above reasons. However, experimental prototypes with SBC devices (RPi3 in household appliances and RPi4 in IoT-Gateway) and MCU (ESP8266 in sensors and actuators and some electrical and security objects) were implemented. In addition, to complete the entire implementation scenario, it was necessary to support some tests using the simulator in mixed configuration with external connection and complement the tests together with the real devices. From the simulator, the deployment of the service was for monitoring using ThingSpeak Cloud. Likewise, the data extracted from the objects were stored in the IoT-Gateway using MySQL Workbench with a JDBC connection between the Python application and the relational SQL tables. Then this same service was deployed in the cloud using Oracle Cloud; however, although they were deployed in different clouds, one for monitoring and the other for storage, both were connected to the same application. We will see both implementations separately below.

6.2. IoT-HOME App

It is a desktop application developed for Cisco Packet Tracer (PT) in Python and Javascript and divided into several stages. However, to create a design based on this architecture, the concepts of IoT-Gateway, IoT-Protocols, IoT-Thing, IoT-Network were defined first, as shown in Figure 82.

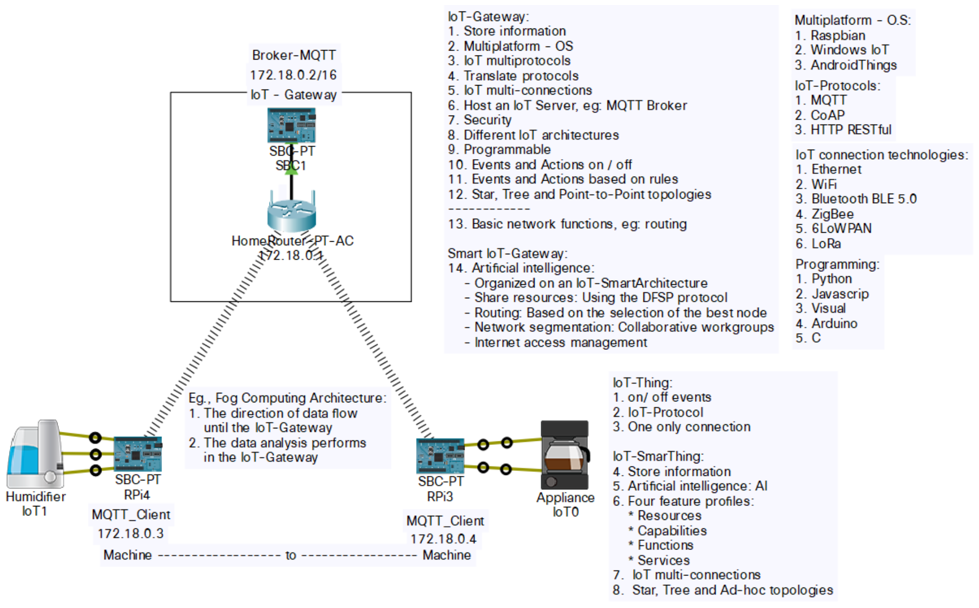

Broker-MQTT
172.18.0.2/16
IoT - Gateway
SBC-PT
SBC1
HomeRouter-PT-AC
172.18.0.1
IoT-Gateway:
1. Store information
2. Multiplatform - OS
3. IoT multiprotocols
4. Translate protocols
5. IoT multi-connections
6. Host an IoT Server, eg: MQTT Broker
7. Security
8. Different IoT architectures
9. Programmable
10. Events and Actions on / off
11. Events and Actions based on rules
12. Star, Tree and Point-to-Point topologies
----
13. Basic network functions, eg: routing
Smart IoT-Gateway:
14. Artificial intelligence:
- Organized on an IoT-SmartArchitecture
- Share resources: Using the DFSP protocol
- Routing: Based on the selection of the best node
- Network segmentation: Collaborative workgroups
- Internet access management
Multiplatform - O.S:
1. Raspbian
2. Windows IoT
3. AndroidThings
IoT-Protocols:
1. MQTT
2. CoAP
3. HTTP RESTful
IoT connection technologies:
1. Ethernet
2. WiFi
3. Bluetooth BLE 5.0
4. ZigBee
5. 6LoWPAN
6. LoRa
Programming:
1. Python
2. Javascrip
3. Visual
4. Arduino
5. C
Eg., Fog Computing Architecture:
1. The direction of data flow until the IoT-Gateway
2. The data analysis performs in the IoT-Gateway
Humidifier
IoT1
SBC-PT
RPi4
MQTT_Client
172.18.0.3
SBC-PT
RPi3
MQTT_Client
172.18.0.4
Appliance
IoT0
Machine ---------------- to ---------------- Machine
IoT-Thing:
1. on/ off events
2. IoT-Protocol
3. One only connection
IoT-SmarThing:
4. Store information
5. Artificial intelligence: AI
6. Four feature profiles:
* Resources
* Capabilities
* Functions
* Services
7. IoT multi-connections
8. Star, Tree and Ad-hoc topologies

Figure 82. Design parameters of an IoT-Network on IoT-Smart Architecture using PT

6.2.1. M2M relation between sensor-actuator based on MQTT payload

Initially, separate tests are performed based on each architecture; in the case of Figure 83, a Fog Computing architecture with MQTT protocol is presented. This simple system seeks to initially test the connection and the sending of data through the protocol. The exercise results are simple but allow an M2M relationship between a sensor and an actuator, each of which will subsequently belong to layers 1 and 2 of this architecture. In this scenario, data is sent to the IoT-Gateway, which previously contains a table of the objects subscribed to the network. With this information, the Broker hosted in the IoT-Gateway routes the MQTT packets to the destination object using a “topic”. In the case of the test simulation in Figure 83, the actuator (led) turns on when it receives a temperature greater than 22 degrees Celsius from the temperature sensor measurements. In the second scenario in the same figure, the same principle is used to turn on a lamp if the motion sensor detects movement. In both cases, it uses the same principle and can be used for any other implementation of the rule-based sensor-actuator type.

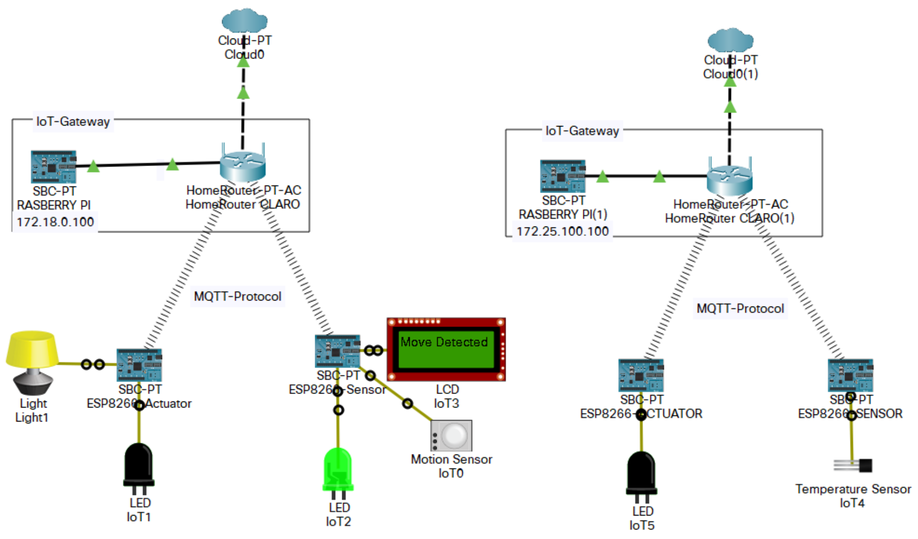
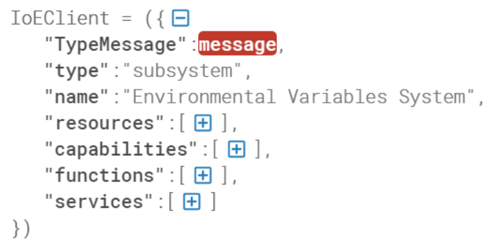


Figure 83. Examples of MQTT data transport between sensor and actuator

6.2.2. M2M relation between sensor-actuator using JSON into MQTT

The architecture shown in the previous Figure 82 presents an M2M connection in Fog Computing architecture. This configuration can be applied to any object in the network regardless of the classification assigned by the AI. Therefore, in the initial tests for data extraction, it is not necessary to specify the layer, level and workgroup. Each object connected to the network through the IoT-Gateway has been made using WiFi technology. On this technology, the TCP transport protocol is used over IP addresses, and on this, the data is sent using the MQTT protocol. Once communication is established, each object sends the IoT-Gateway information about its technical specifications segmented into Resources, Capabilities, Functions and Services. Initially, an object by factory configuration only contains technical specifications, which then is structured by an internal algorithm (AI of the object) that sends them using the DFSP protocol developed in this proposal. In other words, the complete transmission in the system is of the form DFSP / MQTT / TCP / IP / WiFi. As already explained in previous sessions, the data is transmitted using a JSON structure through DFSP messages within the MQTT payload. That is, with the following syntax everything is encapsulated within the payload, [Json {"TypeMessage": message, "data": data}]. The following Figure 84 shows an example of a DSFP login or initialization message.



```
IoEClient = ({⊟
    "TypeMessage":message,
    "type":"subsystem",
    "name":"Environmental Variables System",
    "resources":[ ⊞ ],
    "capabilities":[ ⊞ ],
    "functions":[ ⊞ ],
    "services":[ ⊞ ]
})
```

Figure 84. Example of a DFSP initialization message

Figure 85 shows the network design using PT to extract data from the objects in two ways. The first one extracts data from each sensor and publishes it to the ThingSpeak Cloud. The second uses a single card to collect the data extracted from the sensors and turn it into a single object called a subsystem. In this configuration, based on the proposed architecture, a standalone system is maintained each board is in separate layers, layer 1 (sensors) and layer 2 (mainboard), respectively, but working together. In this way, it is observed that with sensors separate the data arrive at different times to the cloud, while in the second part, the data is collected by layer 2 (mainboard) and sent in a single JSON at the same time. In addition, under this configuration, the layer 2 board can have more processing capacity, allowing hosting an AI for data pre-processing.

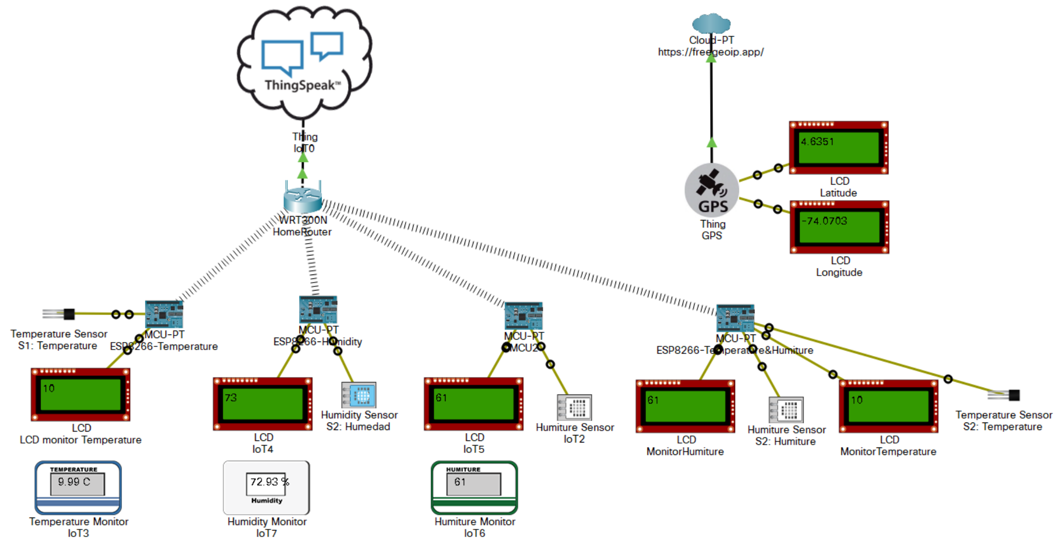
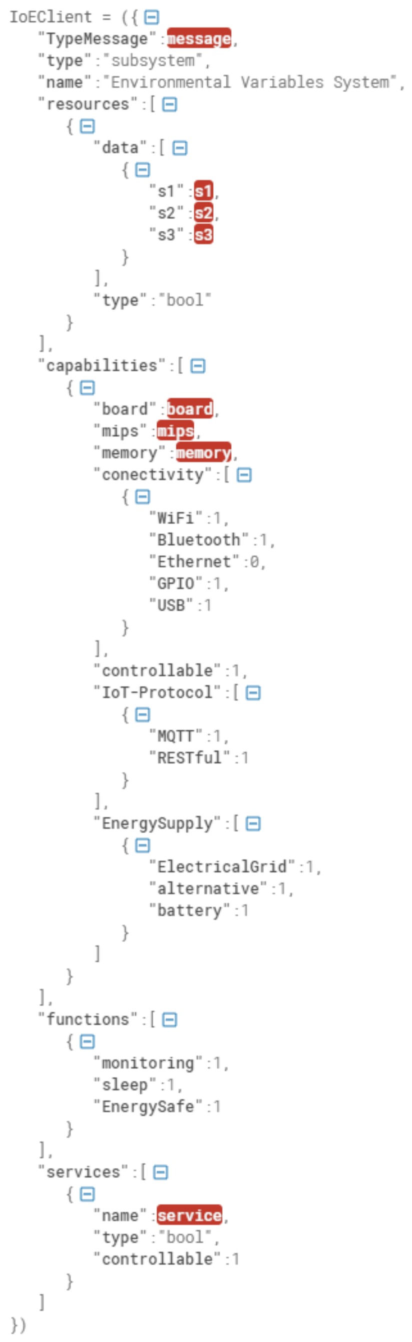
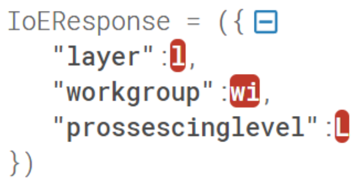


Figure 85. Monitoring data using JSON and published with RESTful on ThingSpeak Cloud



```
IoEClient = ({
   "TypeMessage":message,
   "type":"subsystem",
   "name":"Environmental Variables System",
   "resources":[
      {
         "data":[
            {
               "s1":s1,
               "s2":s2,
               "s3":s3
            }
         ],
         "type":"bool"
      }
   ],
   "capabilities":[
      {
         "board":board,
         "mips":mips,
         "memory":memory,
         "conectivity":[
            {
               "WiFi":1,
               "Bluetooth":1,
               "Ethernet":0,
               "GPIO":1,
               "USB":1
            }
         ],
         "controllable":1,
         "IoT-Protocol":[
            {
               "MQTT":1,
               "RESTful":1
            }
         ],
         "EnergySupply":[
            {
               "ElectricalGrid":1,
               "alternative":1,
               "battery":1
            }
         ]
      }
   ],
   "functions":[
      {
         "monitoring":1,
         "sleep":1,
         "EnergySafe":1
      }
   ],
   "services":[
      {
         "name":service,
         "type":"bool",
         "controllable":1
      }
   ]
})
```

Figure 86. Example of structured data on JSON extracted from each connected object



```
IoEResponse = ({
    "layer":l,
    "workgroup":wi,
    "prossescinglevel":L
})
```

Figure 87. IoT-Gateway AI response

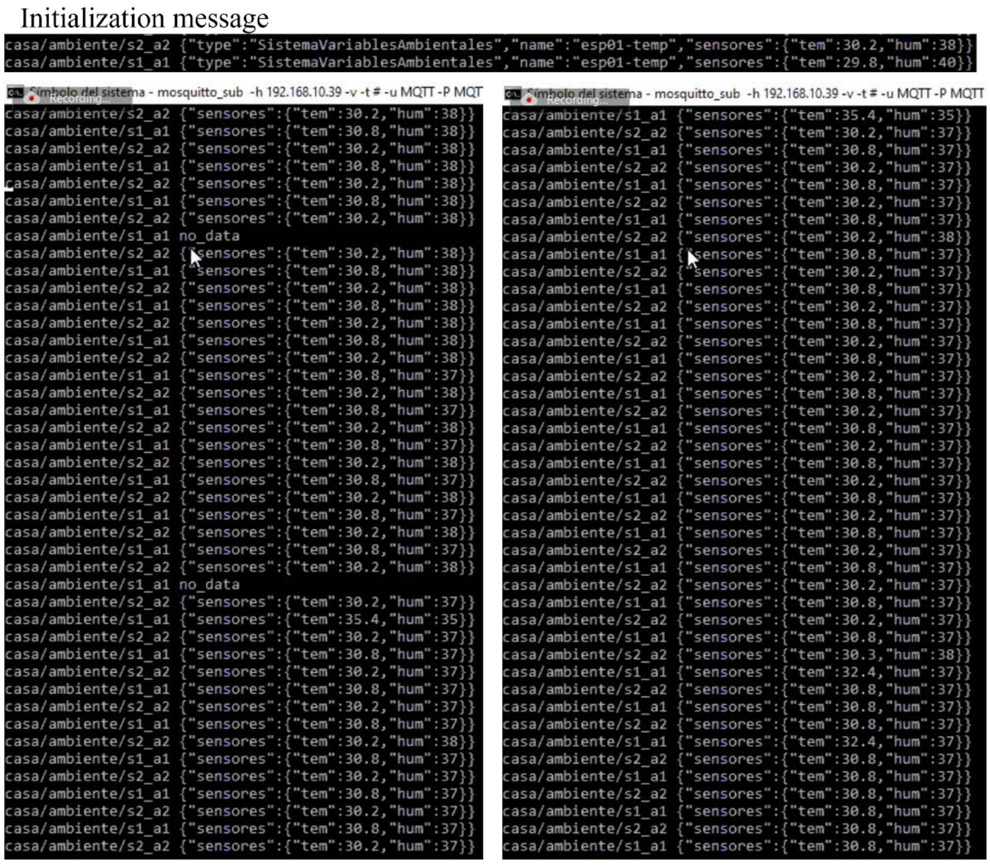
Once the process of discovering or announcing the features of the new object in the network reaches the Gateway, the AI classifies it and sends a response message, see Figure 87. The following DFSP messages will be sent with the information obtained in the response message as part of the topic into the publish message. The topic will contain the workgroup, the layer, the level of processing, and the name of the object. It will only update the information required by the AI using specific DFSP messages such as capabilities, resources, services, and functions within the payload. The following example reflects the syntax, publish (topic, payload) like, publish (w1 / layer1 / level0 / name, capabilitiesMessage).

6.3. Real devices

This session shows some devices based on the architecture explained in the PT simulation of Figure 83. In this case, the sensors are implemented on the ESP8266 devices in Fog Computing architecture using the local router connected to the RPi3 as MQTT Broker. The transmitted data corresponding to the temperature and humidity values using a JSON structure on the MQTT payload. The test consists of capturing with Wireshark the data that is transmitted between two objects. Each object initially sends a JSON structure with the object information in the order previously requested by the AI of the IoT-Gateway. This information is organized as shown in Figure 84 and displayed as shown in Figure 86. After this, updates are sent every time there is a temperature change. An RPi connected to a router is used in this architecture, and this set represents an IoT-Gateway. The RPi provides the MQTT broker, in this case, Mosquitto, and the router provides the routing in the network. The publications that arrive from each object in the network can be viewed and monitored in the Mosquitto Broker. Figure 88 hows the initial transmission with the topic and the JSON payload that contains the object's information. This information is organized according to the DFSP protocol messages using JSON. This information is organized according to the DFSP protocol messages using JSON. The instruction used to view the publication on the network is: C:\Program Files\Mosquitto>mosquitto_sub -h 192.168.10.39 -v -t # -u MQTT -P MQTT.

Figure 88 shows that sometimes no data is transmitted, posting the message "no data", this means that there was some problem in the network, and the algorithm retransmits it. Every time this happens, TCP connection re-establishment packets are resent, and this causes additional delay and packet loss in that event. However, in the re-establishment process, packet snatching also occurs due to the addition of the TCP packets. They are two devices whose topic is sensor-actuator (s1_a1) and the other (s2_a2), which means that one device assumes both roles.

Figure 88. Captures M2M communication using JSON shows the M2M connection between these devices, sending a DFSP message announcing their characteristics to the IoT Gateway. Then, under this topic, only updates the data that corresponds to each sensor.


Initialization message

```
casa/ambiente/s2_a2 {"type":"SistemaVariablesAmbientales","name":"esp01-temp","sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"type":"SistemaVariablesAmbientales","name":"esp01-temp","sensores":{"tem":29.8,"hum":40}}
```

Símbolo del sistema - mosquitto_sub -h 192.168.10.39 -v -t # -u MQTT -P MQT

```
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 no_data
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":38}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 no_data
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":35.4,"hum":35}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
```

Símbolo del sistema - mosquitto_sub -h 192.168.10.39 -v -t # -u MQTT -P MQTT

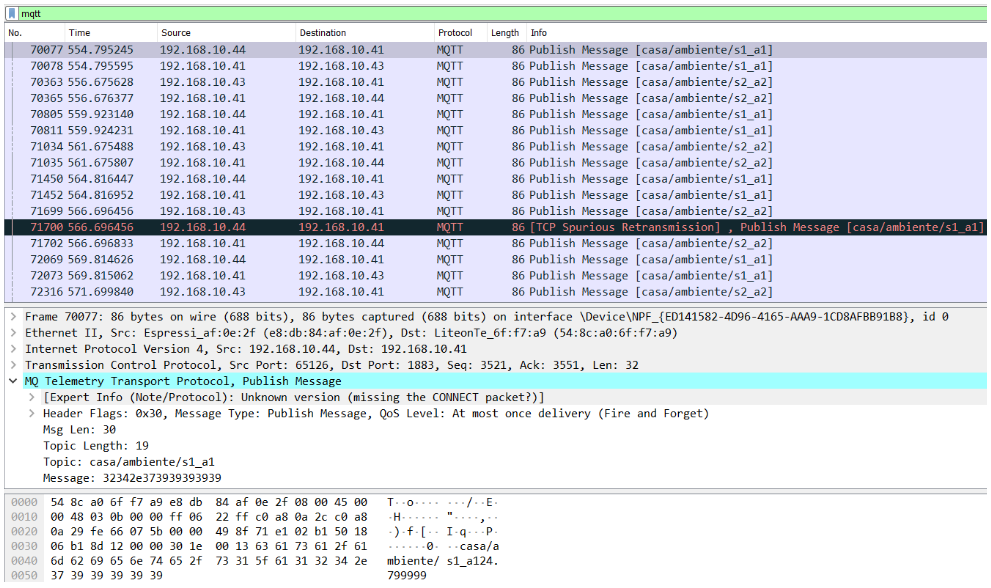
```
casa/ambiente/s1_a1 {"sensores":{"tem":35.4,"hum":35}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.2,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.3,"hum":38}}
casa/ambiente/s1_a1 {"sensores":{"tem":32.4,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":32.4,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s2_a2 {"sensores":{"tem":30.8,"hum":37}}
casa/ambiente/s1_a1 {"sensores":{"tem":30.8,"hum":37}}
```

Figure 88. Captures M2M communication using JSON

All tests are based on measuring with the same devices in different scenarios and under different conditions. The final result is the one shown in the previous Figure.

In the first scenario, the data is sent independently without JSON to an IoT-Gateway shared with network traffic.

The captures of this scenario are observed in Figure 89



```
mqtt
No.     Time        Source          Destination     Protocol  Length Info
  70077 554.795245  192.168.10.44   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  70078 554.795595  192.168.10.41   192.168.10.43   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  70363 556.675628  192.168.10.43   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  70365 556.676377  192.168.10.41   192.168.10.44   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  70805 559.923140  192.168.10.44   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  70811 559.924231  192.168.10.41   192.168.10.43   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  71034 561.675488  192.168.10.43   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  71035 561.675807  192.168.10.41   192.168.10.44   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  71450 564.816447  192.168.10.44   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  71452 564.816952  192.168.10.41   192.168.10.43   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  71699 566.696456  192.168.10.43   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  71700 566.696456  192.168.10.44   192.168.10.41   MQTT         86 [TCP Spurious Retransmission] , Publish Message [casa/ambiente/s1_a1]
  71702 566.696833  192.168.10.41   192.168.10.44   MQTT         86 Publish Message [casa/ambiente/s2_a2]
  72069 569.814626  192.168.10.44   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  72073 569.815062  192.168.10.41   192.168.10.43   MQTT         86 Publish Message [casa/ambiente/s1_a1]
  72316 571.699840  192.168.10.43   192.168.10.41   MQTT         86 Publish Message [casa/ambiente/s2_a2]

> Frame 70077: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface \Device\NPF_{ED141582-4D96-4165-AAA9-1CD8AFBB91B8}, id 0
> Ethernet II, Src: Espressi_af:0e:2f (e8:db:84:af:0e:2f), Dst: LiteonTe_6f:f7:a9 (54:8c:a0:6f:f7:a9)
> Internet Protocol Version 4, Src: 192.168.10.44, Dst: 192.168.10.41
> Transmission Control Protocol, Src Port: 65126, Dst Port: 1883, Seq: 3521, Ack: 3551, Len: 32
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 30
    Topic Length: 19
    Topic: casa/ambiente/s1_a1
    Message: 32342e373939393939

0000  54 8c a0 6f f7 a9 e8 db  84 af 0e 2f 08 00 45 00   T··o···· ···/··E·
0010  00 48 03 0b 00 00 ff 06  22 ff c0 a8 0a 2c c0 a8   ·H······ "····,··
0020  0a 29 fe 66 07 5b 00 00  49 8f 71 e1 02 b1 50 18   ·)·f·[·· I·q···P·
0030  06 b1 8d 12 00 00 30 1e  00 13 63 61 73 61 2f 61   ······0· ··casa/a
0040  6d 62 69 65 6e 74 65 2f  73 31 5f 61 31 32 34 2e   mbiente/ s1_a124.
0050  37 39 39 39 39 39                                  799999
```

Figure 89. Wireshark capture of data without JSON structure

From the above screenshot, Figure 90 shows the number of packets needed to transmit the publish message that contains the device data. From the graph it can be seen that a constant number of packets are maintained except for those necessary when the TCP connection fails and restarts transmission, which vary between 3 and 9 packets in a local network connection.

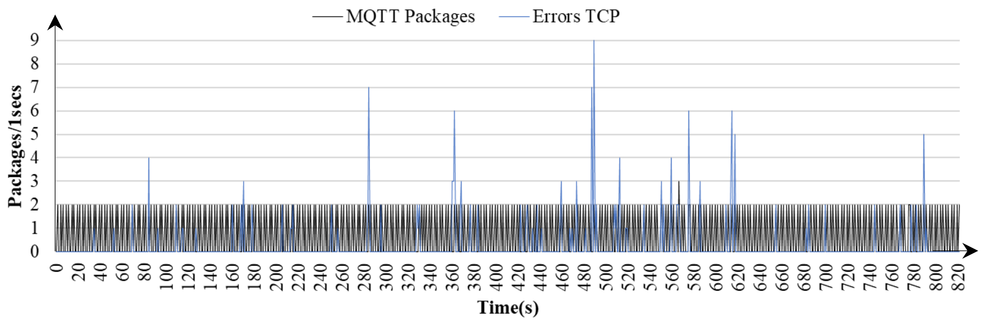


Figure 90. Number of packets transmitted in M2M connection without JSON

Figure 91 shows the Round Trip Time (RTT) of the first 250 seconds of sampling with a maximum of 250 milliseconds (ms) and an average of 130 ms, which is considered very high for a local network.

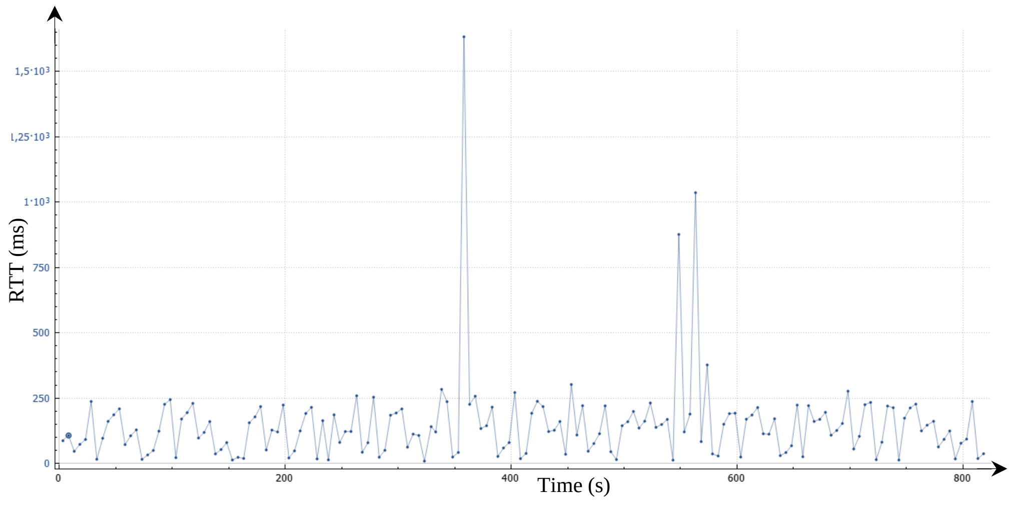
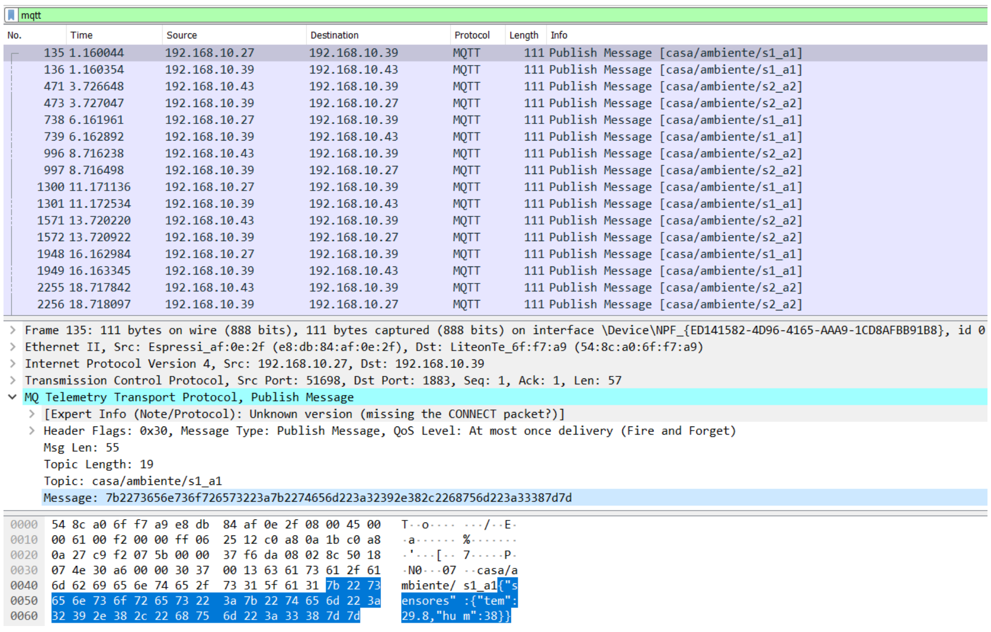


Figure 91. RTT of the M2M connection within the local network witout JSON

Figure 92 shows the capture of the data transmitted between the two devices using JSON in the data structure. The same topic is maintained with the difference that the sensors' measurements, in this case, Temperature and humidity, are sent simultaneously in a single message.



mqtt

No.	Time	Source	Destination	Protocol	Length	Info
135	1.160044	192.168.10.27	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s1_a1]
136	1.160354	192.168.10.39	192.168.10.43	MQTT	111	Publish Message [casa/ambiente/s1_a1]
471	3.726648	192.168.10.43	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s2_a2]
473	3.727047	192.168.10.39	192.168.10.27	MQTT	111	Publish Message [casa/ambiente/s2_a2]
738	6.161961	192.168.10.27	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s1_a1]
739	6.162892	192.168.10.39	192.168.10.43	MQTT	111	Publish Message [casa/ambiente/s1_a1]
996	8.716238	192.168.10.43	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s2_a2]
997	8.716498	192.168.10.39	192.168.10.27	MQTT	111	Publish Message [casa/ambiente/s2_a2]
1300	11.171136	192.168.10.27	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s1_a1]
1301	11.172534	192.168.10.39	192.168.10.43	MQTT	111	Publish Message [casa/ambiente/s1_a1]
1571	13.720220	192.168.10.43	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s2_a2]
1572	13.720922	192.168.10.39	192.168.10.27	MQTT	111	Publish Message [casa/ambiente/s2_a2]
1948	16.162984	192.168.10.27	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s1_a1]
1949	16.163345	192.168.10.39	192.168.10.43	MQTT	111	Publish Message [casa/ambiente/s1_a1]
2255	18.717842	192.168.10.43	192.168.10.39	MQTT	111	Publish Message [casa/ambiente/s2_a2]
2256	18.718097	192.168.10.39	192.168.10.27	MQTT	111	Publish Message [casa/ambiente/s2_a2]

```
> Frame 135: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface \Device\NPF_{ED141582-4D96-4165-AAA9-1CD8AFBB91B8}, id 0
> Ethernet II, Src: Espressi_af:0e:2f (e8:db:84:af:0e:2f), Dst: LiteonTe_6f:f7:a9 (54:8c:a0:6f:f7:a9)
> Internet Protocol Version 4, Src: 192.168.10.27, Dst: 192.168.10.39
> Transmission Control Protocol, Src Port: 51698, Dst Port: 1883, Seq: 1, Ack: 1, Len: 57
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 55
    Topic Length: 19
    Topic: casa/ambiente/s1_a1
    Message: 7b2273656e736f726573223a7b2274656d223a32392e382c2268756d223a33387d7d
```

```
0000  54 8c a0 6f f7 a9 e8 db  84 af 0e 2f 08 00 45 00   T··o···· ···/··E·
0010  00 61 00 f2 00 00 ff 06  25 12 c0 a8 0a 1b c0 a8   ·a······ %·······
0020  0a 27 c9 f2 07 5b 00 00  37 f6 da 08 02 8c 50 18   ·'···[·· 7·····P·
0030  07 4e 30 a6 00 00 30 37  00 13 63 61 73 61 2f 61   ·N0···07 ··casa/a
0040  6d 62 69 65 6e 74 65 2f  73 31 5f 61 31 7b 22 73   mbiente/ s1_a1{"s
0050  65 6e 73 6f 72 65 73 22  3a 7b 22 74 65 6d 22 3a   ensores" :{"tem":
0060  32 39 2e 38 2c 22 68 75  6d 22 3a 33 38 7d 7d      29.8,"hu m":38}}
```

Figure 92. Wireshark capture using JSON

Figure 93 shows the number of packages used in each publishes using a JSON data structure. It is observed that each publishes sent every 5 seconds with only one (1) packet constantly. Every 50 seconds, the system retransmits to maintain a more stable connection at the TCP level. This retransmission requires two (2) packets and can be seen in the figure with a time interval of 50 seconds. In the 800-second time sample performed in the test, they remain error-free until time interval 589. There, a TCP connection error is observed while the algorithm performs the reconnection. In this figure compared to Figure 90, good transmission stability is observed and with fewer errors.

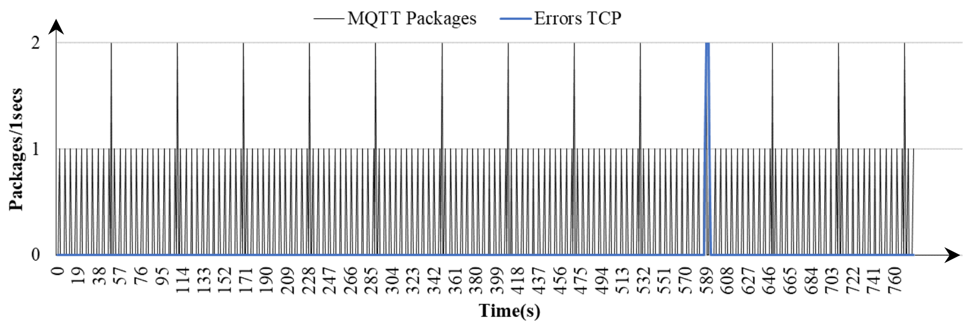


Figure 93. Number of packets transmitted in M2M connection using JSON

Figure 94 shows the RTT of the M2M connection of the two devices within the same local network or Fog Computing architecture. The figure shows that the test was taken with a duration of 800 seconds. The maximum duration of the RTT was 0.063ms, the lowest was 0.01ms, and the average was 0.0365ms. It indicates that the reduction in time concerning that observed in Figure 91 was quite significant.

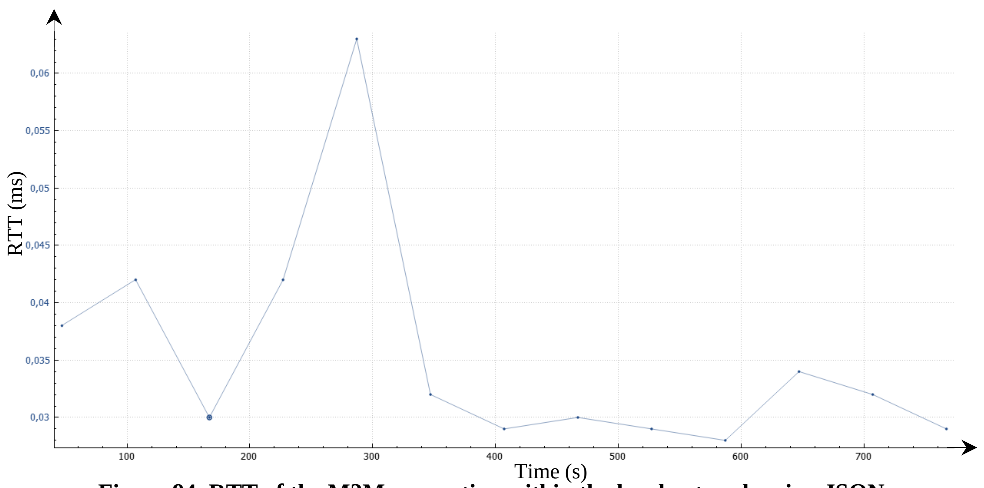
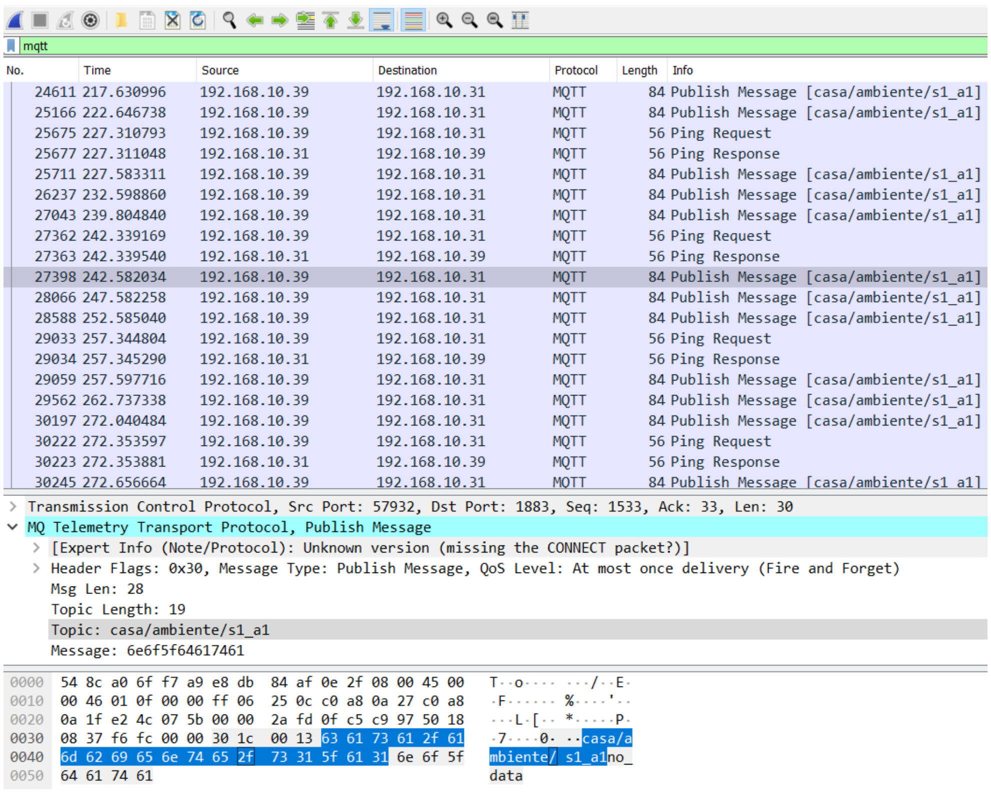


Figure 94. RTT of the M2M connection within the local network using JSON

A similar test is performed with the same JSON packaging algorithm but connected to the Thingspeak Cloud. The test is carried out with the same network conditions as the previous one. Figure 95 shows the Wireshark captures of the connection of the devices to the Thingspeak cloud.



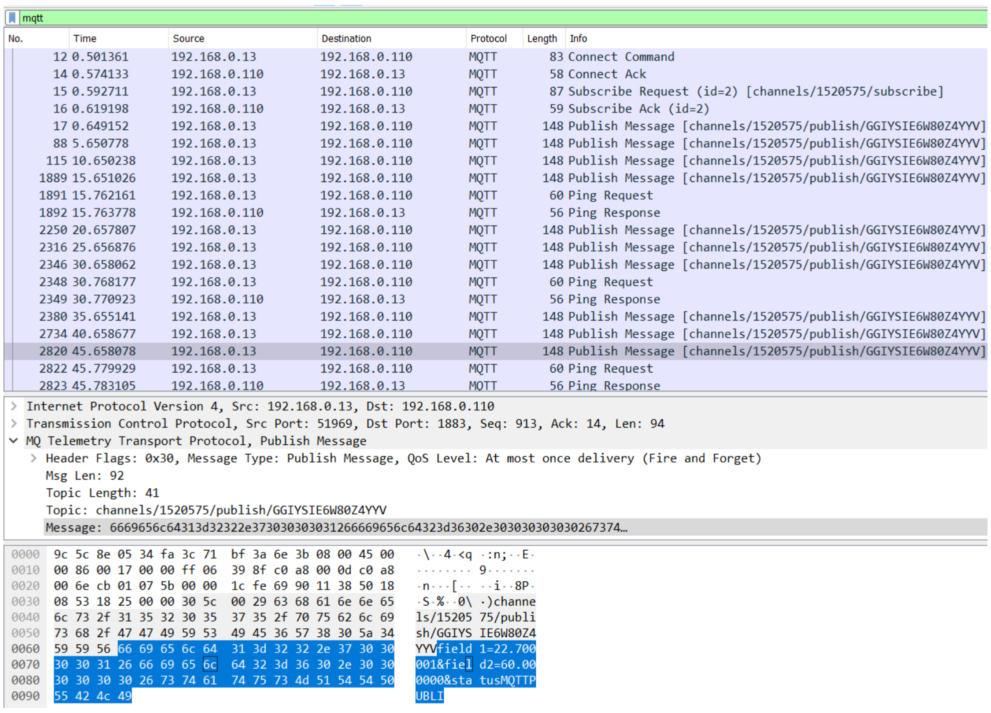
```
mqtt
No.    Time        Source          Destination     Protocol Length Info
24611 217.630996   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
25166 222.646738   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
25675 227.310793   192.168.10.39   192.168.10.31   MQTT     56 Ping Request
25677 227.311048   192.168.10.31   192.168.10.39   MQTT     56 Ping Response
25711 227.583311   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
26237 232.598860   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
27043 239.804840   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
27362 242.339169   192.168.10.39   192.168.10.31   MQTT     56 Ping Request
27363 242.339540   192.168.10.31   192.168.10.39   MQTT     56 Ping Response
27398 242.582034   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
28066 247.582258   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
28588 252.585040   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
29033 257.344804   192.168.10.39   192.168.10.31   MQTT     56 Ping Request
29034 257.345290   192.168.10.31   192.168.10.39   MQTT     56 Ping Response
29059 257.597716   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
29562 262.737338   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
30197 272.040484   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]
30222 272.353597   192.168.10.39   192.168.10.31   MQTT     56 Ping Request
30223 272.353881   192.168.10.31   192.168.10.39   MQTT     56 Ping Response
30245 272.656664   192.168.10.39   192.168.10.31   MQTT     84 Publish Message [casa/ambiente/s1_a1]

> Transmission Control Protocol, Src Port: 57932, Dst Port: 1883, Seq: 1533, Ack: 33, Len: 30
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 28
    Topic Length: 19
    Topic: casa/ambiente/s1_a1
    Message: 6e6f5f64617461

0000  54 8c a0 6f f7 a9 e8 db  84 af 0e 2f 08 00 45 00   T··o···· ···/··E·
0010  00 46 01 0f 00 00 ff 06  25 0c c0 a8 0a 27 c0 a8   ·F······ %····'··
0020  0a 1f e2 4c 07 5b 00 00  2a fd 0f c5 c9 97 50 18   ···L·[·· *·····P·
0030  08 37 f6 fc 00 00 30 1c  00 13 63 61 73 61 2f 61   ·7····0· ··casa/a
0040  6d 62 69 65 6e 74 65 2f  73 31 5f 61 31 6e 6f 5f   mbiente/ s1_a1no_
0050  64 61 74 61                                        data
```

Figure 95. Connection to Thingspeak cloud

The screenshot in the previous figure shows the data sent from a single device with a length between 56 and 84 Bytes per packet. This data is only monitored from the cloud and displayed in behaviour graphs. These messages are only published as they are used for display. In 5 seconds, three publish messages are sent, a ping request and a ping response are obtained. The test is repeated with a topic and a longer payload since it connects to an MQTT broker in the cloud. Figure 96 shows the packages needed to connect to the broker in the cloud. Among these are: Connect command, Connect Ack, Subscribe Request, Subscribe Response, and then three publish messages followed by a ping request and a ping response. All in an average of 5 seconds, and then they are repeated successively to publish new data.



```
mqtt
No.     Time        Source          Destination     Protocol  Length Info
   12 0.501361   192.168.0.13    192.168.0.110   MQTT       83 Connect Command
   14 0.574133   192.168.0.110   192.168.0.13    MQTT       58 Connect Ack
   15 0.592711   192.168.0.13    192.168.0.110   MQTT       87 Subscribe Request (id=2) [channels/1520575/subscribe]
   16 0.619198   192.168.0.110   192.168.0.13    MQTT       59 Subscribe Ack (id=2)
   17 0.649152   192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
   88 5.650778   192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
  115 10.650238  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 1889 15.651026  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 1891 15.762161  192.168.0.13    192.168.0.110   MQTT       60 Ping Request
 1892 15.763778  192.168.0.110   192.168.0.13    MQTT       56 Ping Response
 2250 20.657807  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2316 25.656876  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2346 30.658062  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2348 30.768177  192.168.0.13    192.168.0.110   MQTT       60 Ping Request
 2349 30.770923  192.168.0.110   192.168.0.13    MQTT       56 Ping Response
 2380 35.655141  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2734 40.658677  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2820 45.658078  192.168.0.13    192.168.0.110   MQTT      148 Publish Message [channels/1520575/publish/GGIYSIE6W80Z4YYV]
 2822 45.779929  192.168.0.13    192.168.0.110   MQTT       60 Ping Request
 2823 45.783105  192.168.0.110   192.168.0.13    MQTT       56 Ping Response

> Internet Protocol Version 4, Src: 192.168.0.13, Dst: 192.168.0.110
> Transmission Control Protocol, Src Port: 51969, Dst Port: 1883, Seq: 913, Ack: 14, Len: 94
v MQ Telemetry Transport Protocol, Publish Message
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 92
    Topic Length: 41
    Topic: channels/1520575/publish/GGIYSIE6W80Z4YYV
    Message: 6669656c64313d32322e37303030303031266669656c64323d36302e303030303030267374…

0000  9c 5c 8e 05 34 fa 3c 71  bf 3a 6e 3b 08 00 45 00   ·\··4·<q ·:n;··E·
0010  00 86 00 17 00 00 ff 06  39 8f c0 a8 00 0d c0 a8   ········ 9·······
0020  00 6e cb 01 07 5b 00 00  1c fe 69 90 11 38 50 18   ·n···[·· ··i··8P·
0030  08 53 18 25 00 00 30 5c  00 29 63 68 61 6e 6e 65   ·S·%··0\ ·)channe
0040  6c 73 2f 31 35 32 30 35  37 35 2f 70 75 62 6c 69   ls/15205 75/publi
0050  73 68 2f 47 47 49 59 53  49 45 36 57 38 30 5a 34   sh/GGIYS IE6W80Z4
0060  59 59 56 66 69 65 6c 64  31 3d 32 32 2e 37 30 30   YYVfield 1=22.700
0070  30 30 31 26 66 69 65 6c  64 32 3d 36 30 2e 30 30   001&fiel d2=60.00
0080  30 30 30 30 26 73 74 61  74 75 73 4d 51 54 54 50   0000&sta tusMQTTP
0090  55 42 4c 49                                        UBLI
```

Figure 96. Connection with MQTT Broker in the Thingspeak cloud

In Figure 96, it can see that the packets are larger than the previous one; each publishes kept at a constant value of 148 Bytes in each publication. Figure 97 shows the behaviour of the packets in a test interval of approximately 800 s.

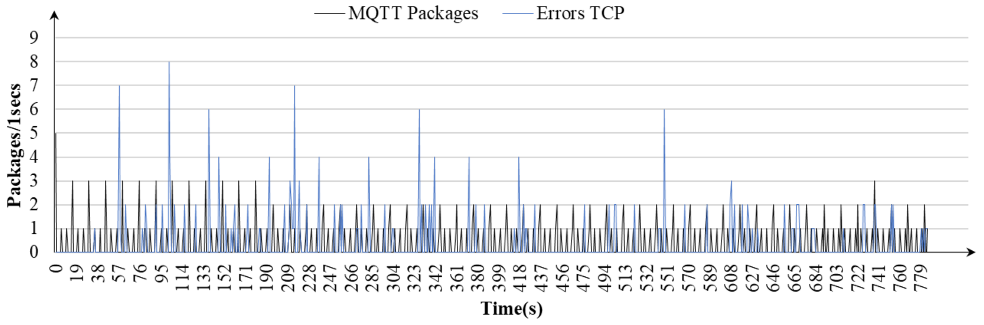


Figure 97. Number of packets transmitted to the Thingspeak cloud

The figure above maintains an average of 1 packet for the broadcast of the publish message, initially, with reconnections every 5 s. In a 50 s sample range, 10 publish messages are sent 5 s apart. This interval is repeated throughout the transmission, with drops of the TCP connection, increasing the packets' size from 1 minimum packet to 8 maximum packets. In some of these transmission errors, communication is re-established by sending all the 7 MQTT messages between requests and responses again. This makes these ridges appear high in the figure.

Figure 97 shows the RTT of the data going from the local network through the IoT-Gateway and then arriving at the new Thingspeak.

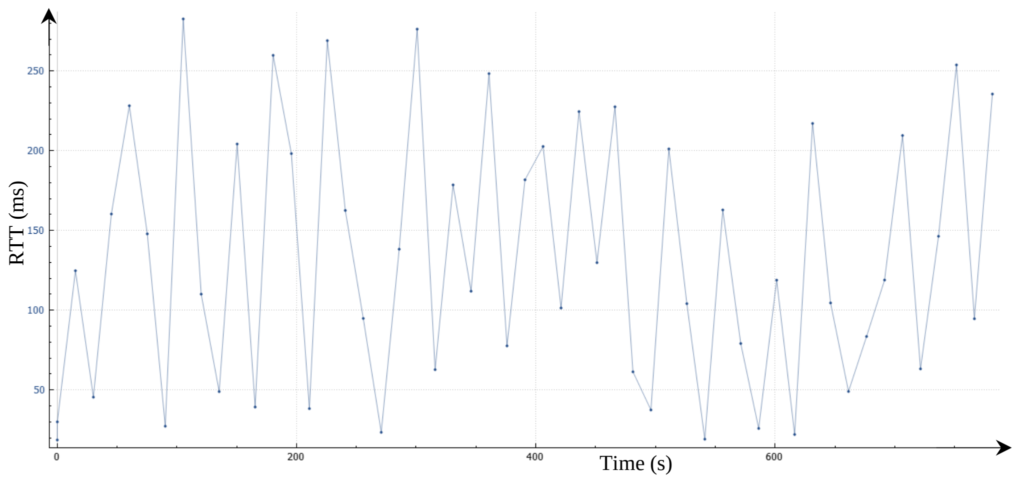
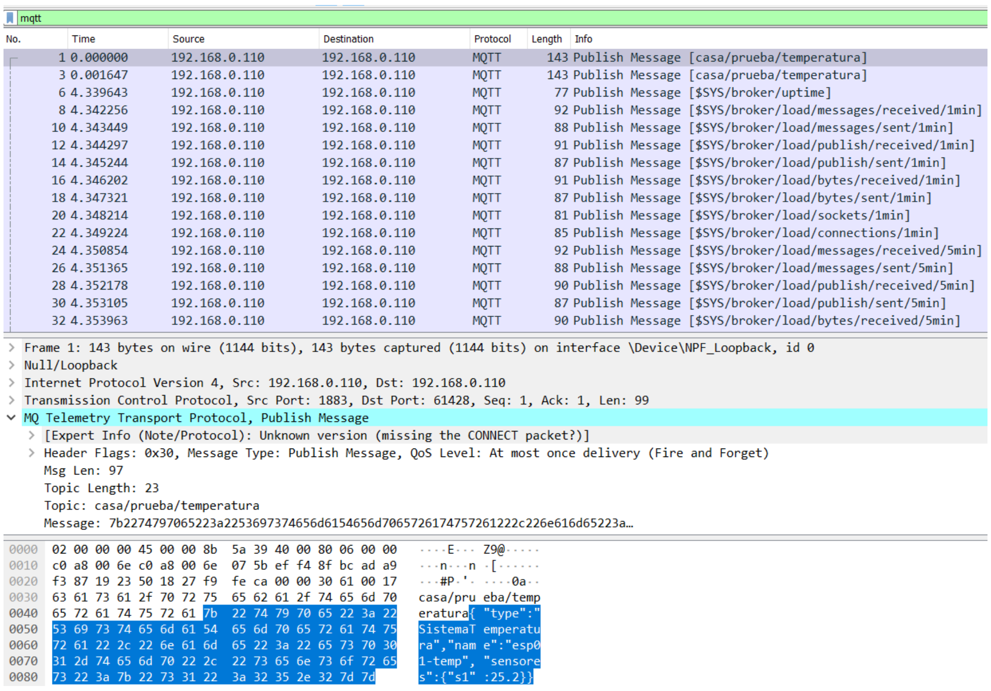


Figure 98. RTT with the Thingspeak cloud

The RTT times of a local connection for IoT devices seen in Figure 94 have an average of 0.0365ms; even when there is a disconnection, the peak reaches a maximum value of 0.063ms. Compared to this figure, the average value is approximately 150ms, which is still too high to maintain an M2M connection through the cloud.

Figure 99 shows M2M data capture between MQTT virtual clients such as MQTT Explore and MQTT Len. These devices can test send data and JSON structures in a development environment before deploying it on the devices.



```
mqtt
No.   Time       Source          Destination     Protocol  Length Info
    1 0.000000   192.168.0.110   192.168.0.110   MQTT       143 Publish Message [casa/prueba/temperatura]
    3 0.001647   192.168.0.110   192.168.0.110   MQTT       143 Publish Message [casa/prueba/temperatura]
    6 4.339643   192.168.0.110   192.168.0.110   MQTT        77 Publish Message [$SYS/broker/uptime]
    8 4.342256   192.168.0.110   192.168.0.110   MQTT        92 Publish Message [$SYS/broker/load/messages/received/1min]
   10 4.343449   192.168.0.110   192.168.0.110   MQTT        88 Publish Message [$SYS/broker/load/messages/sent/1min]
   12 4.344297   192.168.0.110   192.168.0.110   MQTT        91 Publish Message [$SYS/broker/load/publish/received/1min]
   14 4.345244   192.168.0.110   192.168.0.110   MQTT        87 Publish Message [$SYS/broker/load/publish/sent/1min]
   16 4.346202   192.168.0.110   192.168.0.110   MQTT        91 Publish Message [$SYS/broker/load/bytes/received/1min]
   18 4.347321   192.168.0.110   192.168.0.110   MQTT        87 Publish Message [$SYS/broker/load/bytes/sent/1min]
   20 4.348214   192.168.0.110   192.168.0.110   MQTT        81 Publish Message [$SYS/broker/load/sockets/1min]
   22 4.349224   192.168.0.110   192.168.0.110   MQTT        85 Publish Message [$SYS/broker/load/connections/1min]
   24 4.350854   192.168.0.110   192.168.0.110   MQTT        92 Publish Message [$SYS/broker/load/messages/received/5min]
   26 4.351365   192.168.0.110   192.168.0.110   MQTT        88 Publish Message [$SYS/broker/load/messages/sent/5min]
   28 4.352178   192.168.0.110   192.168.0.110   MQTT        90 Publish Message [$SYS/broker/load/publish/received/5min]
   30 4.353105   192.168.0.110   192.168.0.110   MQTT        87 Publish Message [$SYS/broker/load/publish/sent/5min]
   32 4.353963   192.168.0.110   192.168.0.110   MQTT        90 Publish Message [$SYS/broker/load/bytes/received/5min]

> Frame 1: 143 bytes on wire (1144 bits), 143 bytes captured (1144 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
> Internet Protocol Version 4, Src: 192.168.0.110, Dst: 192.168.0.110
> Transmission Control Protocol, Src Port: 1883, Dst Port: 61428, Seq: 1, Ack: 1, Len: 99
v MQ Telemetry Transport Protocol, Publish Message
  > [Expert Info (Note/Protocol): Unknown version (missing the CONNECT packet?)]
  > Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
    Msg Len: 97
    Topic Length: 23
    Topic: casa/prueba/temperatura
    Message: 7b2274797065223a2253697374656d6154656d706572617475726122 2c226e616d65223a…

0000  02 00 00 00 45 00 00 8b  5a 39 40 00 80 06 00 00   ····E··· Z9@·····
0010  c0 a8 00 6e c0 a8 00 6e  07 5b ef f4 8f bc ad a9   ···n···n ·[······
0020  f3 87 19 23 50 18 27 f9  fe ca 00 00 30 61 00 17   ···#P·'· ····0a··
0030  63 61 73 61 2f 70 72 75  65 62 61 2f 74 65 6d 70   casa/pru eba/temp
0040  65 72 61 74 75 72 61 7b  22 74 79 70 65 22 3a 22   eratura{ "type":"
0050  53 69 73 74 65 6d 61 54  65 6d 70 65 72 61 74 75   SistemaT emperatu
0060  72 61 22 2c 22 6e 61 6d  65 22 3a 22 65 73 70 30   ra","nam e":"esp0
0070  31 2d 74 65 6d 70 22 2c  22 73 65 6e 73 6f 72 65   1-temp", "sensore
0080  73 22 3a 7b 22 73 31 22  3a 32 35 2e 32 7d 7d      s":{"s1" :25.2}}
```

Figure 99. Wireshark capture using JSON and MQTT Explore and MQTT Lens virtual clients

6.4. Conclusion

The time results obtained in the data transmission tests between objects in a local network compared to the same objects, but connected through the cloud, evidenced a high delay when controlling an IoT device using the cloud. These results support the use of IoT networks locally and proposing a different use of the cloud, such as consultations with other clouds and storage of large volumes of information and its processing, consulted asynchronously. This allows a high degree of fault tolerance since objects can be controlled locally without the internet. In this way, it can continue to take advantage of cloud resources through asynchronous connections only when the local or main network requires it.

Chapter 7. Conclusion and future lines of research

7.1. Introduction

This thesis has largely been done with simulations because there are still no smart IoT-Networks interconnected through the cloud in different environments. Although in a real situation, a user acquires a Thing with AI included and connects it to his usual network in his home or office, these networks could not intercommunicate things locally because their system is simple and designed for traditional devices and not for things with AI. The only way to get a minimum profit is for the device to send the information to a cloud platform so that this is the one who manages all its requests. Hence the need for this architecture, which manages all information and internet access locally through a Gateway and externally through AI in the cloud. However, the results of the tests showed how smart things would perform on an intelligent architecture designed for this type of scenario. There are conclusions presented at the end of each section in which the models (Chapter 3) and the testbed of each experiment (Chapter 4) using these models are shown. Therefore, the global conclusions of this thesis are presented below with an overview of their contribution and advantages when implementing them in real situations.

7.2. Conclusion and contributions

By centralizing the data in the programmable IoT-Gateway, which replaces the Fog or Network layer with a single device or leader, it is ensured that each object reports its data in a structured way to the same point in the network. Therefore, the data extracted from the objects (features) are combined with the network parameters facilitating their treatment. In this way, a single dataset is created, which the Gateway’s AI uses for grouping and routing and, in turn, in managing internet access and its relations with the cloud’s AI (Level 3).

By separating the IoT layer into three layers maintaining the functionalities of each object as a stand-alone system, it makes it easier for the Gateway’s AI (Level 2) to more accurately group objects. In addition, it allows better identification, and thus it can create M2M routing between objects in the same group based on their resources as evidenced by the tests. In the same way, by maintaining its functions stand-alone, the object’s AI (Level 1) can do deep learning and reinforcement learning on the habits of use of its users, estimate a profile, preferences and most used resources and even predict when and how it will be used.

In the cloud, the architecture shows an integration between different clouds based on the type of parameter and the service it provides to its main local network. This extension of the IoT proximity networks through the cloud constitutes an advantage over other types of conventional networks since the Cloud's AI learns which clouds process information based on the types of parameters it requests and puts it at the user's service in any location in the world. Through the implemented applications shown in section 5.1.2, it is possible to observe the coverage and scope of the network through the internet at any user location.

Currently, users do not have access to their data or their objects connected to the internet. These are managed directly by the manufacturers of the objects that the user buys and connects to the internet through the mobile application they provide for their control and monitoring. It means that a user delivers his data through the objects that he buys from the manufacturers, without the possibility in most cases to manage his data. According to them, this is done to improve and make new versions of Apps available to users to provide a better service. But in reality, it is only possible to create rules that automate some tasks and unify several manufacturers' brands through alliances on a single platform to make them compatible. For now, there is no cloud computing architecture service that does it through an AI, as proposed in this thesis. That is that the management and automation are completely limited to the manufacturers' rules available in the Apps. In addition to delays, many devices cannot be controlled locally if there is no internet connection. Not being enough, these manufacturers continue to keep the data extracted from users for other purposes such as marketing analysis and user consumption preferences. Data that are not used to intelligently improve the interaction between objects and thus revolutionize the operation of IoT-Networks, but is used for commercial benefits. This thesis shows that when using this architecture, the data stays in the local network and is only shared with the cloud within the same network domain. In this way, data management is intelligent, and AI can structure its data set adjusted to the profile of its user and the services it must provide. In this case, manufacturers should ask permission to use this data and even buy it. In this order of ideas, each network would be adjusted to each user, and although they are based on the same architecture, the data set may be structured differently depending on the analysis of each AI.

There is no source of datasets available on the internet under these scenarios to do research, nor on real networks since they do not exist. Therefore, the data used was extracted from the objects using simulations or some real implementations. With this data, it was possible to show how an AI hosted in a Gateway can be trained to create a dataset structure that is then shared through the AI interface using the three levels of processing. This communication and mutual agreement between these AIs solve preprocessing, connection management, and network scope and coverage problems.

Today's IoT-Networks are based on events, rules and permissions that require constant monitoring by the user. The contribution of this thesis will allow the machines to work collaboratively, intelligently and without supervision. This, in addition to its ease of installation through architecture, also allows the response times of systems that are grouped together to provide faster service to the user. Therefore, a user does not have to waste time connecting through an application or an assistant to activate one Thing in different environments (home, city, or work). Each Thing is automatically connected in anticipation of the service before the user requests it. This reduces the time to establish the connection since the integration of the architectures allows the resolution and attention of service to be made from where the data originates (Edge) or from where it is managed (Fog) or from where it is processed (Cloud) using a single architecture.

Although throughout the thesis there has been talked of IoT-Networks, the truth is that the IoT as it is currently known and as it is being commercialized is still far from becoming IoT-Networks, since they could not be called IoT-Networks since they do not have an integrated system of shared resources, segmentation and own routing. The traditional conventions based on IP computer networks are still used. However, through its architecture and taking advantage of AI, this thesis presents new alternatives to share resources, changing the conventional segmentation by clustering (creating collaborative workgroups) and the conventional IP routing by routing based on the best node.

7.3. Future lines of research

ML, PL, RL algorithms have been little applied to data extracted from objects in the IoT. Few the datasets can be obtained from a real IoT-Network that contains the features of the objects, events, and parameters of the network to apply these models. However, through this thesis, it was demonstrated that it has a dataset of different connected objects is possible to use different AI models on the proposed architecture and obtain IoT networks with new features and functionalities according to smart things. Therefore, it is possible to extend the study carried out in this research in multiple directions. In the following, we describe some proposals that may be interesting for further research in terms of interoperability and the use of AI.

- Extend the architecture functionalities:

The implementations of this IoT-SmartArchitecture require a smart IoT-Gateway, where the network administration is centralized, and a Smart IoT-Platform in the cloud where the network is extended through the interconnection with other clouds. Therefore, continuing with the development and implementation of different AI models at these points where the information is centralized would expand the functionalities of the architecture that integrates smart things.

It has been seen that with the increase of WiFi and BLE modules in sensors and everyday objects complemented with wired networks (Ethernet), they allow IoT-Gateway to be easier to implement through SBCs. With these two technologies, a complete IoT-Network can be built. However, some sensors and objects remain using short-range (ZigBee, 6LoWPAN) and long-range (LoRA) wireless networks. Some IoT-Gateway could be implemented and gather a complete dataset with features of this type of object and network parameters of these connections. Thus, it could be observed with more information how the AI would classify these objects within workgroups and how it would route them.

7.4. Faced problems

The main difficulties presented during the realization of this thesis were the simulation and implementation of the IoT network scenarios. It is because simulators are still in the development stage and are used for teaching some IoT technologies, so using them to test data transmission and see how the network behaved was very difficult. It had to complement it with real tests on real devices since, in the simulation, It could only see the behaviour of the data transmission and the protocol, and with the real devices, the behaviour at the network level. The architecture proposed in this thesis seeks to classify the objects of the IoT layer (according to other architectures) in different layers through their characteristics. Among them, separating this layer is carried out based on its functions, maintaining its layer independence but collaborating as a group to provide a service. So achieving a scenario of using the things layer in the tests was a great challenge since there are still no smart or reprogrammable things in the current market. We had to use an RPi3 to emulate the operation of smart things, such as appliances. These were easier to implement through a smart home setting, as it uses WiFi and Bluetooth technology and is more affordable to test in a local environment. In this way, tests can be done at home and not necessarily in a laboratory.

The dataset of an IoT network under certain architectures is not easy to obtain, nor is it available in the current literature sources, so it was necessary to build it from multiple tests under the organizational policies of the proposed architecture. It involved a great effort since to achieve these tests, it was necessary to coordinate between simulated and implemented in real devices. As projected at the beginning of the thesis, these scenarios do not yet exist in homes or industries; there are only some commercial approaches to what the IoT is thought to be from a commercial point of view.

7.5. Personal contributions

From the thesis, I learned how to propose better design schemes for IoT networks from everyday life. Most people do not distinguish the engineering concepts embedded in the systems because they are manipulated by commercial advertising that seeks to sell. However, from these business schemes and current academic proposals, I could discern differences and potential changes that would help improve the next generations of IoT networks. I learned to take advantage of simulators that are seen only as teaching, and I was able to simulate systems in scenarios that do not yet exist. In addition, real devices were implemented with operations different from those commercially conceived, changing their logic and accommodating it to the proposed architecture.

The appearance of the Smart IoT-Networks, together with the massification of objects with AI included (Smart Things), implies a greater challenge than required by a conventional data network. Many IoT objects are activated by the user in these networks through commands, either with the mobile phone or an intelligent assistant. On the other hand, in Smart IoT-Networks, the objects can be autonomous and create their relations. However, no one knows yet how they will behave, since there are not yet a significant number of objects working together to achieve a common goal. There is very little information about these types of networks and how they will work in the future. Therefore, it is possible to suppose different scenarios in different ways to propose its possible operation. Considering the above, it opens opportunities to propose different architectures that support the transport of information that an AI requires and solves interoperability between heterogeneous objects.

The main goal is to use the proposed IoT-SmartArchitecture architecture as the basis for the future design of Smart Networks. However, there are no real Smart Network scenarios that provide a suitable ecosystem for smart objects. For this reason, it is proposed a possible Smart Network scenario on this architecture, assuming that smart objects will work collectively to provide automatic services to users. Therefore, this paper focuses on the algorithm for creating workgroups (grouping) in a Smart IoT-Network and its routing. In which the Architecture's AI uses an ML classifier with a dataset based on the features extracted from different objects. With this information, the classifier assigns a workgroup to a "new object" connecting to the network's first time. In addition to this, it classifies it and assigns it a role in one layer of the proposed architecture. Although IoT can be a much broader vision that implies a global infrastructure in all aspects, we will only study the cases present in proximity networks. Therefore, the test scenario is a Smart Home based on wireless technologies due to its ease of simulation.

Currently, conventional networks can include everyday objects with the ability to connect. However, if the object includes AI, this type of network can limit its potential. This paper shows what a Smart Home would look like in the future and recreates possible scenarios where objects are related to serving users. The above is necessary since the proposed architecture is designed to organize objects that include AI. Therefore, without scenarios based on an AI's control, it would not be possible to test this architecture. In turn, the obtained algorithms can be used in similar applications like Industrial Internet of Things (IIoT) environments, among others. Furthermore, this work shows the Discovery of Functions and Services Protocol (DFSP) protocol's use over an IoT-Protocol and implementing new messages that facilitate AI work on the architecture. It also presents the accuracy of different AI techniques to classify objects in workgroups, which will help determine which the most adequate to apply in Smart IoT-Networks. Finally, it is shown the importance of replacing the conventional router with one more specialized and higher capacity that keeps the network's centrality.

7.6. Publications derived from the PhD Thesis

The journal publications derived from the work that has been presented in this thesis are the following ones:

P. L. G. Ramirez, M. Taha, J. Lloret, and J. Tomas, "An Intelligent Algorithm for Resource Sharing and Self-Management of Wireless-IoT-Gateway," IEEE Access, vol. 8, pp. 3159–3170, 2020, doi: 10.1109/ACCESS.2019.2960508. **JCR Impact Factor (Q1)**.

P. L. G. Ramirez, J. Lloret, J. Tomás, and M. Hurtado, "IoT-Networks group-based model that uses AI for workgroup allocation," Computer Networks, p. 107745, Dec. 2020, doi: 10.1016/j.comnet.2020.107745. **JCR Impact Factor (Q1)**.

A. Rego, **P. L. G. Ramírez**, J. M. Jimenez, and J. Lloret, "Artificial intelligent system for multimedia services in smart home environments," Cluster Computing Journal (Springer), 2021, doi: 10.1007/s10586-021-03350-z. **JCR Factor (Q3)**.

Furthermore, the publications in international conferences that include the work presented in this thesis are the ones listed below:

J. Lloret, S. Sendra, **P. L. G. Ramirez**, L. Parra, "An IoT Group-Based Protocol for Smart City Interconnection". Communications in Computer and Information Science (Springer), ISSN: 18650929, v.978, p.164-178, 21 February 2019. DOI: https://doi.org/10.1007/978-3-030-12804-3_13. JCR Impact Factor (Q3). Conference proceedings First Ibero-American Congress, ICSC-CITIES 2018, Soria, Spain, September 26–27, 2018, book series (CCIS, volume 978): https://link.springer.com/book/10.1007/978-3-030-12804-3

P. L. G. Ramirez, J. Lloret, T. Miran, J. Tomás, "Architecture to Integrate IoT Networks using Artificial Intelligence in the Cloud". 5th Annual International Conference on Computational Science and Computational Intelligence (CSCI'18), 13-15 December 2018, Las Vegas, Nevada, USA. Publisher: IEEE Computer Society, IEEE Xplore. IEEE

Catalog Number: CFP1871X-USB. ISBN-13: 978-1-7281-1360-9; Pages 996-1001, doi: 10.1109/CSCI46756.2018.00193.

P. L. G. Ramirez, J. Lloret, J. Tomás, O. Rodriguez and M. Hurtado, "IoT-WLAN Proximity Network for Potentiostats," 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 94-99, doi: 10.1109/FMEC49853.2020.9144776.

7.7. Other publications

P. L. G. Ramirez, J. Lloret, S. Martínez Cordero, L. C. Trujillo Arboleda, "Design and Implementation of ForCES Protocol". Network Protocols and Algorithms - Macrothink Institute, ISSN: 1943-3581, v.9, fasc.1-2, p.1 - 27, June 30 - 2017. DOI: https://doi.org/10.5296/npa.v9i1-2.10943

P. L. G. Ramirez, J. Lloret, S. Martínez Cordero, L. C. Trujillo Arboleda, "Design and implementation of a prototype of the entity Control Element (CE) of the Architecture ForCES". Network Protocols and Algorithms - Macrothink Institute, ISSN: 1943-3581, v.9, fasc.3-4, p.1 - 30, November-2017. DOI: https://doi.org/10.5296/npa.v9i3-4.12433

P. L. G. Ramirez, J. Lloret, Poster: "IoE architecture based on parameter groups". Second International MEMS-AL School: Microsystems for Latin America. Tecnológico de Monterrey (Monterrey - Mexico). November 21-27, 2018. https://mems-al-2018.sciencesconf.org/

Chapter 8. Bibliography

[1] F. Aïssaoui, G. Garzone, and N. Seydoux, "Providing interoperability for autonomic control of connected devices," in *InterIoT,* 2016, pp. 1–7, doi: 10.1007/978-3-319-52727-7_5.

[2] Y. Wang, L. Wei, Q. Jin, and J. Ma, "Alljoyn based direct proximity service development: Overview and prototype," in *Proceedings - 17th IEEE International Conference on Computational Science and Engineering, CSE 2014, Jointly with 13th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2014, 13th International Symposium on Pervasive Systems,* , Dec. 2015, pp. 634–641, doi: 10.1109/CSE.2014.138.

[3] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," in *2014 IEEE World Forum on Internet of Things, WF-IoT 2014,* Mar. 2014, pp. 514–519, doi: 10.1109/WF-IoT.2014.6803221.

[4] M. Villari, A. Celesti, M. Fazio, and A. Puliafito, "AllJoyn Lambda: An architecture for the management of smart environments in IoT," in *Proceedings of 2014 International Conference on Smart Computing Workshops, SMARTCOMP Workshops 2014,* Nov. 2015, pp. 9–14, doi: 10.1109/SMARTCOMP-W.2014.7046676.

[5] A. P. Castellani, N. Bui, P. Casari, M. Rossi, Z. Shelby, and M. Zorzi, "Architecture and protocols for the internet of things: A case study," in *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2010,* Mar. 2010, pp. 678–683, doi: 10.1109/PERCOMW.2010.5470520.

[6] AllJoyn, "Open source software framework," 2016. https://openconnectivity.org/developer/reference-implementation/alljoyn.

[7] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, “A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration,” *ACM Comput. Surv.*, vol. 51, no. 6, 2019, doi: 10.1145/3292674.

[8] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, “A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT),” in *2015 Internet Technologies and Applications (ITA)*, Sep. 2015, pp. 219–224, doi: 10.1109/ITechA.2015.7317398.

[9] S. Schei, “Understanding Data Analysis in an End-to-End IoT System,” 2016.

[10] K. Rose, S. Eldridge, and L. Chapin, “La Internet De Las Cosas — Una Breve Reseña,” *Internet Soc.*, p. 83, 2015, Accessed: Jul. 04, 2018. [Online]. Available: https://www.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf.

[11] N. N. Srinidhi, S. M. Dilip Kumar, and K. R. Venugopal, “Network optimizations in the Internet of Things: A review,” *Eng. Sci. Technol. an Int. J.*, vol. 22, no. 1, pp. 1–21, Feb. 2019, doi: 10.1016/j.jestch.2018.09.003.

[12] IETF, “Internet of Things,” 2018. https://www.ietf.org/topics/iot/.

[13] IEEE, “Internet of Things,” 2018. https://iot.ieee.org/.

[14] A. B. Chebudie, R. Minerva, and D. Rotondi, “Towards a definition of the Internet of Things (IoT),” 2014. [Online]. Available: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf.

[15] AIOTI, “The Alliance for Internet of Things Innovation,” *Digital Single Market*, 2018. https://aioti.eu/.

[16] ANSI/ISA, “IEC62443/ISA99 - Industrial Automation and Control Systems Security,” *ISA*, 2018. https://www.isa.org/isa99/.

[17] OIC, “Open Interconnect Consortium,” 2016. https://openconnectivity.org/news/open-interconnect-consortium-helps-developers-tackle-internet-of-things-with-new-developer-toolkit-2.

[18] IIC, “Industrial Internet Consortium,” 2015. https://www.iiconsortium.org/.

[19] ISO, “International Organization for Standardization,” *Normas ISO*, 2018. https://www.iso.org/home.html.

[20] ISO/IEC, “Information technology Smart cities,” *Smart cities - Prelim. Rep. 2014*, pp. 1–71, 2014, doi: 10.1109/HICSS.2012.615.

[21] UIT, “Committed to connecting the world,” pp. 18–20, 2011, [Online]. Available: https://www.itu.int/en/Pages/default.aspx.

[22] U. I. de T. ITU, “Descripción General de Internet de los Objetos Y.2060- Y.4000,” *Y.2060 Y.4000*, p. 20, 2016, [Online]. Available:

http://handle.itu.int/11.1002/1000/11559.

[23] OASIS, "Advancing Open Standards for the information society," 2011. https://www.oasis-open.org/.

[24] IPSO, "IPSO Smart Objects," 2016. https://www.omaspecworks.org/develop-with-oma-specworks/ipso-smart-objects/.

[25] OZOM, "OZOM Smarthome for everyone," 2017. https://ozom.com/es/ (accessed May 26, 2017).

[26] OZOM, "Control your home remotely," 2018. https://ozom.com/marcas-compatibles (accessed Jan. 21, 2021).

[27] Google Inc., "Google Assistant," 2018. https://developers.google.com/assistant.

[28] Google Inc., "Create a smart home Action | Actions on Google Smart Home." https://developers.google.com/assistant/smarthome/develop/create.

[29] Amazon, "Amazon Alexa Official Site: What is Alexa?," 2019. https://developer.amazon.com/es-MX/alexa (accessed Nov. 18, 2019).

[30] VTA, "Casa Inteligente, Productos Innovadores De Audio, Seguridad Y Domótica." https://www.vta.co/ (accessed Jan. 22, 2021).

[31] D. Evans, "The Internet of Things - How the Next Evolution of the Internet is Changing Everything," *CISCO white Pap.*, no. April, pp. 1–11, 2011, doi: 10.1109/IEEESTD.2007.373646.

[32] CISCO, "Internet of Things (IoT)," 2016. https://www.cisco.com/c/en/us/solutions/internet-of-things/overview.html.

[33] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," *J. Electr. Comput. Eng.*, vol. 2017, 2017, doi: 10.1155/2017/9324035.

[34] P. Mell and T. Grance, "Final Version of NIST Definition of Cloud Computing," *Technical report, National Institute of Standards and Technology*, no. October. pp. 1–2, 2011, Accessed: Jul. 28, 2020. [Online]. Available: https://www.nist.gov/news-events/news/2011/10/final-version-nist-cloud-computing-definition-published.

[35] MathWorks, "What Machine Learning Model is Right Algorithm Table," 2021. https://www.mathworks.com/content/dam/mathworks/data-sheet/what-ml-model-is-right-algorithm-table.pdf (accessed Jan. 13, 2021).

[36] Mathworks, "Which Machine Learning Algorithm is Right for You?," 2021. https://explore.mathworks.com/choosing-machine-learning-algorithms/landing-121LO-9276S.html#dataset (accessed Jan. 13, 2021).

[37] MathWorks, "Choose Classifier Options - MATLAB & Simulink," 2021. https://www.mathworks.com/help/stats/choose-a-classifier.html (accessed Jan. 13, 2021).

[38] MathWorks, "Interpretability - MATLAB & Simulink," 2021.

https://www.mathworks.com/discovery/interpretability.html (accessed Jan. 13, 2021).

[39] MathWorks, “Hyperparameter Optimization in Classification Learner App,” 2020. https://www.mathworks.com/help/stats/hyperparameter-optimization-in-classification-learner-app.html (accessed Jan. 13, 2021).

[40] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, “Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture,” *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4183–4194, May 2020, doi: 10.1109/JIOT.2019.2944695.

[41] D. Kimovski, H. Ijaz, N. Saurabh, and R. Prodan, “Adaptive Nature-Inspired Fog Architecture,” in *2018 IEEE 2nd International Conference on Fog and Edge Computing, ICFEC 2018 - In conjunction with 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, IEEE/ACM CCGrid 2018*, 2018, pp. 1–8, doi: 10.1109/CFEC.2018.8358723.

[42] M. G. dos Santos, D. Ameyed, F. Petrillo, F. Jaafar, and M. Cheriet, “Internet of Things Architectures: A Comparative Study,” 2020, Accessed: Jun. 05, 2020. [Online]. Available: http://arxiv.org/abs/2004.12936.

[43] J. Lloret, S. Sendra, P. L. González Ramírez, and L. Parra, “An IoT Group-Based Protocol for Smart City Interconnection,” in *Communications in Computer and Information Science*, 2019, vol. 978, pp. 164–178, doi: 10.1007/978-3-030-12804-3_13.

[44] J. Lloret, M. Garcia, J. Tomas, and J. J. P. C. Rodrigues, “Architecture and protocol for intercloud communication,” *Inf. Sci. (Ny).*, vol. 258, pp. 434–451, 2014, doi: 10.1016/j.ins.2013.05.003.

[45] P. L. G. Ramirez, M. Taha, J. Lloret, and J. Tomas, “An Intelligent Algorithm for Resource Sharing and Self-Management of Wireless-IoT-Gateway,” *IEEE Access*, vol. 8, pp. 3159–3170, 2020, doi: 10.1109/ACCESS.2019.2960508.

[46] P. Gonzalez, J. Lloret, J. Tomas, O. Rodriguez, and M. Hurtado, “IoT-WLAN Proximity Network for Potentiostats,” in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, Apr. 2020, pp. 94–99, doi: 10.1109/FMEC49853.2020.9144776.

[47] V. Cerf and R. Kahn, “A Protocol for Packet Network Intercommunication,” *IEEE Trans. Commun.*, vol. 22, no. 5, pp. 637–648, 1974, doi: 10.1109/TCOM.1974.1092259.

[48] A. E. Khaled and S. Helal, “Interoperable communication framework for bridging RESTful and topic-based communication in IoT,” *Futur. Gener. Comput. Syst.*, vol. 92, pp. 628–643, Mar. 2019, doi: 10.1016/j.future.2017.12.042.

[49] P. Thota and Y. Kim, “Implementation and Comparison of M2M Protocols for Internet of Things,” in *Proceedings - 4th International Conference on Applied*

Computing and Information Technology, 3rd International Conference on Computational Science/Intelligence and Applied Informatics, 1st International Conference on Big Data, Cloud Computing, Data Science, Dec. 2017, pp. 43–48, doi: 10.1109/ACIT-CSII-BCD.2016.021.

[50] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S - A publish/subscribe protocol for wireless sensor networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, Jan. 2008, pp. 791–798, doi: 10.1109/COMSWA.2008.4554519.

[51] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[52] P. L. Gonzalez Ramirez, J. Lloret, M. Taha, and J. Tomas, "Architecture to Integrate IoT Networks Using Artificial Intelligence in the Cloud," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2018, pp. 996–1001, doi: 10.1109/CSCI46756.2018.00193.

[53] S. Cope, "Understanding the MQTT Protocol Packet Structure," *Steve's internet*, 2018. http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/.

[54] Jupyter Org, "Project Jupyter | Jupyter Software," 2020. https://jupyter.org/ (accessed Feb. 12, 2020).

[55] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011, [Online]. Available: http://scikit-learn.sourceforge.net.

[56] P. L. González Ramírez, J. Lloret, J. Tomás, and M. Hurtado, "IoT-networks group-based model that uses AI for workgroup allocation," *Comput. Networks*, vol. 186, p. 107745, Feb. 2021, doi: 10.1016/j.comnet.2020.107745.

[57] Deep Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, 2nd Editio. Elsevier, 2018.

[58] A. Rego, S. Sendra, J. M. Jimenez, and J. Lloret, "Dynamic metric OSPF-based routing protocol for Software Defined Networks," *Cluster Comput.*, vol. 22, no. 3, pp. 705–720, Sep. 2019, doi: 10.1007/s10586-018-2875-7.

[59] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002, doi: 10.1109/4235.996017.

[60] K. Deb *et al.*, "Evolutionary Multi-Criterion Optimization," *Springer*, vol. 11411, pp. 1–768, 2019, doi: 10.1007/978-3-030-12598-1.

[61] Y. Vesikar, K. Deb, and J. Blank, "Reference Point Based NSGA-III for Preferred Solutions," in *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018*, Jan. 2019, pp. 1587–1594, doi:

10.1109/SSCI.2018.8628819.

[62] A. Rego, P. L. G. Ramírez, J. M. Jimenez, and J. Lloret, “Artificial intelligent system for multimedia services in smart home environments,” *Cluster Comput.*, 2021, doi: 10.1007/s10586-021-03350-z.

[63] T. Kurbiel and S. Khaleghian, “Training of Deep Neural Networks based on Distance Measures using RMSProp,” 2017, [Online]. Available: http://arxiv.org/abs/1708.01911.

[64] Y. A. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, “Efficient backprop,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7700 LECTU, pp. 9–48, 2012, doi: 10.1007/978-3-642-35289-8_3.

[65] “ROBOT LEARNING, edited by Jonathan H. Connell and Sridhar Mahadevan, Kluwer, Boston, 1993/1997, xii+240 pp., ISBN 0-7923-9365-1.,” *Robotica*, vol. 17, no. 2, pp. 229–235, 1999, doi: DOI: 10.1017/S0263574799271172.

[66] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. J. Barth, “Deep reinforcement learning-based vehicle energy efficiency autonomous learning system,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, Jul. 2017, pp. 1228–1233, doi: 10.1109/IVS.2017.7995880.

[67] Raspberry Pi, “Raspberry Pi 3 Model B+: Small single-board computers,” 2018. https://www.raspberrypi.org/products/.

[68] ThingsBoard, “ThingsBoard IoT Platform,” 2018. https://thingsboard.io/.

[69] The Mathworks Inc, “ThingSpeak Internet of Things,” 2017. https://thingspeak.com/.

[70] Google Inc., “Android Things.” 2015, [Online]. Available: https://developer.android.com/things/.

[71] Google Inc, “Android Studio.” https://developer.android.com/studio/ (accessed Jun. 18, 2019).

[72] Cooja, “Cooja: Contiki network simulator,” 2018. http://www.contiki-os.org/start.html#start-cooja.

[73] Thingsquare, “Get Started with Contiki and Cooja,” 2016. http://www.contiki-os.org/start.html (accessed May 16, 2019).

[74] T. Issariyakul and E. Hossain, *Introduction to network simulator NS2*, vol. 9781461414. 2012.

[75] P. P. Jeremy Grossmann, Dominik Ziajka, “GNS3 Simulator.” https://www.gns3.com/ (accessed Jun. 18, 2019).

[76] Cisco Academy, “Packet Tracer Simulator | Networking Academy,” *Cisco Academy*, 2019. https://www.netacad.com/courses/packet-tracer (accessed May 16, 2019).

[77] Proteus, “Raspberry Pi | Visual IoT Programming - Proteus Design Suite.” https://www.labcenter.com/raspberry_pi/?gclid=EAIaIQobChMInaitxZah4gIVl0wNCh3yjgNzEAAYASAAEgK3H_D_BwE (accessed May 16, 2019).

[78] Microsoft Azure, “Raspberry Pi Azure IoT Web Simulator.” https://azure-samples.github.io/raspberry-pi-web-simulator/.

[79] IBM and Eurotech, “MQTT V3.1 Protocol Specification,” 2010. https://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html (accessed May 08, 2019).

[80] P. R. Egli, “An introduction to MQTT, a protocol for M2M and IoT applications,” *Indigoo.com*, 2015.

[81] Paessler, “NetFlow Traffic Analyzer,” 2019. https://www.paessler.com/netflow_monitoring.

[82] S.-H. P. Khamdamboy Urunov, Jung-Il Namgung, Soo Young Shin, “The Interworking Functions based on Service Discovery Protocols (SDP) for Constrained Environment M2M/UIoT Communication,” in *Proceedings of the IEICE Conference*, 2016, pp. 742–745.

[83] V. K. Bryan Boyd, Joel Gauci, Michael P Robertson, Nguyen Van Duy, Rahul Gupta, Vasfi Gucer, *Building Real-time Mobile Solutions with MQTT and IBM MessageSight*, First edit. Place of publication not identified IBM Corporation International Technical Support Organization 2014, 2014.

[84] J. Rocher, M. Taha, L. Parra, and J. Lloret, “IoT Sensor to detect fraudulent use of dyed fuels in smart cities,” in *2018 5th International Conference on Internet of Things: Systems, Management and Security, IoTSMS 2018*, Nov. 2018, pp. 86–92, doi: 10.1109/IoTSMS.2018.8554631.

[85] “WIFI LoRa 32 (V2) – Heltec Automation,” 2019. https://heltec.org/project/wifi-lora-32/ (accessed Mar. 11, 2020).

[86] E. Longo, A. E. C. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, “MQTT-ST: a Spanning Tree Protocol for Distributed MQTT Brokers,” 2019, Accessed: Apr. 22, 2020. [Online]. Available: https://github.com/krylovsk/mqtt-benchmark.

[87] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments,” *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, Jun. 2016, doi: 10.1002/spe.2509.

[88] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, “A job scheduling algorithm for delay and performance optimization in fog computing,” *Concurr. Comput. Pract. Exp.*, vol. 32, no. 7, Apr. 2020, doi: 10.1002/cpe.5581.

[89] RAKwireless, “WisGate Developer D4+ (EG95-NA) / US915.” https://store.rakwireless.com/products/rak7244-lpwan-developer-gateway?variant=36313262129310 (accessed Nov. 06, 2020).

[90] K. Kolderup, “Bluetooth Mesh Networking,” 2017. [Online]. Available: http://blog.bluetooth.com/introducing-bluetooth-mesh-networking.

[91] P. Gomathinayagam and S. Jayanthy, “Implementation of mesh network using bluetooth low energy devices,” in *Lecture Notes in Electrical Engineering*, 2018, vol. 446, pp. 205–213, doi: 10.1007/978-981-10-4852-4_19.

[92] J. Blank and K. Deb, “Pymoo: Multi-Objective Optimization in Python,” *IEEE Access*, vol. 8, pp. 89497–89509, Jan. 2020, doi: 10.1109/ACCESS.2020.2990567.

[93] J. Blank, “pymoo - Getting Started.” https://pymoo.org/getting_started.html (accessed Dec. 04, 2020).

[94] Y. R. Chen, A. Rezapour, W. G. Tzeng, and S. C. Tsai, “RL-Routing: An SDN Routing Algorithm Based on Deep Reinforcement Learning,” *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 3185–3199, Oct. 2020, doi: 10.1109/TNSE.2020.3017751.

[95] X. Chang, F. Nie, S. Wang, Y. Yang, X. Zhou, and C. Zhang, “Compound rank-k projections for bilinear analysis,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 27, no. 7, pp. 1502–1513, Jul. 2016, doi: 10.1109/TNNLS.2015.2441735.

[96] D. Yuan, X. Chang, P. Y. Huang, Q. Liu, and Z. He, “Self-Supervised Deep Correlation Tracking,” *IEEE Trans. Image Process.*, vol. 30, pp. 976–985, 2021, doi: 10.1109/TIP.2020.3037518.

[97] SIG, “Bluetooth Technology,” *Faq*, 2013. https://www.bluetooth.com/.

[98] Zigbee Alliance, “ZigBee,” 2018. https://www.zigbee.org/.

[99] LoRa Alliance^{TM}, “LoRa technology,” 2018. https://lora-alliance.org/.

[100] LoRa Alliance^{TM}, “LoRaWAN,” 2018. https://lora-alliance.org/about-lorawan.

[101] Nordic Semiconductor, “Thingy:52 - IoT Sensor Kit Bluetooth 5.0.” 2018, [Online]. Available: https://www.nordicsemi.com/eng/Products/Nordic-Thingy-52.

[102] Microchip, “Smart Human Project,” 2018. http://smarthumanproject.com/.

[103] Microchip, “Smarthuman Air Quality Monitor.” Accessed: Nov. 09, 2018. [Online]. Available: http://microchip.com.ar/masters/wp-content/uploads/2017/08/smarthuman_brochure.pdf.

[104] A. Javed, *Building Arduino Projects for the Internet of Things*. Berkeley, CA: Apress, 2016.

[105] World Health Organization, “Ambient (outdoor) air quality and health,” 2018. http://www.who.int/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health.

[106] ProOxygen, “global warming update,” 2018. https://www.co2.earth/.

[107] R. Diestel, *Graph Theory*. Springer-Verlag New York, 2000.

[108] B. A. Galler and M. J. Fisher, “An improved equivalence algorithm,” *Commun.*

ACM, vol. 7, no. 5, pp. 301–303, 1964, doi: 10.1145/364099.364331.

[109] "Linear Sweep and Cyclic Voltametry: The Principles | Department of Chemical Engineering and Biotechnology." https://www.ceb.cam.ac.uk/research/groups/rg-eme/Edu/linear-sweep-and-cyclic-voltametry-the-principles (accessed Dec. 16, 2019).

[110] Arduino, "Accessories and devices of reduced board." https://www.arduino.cc/ (accessed Oct. 10, 2019).

[111] T. Instruments, "LMP91000 Sensor AFE System: Configurable AFE Potentiostat for Low-Power Chemical-Sensing Applications," p. 36, 2014, [Online]. Available: http://www.ti.com/lit/ds/symlink/lmp91000.pdf.

[112] Friends-of-Fritzing foundation, "Fritzing Fritzing," *fritzing.org*, 2017. https://fritzing.org/home/%0Ahttp://fritzing.org/home/.

[113] "MQTT Broker on Android | How To Run MQTT Broker in Android." https://iotbyhvm.ooo/mqtt-broker-on-android-how-to-run-mqtt-broker-in-android/ (accessed Oct. 01, 2019).