Computer Networks 186 (2021) 107745

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

IoT-networks group-based model that uses AI for workgroup allocation

Pedro Luis González Ramírez [a,b], Jaime Lloret [b,*], Jesús Tomás [b], Mikel Hurtado [a]

[a] Departamento de Ingeniería Electrónica, Universidad Central, Bogotá D.C, Colombia
[b] Instituto de Investigación para la Gestión Integrada de zonas Costeras, Universitat Politècnica de València, Valencia, España

ARTICLE INFO

Keywords:
IoT
IoT-smart architecture
IoT-gateway
M2M protocols
ML classifiers
Collaborative workgroups
Network model
Graph theory
Smart IoT-networks

ABSTRACT

This paper presents a centralized management architecture model for designing workgroup-based Internet of Things (IoT) and Internet of Everything (IoE) networks. The architecture establishes the organization of an object according to its functions and capacities in layers. From its model, it is derived the design of the algorithms that give the network operation. These algorithms include the multi-protocol communication and interconnectivity algorithm, the routing algorithm, the resource sharing algorithm, and the grouping algorithm, all controlled by Artificial Intelligence (AI). The grouping algorithm consists of creating collaborative workgroups based on Machine Learning (ML) techniques that use the objects' features to allocating these within a workgroup that attends a type of service and within an architecture layer according to its capabilities. The model was tested with a simulation that shows the Machine-to-Machine (M2M) interaction between the devices involved in providing a service to a user within a Smart Home. This simulation uses an AI hosted within an IoT-Gateway to collect data on the features that define a connected object's functions and services. The extraction of the features is done using the Discovery of Functions and Services Protocol (DFSP) transported through an IoT-Protocol. With this information, the AI assigns a layer and a workgroup to a new object when it enters the network. The result of these tests can be used to know which ML technique has better accuracy.

1. Introduction

The Artificial Intelligence (AI) techniques have gained high relevance in recent years since it can be applied in fields previously unthinkable. After the information age we live in, we may move on to the AI age. This revolution has been possible because Big Data has made vast amounts of information available to Machine Learning (ML) techniques [1]. On the other hand, the network's analysis and modeling through mathematical theories allow a better understanding of its operation while allowing us efficient algorithms or even reaching the optimal one. Today, the mathematical models of networks are of great importance when considering algorithms and traffic through numbers, allowing ML algorithms to analyze these numbers and learn the patterns that will infer how they will behave in the future. For this purpose, this paper studies the different mathematical theories used in computing and implements the most appropriate ones applied to communication networks. The Internet of Things (IoT) is an excellent scenario to apply these new technologies and integrate new concepts that will solve autonomy problems in these networks [2].

The most widely accepted concept defines IoT-Networks as the interconnection of everyday "things or objects" to the Internet [3]. In this sense, a local IoT-Network can intercommunicate "objects" through an IoT-Gateway between them and the Internet. Most of them differ in communication technologies and protocols, and not all are of the same type. Whereby some models have been proposed to resolve the interoperability of these networks. However, there is still no standard, and it is still possible to propose new alternatives.

In order to achieve interoperability between objects with different types of IoT-Protocols on the same network, it is needed a control algorithm to act as an intermediary. It is the function that AI performs in the management layer of the architecture within the IoT-Gateway. Some of these IoT-Protocols are Machine-to-Machine (M2M) [4] used to monitor and control objects. However, by incorporating ML techniques [5] into algorithms, we can achieve the desired autonomy, although we still have to explore more techniques that allow the same machines to communicate autonomously. Therefore, the idea is to leverage and modify an M2M protocol [6] and include an autonomous ML controller that performs the task of making the network interoperable. This

* Jaime Lloret
E-mail addresses: pgonzalezr1@ucentral.edu.co (P.L. González Ramírez), jlloret@dcom.upv.es (J. Lloret), jtomas@upv.es (J. Tomás), mhurtadom1@ucentral.edu.co (M. Hurtado).

https://doi.org/10.1016/j.comnet.2020.107745
Received 9 October 2020; Received in revised form 29 November 2020; Accepted 14 December 2020
Available online 16 December 2020

controller's core can be of any type, from a classifier based on neural networks [7] to a deep learning system [8].

The benefits of objects working in groups allow a user to receive a service automatically without worrying about keeping control of the activity. For example, in their home, a group of things would maintain air quality at normal levels and activate an emergency system when detecting any danger to their health. Another case could be when someone visits their home, and the door will announce the visit, and it will be shown on the screen closest to the user who is at the door. In any case, the system will try to solve the problem locally before accessing the Internet for any consultation, alarm, or follow up of the user when they are away from home through the cloud connection. Many variants of these services would be monitored by the IA, continuously learning from their user's activities, and providing increasingly automated services.

This paper proposes a network model for creating groups in an IoT-Network using ML. This model describes the possible relations that a connected object can have when it assumes a role (e.g., Thing, Sensor) and shares resources, functions, and services in common. The ML learns these patterns and establishes relations by creating workgroups, turning it into a collaborative proximity network. Possible application scenarios are proximity networks in homes, offices, industry, and even smart cities [9]. All interconnected through the cloud to provide a service to the same user [10], inside or outside their home.

1.1. Motivation

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.

1.2. Goals

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. 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.

1.3. Contributions

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 [11], 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.

This paper is structured as follows. Section 2 includes the related work about groups, architectures, and machine learning. Section 3 formulates the problem and presents the proposal and the network model. Section 4 presents the implemented algorithms and the time complexity. In Section 5, the performance test, the devices' implementation, and the simulations together with the testbed results. Finally, in Section 6, we highlight our conclusion and future work.

2. Related work

In this section, it presents the revision of works related to IoT-Networks group-based and ML. There are already many group-based network models proposed for IoT or sensor networks in the literature. However, to highlight this work's unique contribution, it is necessary to study and analyze its operation mechanics. Therefore, it is important to review which works include the use of ML within its algorithms and what methods had used, and in this way, to know what the evolution of IoT-Networks has been in the inclusion of AI to create collaborative workgroups. The following are the most relevant selected works that show the techniques involved in making grouping, modeling, and simulation IoT-Networks and ML.

D. Kimovski, H. Ijaz, N. Saurabh, and R. Prodan [12] propose an architecture called SmartFog, bio-inspired by the human brain. Its structure describes through three layers: Cloud Layer, Fog Layer, IoT Layer. In the Fog layer, the devices mimic neurons' function, and synapses correlate with communication channels, and in the IoT layer, the devices and sensors represent the sensory nervous system. The groups are formed autonomously from a set of fog devices called gateways, using machine learning algorithms. For the selection of communication gateways or neurons, classification algorithms are used considering multiple criteria to group by the same type of function. In this way, clusters with similar functionalities are generated that will execute the works and tasks requested by the sensor network or from applications in the cloud.

M. Garcia [13] has demonstrated through simulations that making groups of nodes in a Wireless Sensor Networks (WSN) improve its performance. The author has proposed creating an architecture based on groups, making groups of nodes with the same function. A group has a central node that identifies the group and delimits the group, and each node has its identifier. Therefore, the proposed architecture allows collaboration between groups. Finally, the author concludes that collaborative work between nodes group improves performance and average delay, avoids information forwarding, and saves energy.

C. Napoli, G. Pappalardo, E. Tramontana et al. [14] describe some neural network methods used to obtain a classifier that analyzes a company's employees through workgroups. The data used on their professional skills and attitudes are processed, and the resulting information indicates how the groups should be created to be more efficient. To process the data, they used a classifier based on a probabilistic neural network, which creates workgroups based on the type of relations generated between employees in terms of collaboration. In this way, this tool allows employees to be evaluated more fairly and quickly. From the

results, the authors conclude that the classifier's accuracy is very high, which allows demonstrating that the proposal based on a bio-inspired classification system is viable.

B. Yao, X. Liu, W. Zhang et al. [15] explains the use of graph theory as a mathematical tool to describe and redefine IoT. The authors also show IoT as the union and grouping of topological and data-functional networks. However, this proposal presents some challenges when using graph theory in IoT-Networks. Two of these are presented in the paper's conclusions and are related to the networks' topologies and balanced sets.

K. Batool and M. A. Niazi [16] present a modeling methodology based on a combination of agents and complex networks for IoT. This technique uses an agent-based cognitive computing (CABC) framework to simulate these networks. The IoT is considered a complex network system because of its rapid growth, and through this tool, it is possible to model its structure. Additionally, the authors demonstrate the proposal's effectiveness through a case study where testing a new algorithm and the devices' energy consumption monitoring on an IoT-Network.

G. Fortino et al. [17] describe a model IoT-Networks through agents. This system uses the Agent-based Cooperative Smart Object (ACOSO) model. Where each Smart Object or Smart Thing in an IoT-Network is described in this model under the name 'agent.' Therefore, it analyzes its features, relations, and communication level. Authors ensure that this proposal is an excellent option to model IoT-Networks since there is no already system for modeling and simulation. The simulation tests used the OMNET ++ platform to evaluate and analyze possible communication bottlenecks.

H. Yu and X. Xia [18] present the study of a problem in a network with multiple agents and a leader. This problem consists of achieving consensus coordination for dynamic agent networks. It is because multi-agent systems have applications in many areas, including control and robotics. However, its use in this article has extended to the area of dynamic networks. Authors show the design of an adaptive method based on the theory of graphics and Lyapunov's theory through an algorithm. Finally, they provide the results obtained from different tests.

Some of the previous proposals show the techniques for conforming groups of nodes on different architectures and selecting a leader. However, the proposal [12] groups Gateways into a Fog layer of the proposed architecture using ML to select the best Gateway (leader) from each group to connect to the cloud layer. In our proposal, layer four groups the objects of the lower layers of the architecture using AI (only leader). The IoT layer used in most architectures is separated in this proposal into three layers. In this way, the functionalities are separated and kept as a stand-alone system. Furthermore, it is easier to classify groups and to do centralized management.

3. Proposal description

3.1. Problem formulation

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.

3.2. Architecture proposal

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.

3.3. Architecture with centralized management

This IoT-Architecture with AI (IoT-SmartArchitecture) and centralized management encompasses the most important aspects of Cloud, Edge [19] and Fog computing through SmartFog [12] and integrates them with other architectures to give place to smart objects in a network IoT. It is an architecture with additional functionalities that allows, through the fusion of different architectures [20], 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 an 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 Fig. 1, 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.

Layer 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 [21,22].

Layer 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 [23]. 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 [23], 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.

Layer 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

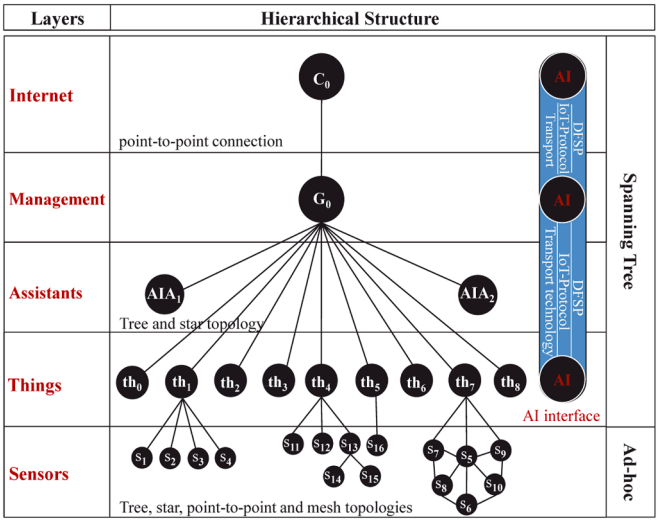

Fig. 1. IoT Stack SmartArchitecture.

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.

Layer 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.

Layer 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 [24].

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.

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.

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.

3.4. 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).

3.5. Network model

Let U be the universe of Smart IoT-Networks (SmartHome, SmartOffice, SmartFactories, SmartCity), connected through clouds. According to the design of Fig. 2, 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, λ, E) be a network of connected objects (shown in Fig. 2), 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 Fig. 1).

Fig. 3 shows the connected objects according to the network shown in Fig. 2. 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

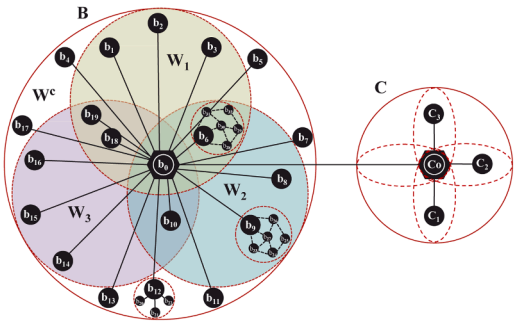

Fig. 3. IoT-Network model workgroups-based.

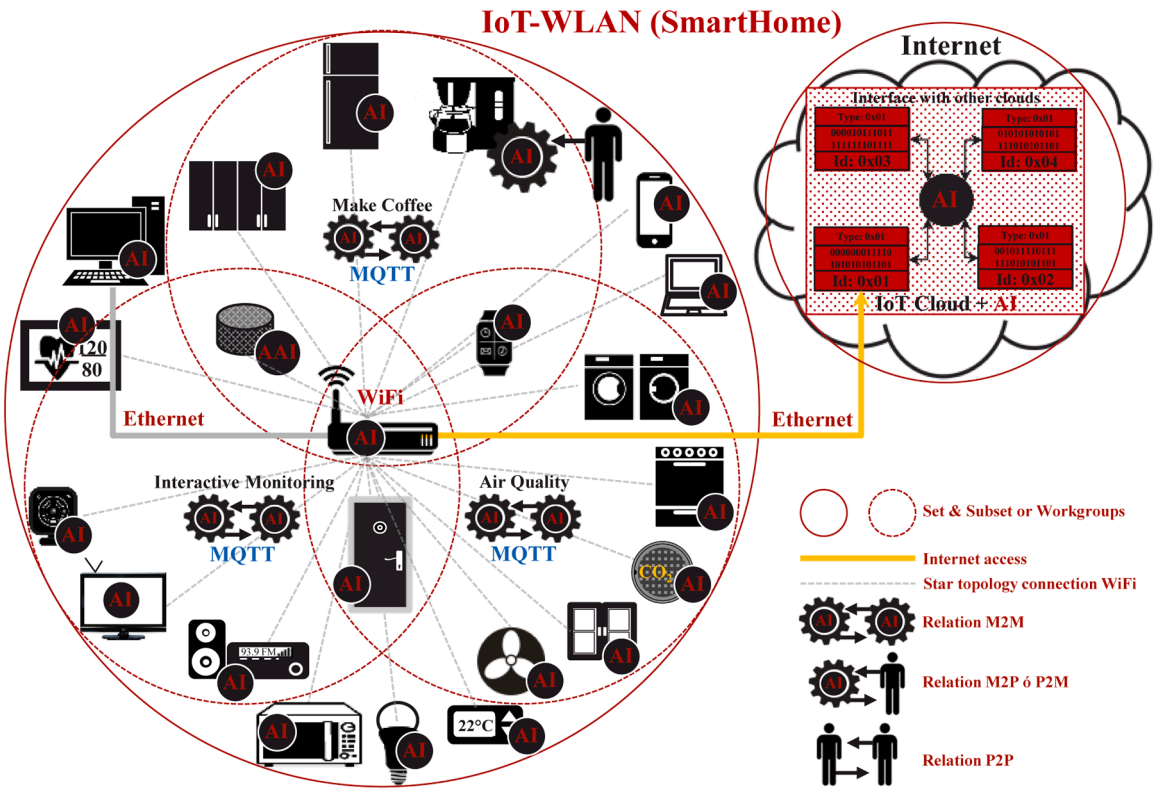

Fig. 2. Architecture of smart IoE proximity networks with centralized management.

workgroup and a layer (l_i). 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 Eq. 1, where m is the total number of subsets.

$$W = \left(\bigcup_{i=1}^{m} W_i \right) \cup W^C \tag{1}$$

It can also be noted that:

$$n + 1 = \left| W^C \right| + \left| \bigcup_{i=1}^{m} W_i \right| \tag{2}$$

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

$$\bigcap_{i=1}^{m} W_i \geq 1 \tag{3}$$

1 Initial working conditions are pre-established from the design shown in Fig. 2. It assumes that there are already three workgroups and that the objects are organized in the architecture shown in Fig. 1. Under these conditions, it is possible to characterize each of the objects and structure a dataset used to find the best classifier.

In Fig. 3, 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_i, G_i, AIA_i, Th_i, or S_i 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: $b_i \in W = \{b_1, b_2, b_3, \ldots, b_n\}$.
- feature: $f_i \in F = \{ f_1, f_2, f_3, \ldots, f_n\}$.
- function 1: $f(W) = \{ f: W \rightarrow \{0, 1\}\}$.
  - Register of features in a dataset:
- class:$\mathbf{C}_j \in \mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \ldots, \mathbf{C}_n\}$.
- dataset: $\mathbf{d}_i \in D = \{d1, d2, d3, \ldots, dn\}$, where $\mathbf{d}_i = \{ f_1(b_i), f_2(b_i), f_j(b_i), \ldots, f_n(b_i), \mathbf{C}_j\}$.
- function 2: $f(D) = \{ f: D \rightarrow \mathbf{C}\}$, such that each $\mathbf{d}_i$ is assigned to a class $\mathbf{C}_j$.

F is defined as a finite set of features related to a node and predefined in a dataset structure. Therefore features of a node (d_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 $\mathbf{C}_j$ 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: $D \rightarrow \mathbf{C}$ such that each $f_j(b_i)$ is assigned to a $\mathbf{C}_j$ class.

f: $D \rightarrow \mathbf{C}$ 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 $y = [\mathbf{C}_j]$. Therefore, the register could be rewritten as $d_i = \{(X_1, \mathbf{C}_1), \ldots, (X_i, \mathbf{C}_j), \ldots, (X_n, \mathbf{C}_n)\}$.

In a K-NN classifier, dataset D_X^K, has been classified previously utilizing like training matrix (X_{Tn}, $\mathbf{C}_{Tn}$) 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.

Fig. 7 shows how some nodes of Fig. 4 would look organized by groups and layers.

3.6. Algorithms implementation

Below are the algorithms implemented in the simulations and the time to compute it. Each algorithm and procedure preexisting in [25, 26], 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 [26]. This running time is shown in Table 1.

For the proper run of the algorithm, it is necessary to consider that there is a dataset structure with four feature profiles (Xp) in the IoT-Gateway (G0). 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.

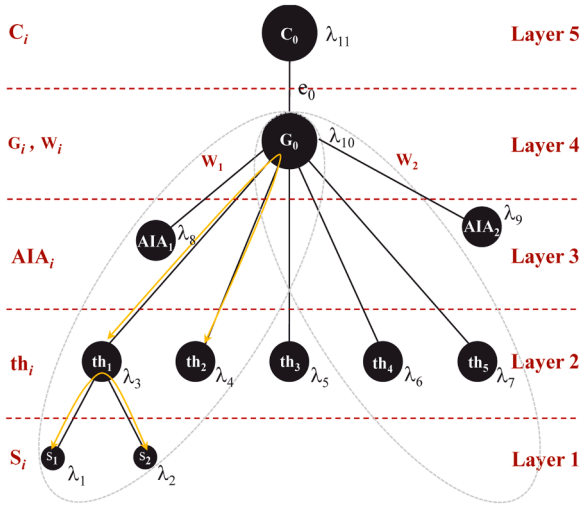

Fig. 4. IoT-Network model workgroups-based.

Table 1
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|·|F|) + O(|**C** |·|F|)

The total structure of a dataset register would be as follows, di = {(X_1, **C** 1), (X_2, **C** 2), (X_3, **C** 3), (X_4, **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: d_i = {(X_1, **C** 1), (X_2, **C** 2)} and for routing the last two segments defined like this d_i = {(X_3, **C** 3), (X_4, **C** 4)}. The class label for **C** 1 is "workgroups", and for**C** 2 it is "layer".

Fig. 5 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.

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 (**C** 1) and in an architecture layer (**C** 2). That is, the input values of the classes function are initially set by default. The values of **C** 1 are set as {1, 2, 3} representing {w_1, w_2, w_3} respectively. The **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 2 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 3 indicates the parameters' notation be used in the algorithm and its corresponding meaning.

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."

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 characteristics in a working group (Algorithm 2) and an architecture layer (Algorithm 3). Then, it is updating the new information in the dataset (Algorithm 1) and stores it.

Algorithm 1 updates the dataset's information hosted in G_0 when a new node (bi) 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.

Table 2
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

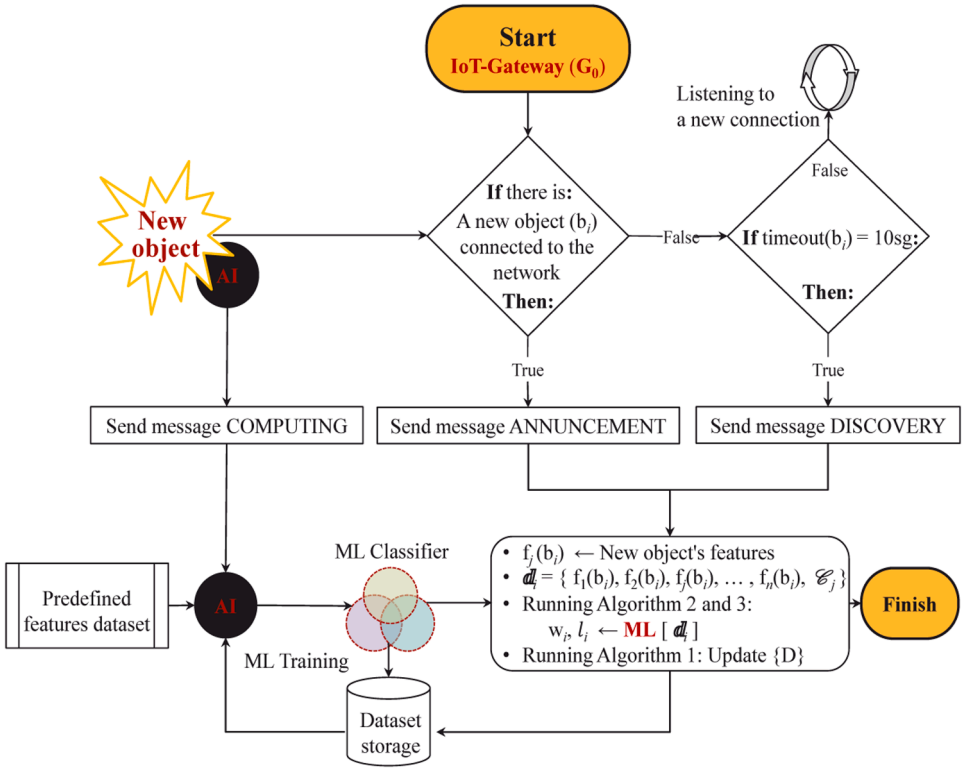

Fig. 5. Flowchart of the ML classifier.

Table 3
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.

Algorithm 1
Updating dataset {D}.

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

Output: Updated dataset for two feature profiles {D}.

Algorithm 2 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 bi and assign it to a workgroup (wi).

Algorithm 3, like Algorithm 2, 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 2, except that use class $\mathbf{C}_2$. In step 16, the algorithm renames the identifier of the object

Algorithm 2
Creating collaborative workgroups (AI).

Input: dataset { D}, d_i
Process:
1. Read and load to { D}.
2. X ← [$X_1 + X_2$].
3. y ← [$\mathbf{C}_1$].
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 3
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 ← [$\mathbf{C}_2$].
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_i
19. **case** 2:
20. Rename(b_i) ← th_i
21. **case** 3:
22. Rename(b_i) ← AIA_i
23. **case** 4:
24. Rename(b_i) ← G_i
25. **End Switch**
26. **End while.**

Output: New node assigned to a layer (l).

according to the layer.

Figure 6 shows the DFSP datagram and the messages used in Algorithm 1 to update the feature dataset.

4. Performance test

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. 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) [21]. 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 [27], to implement a sensor.

4.2. 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

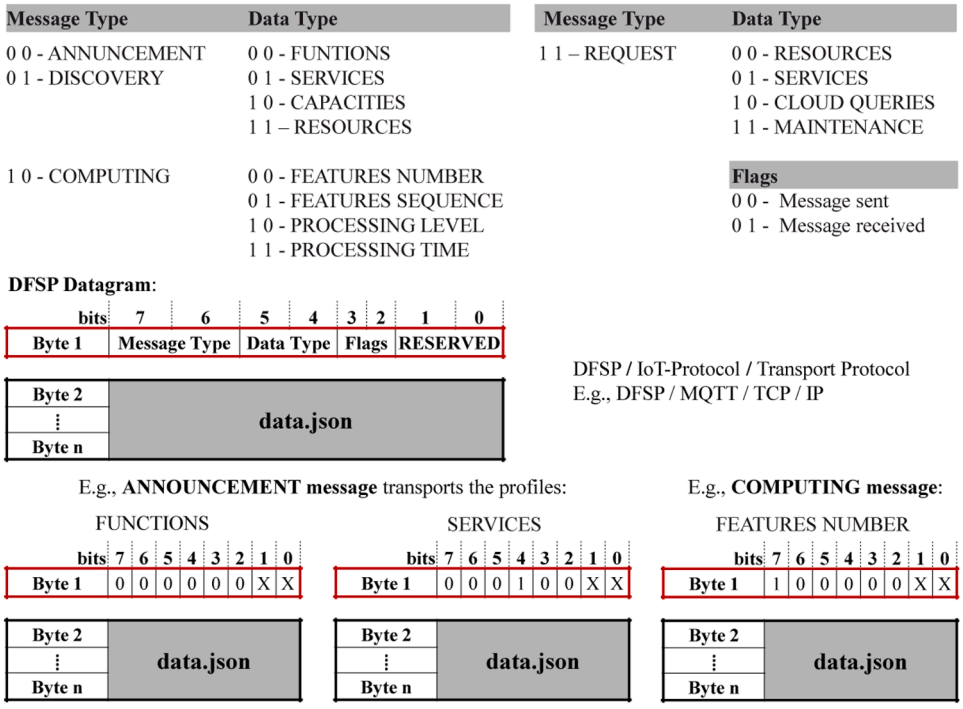

Fig. 6. DFSP Messages.

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) [28] 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.

4.3. Simulators

The simulation was carried out with several simulators such as Cisco Packet Tracer 7.3, Jupyter [25], and iFogSim [29,30]. 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 preinstalled 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 1 and Table 3 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 [26], together with other important libraries for data preprocessing.

4.4. 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 Fig. 7 is labeled with W_i, representing its participation in each group. With this scenario, the initial data for ML training is collected.

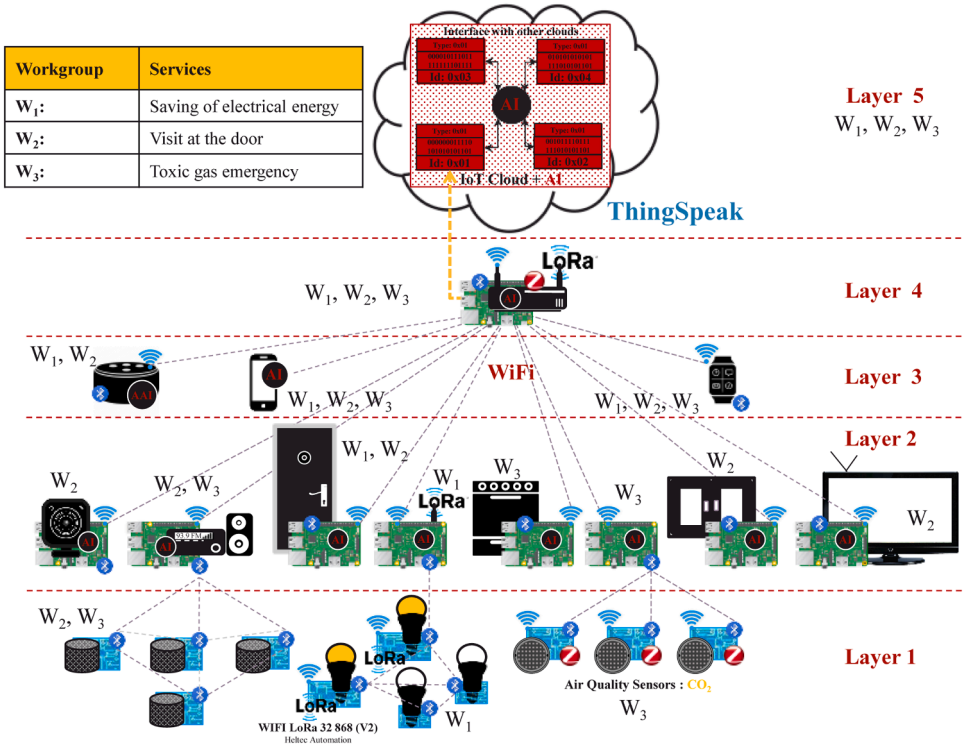

Fig. 7. Simulation scenario organized by groups and layers.

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.

Fig. 8 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%.

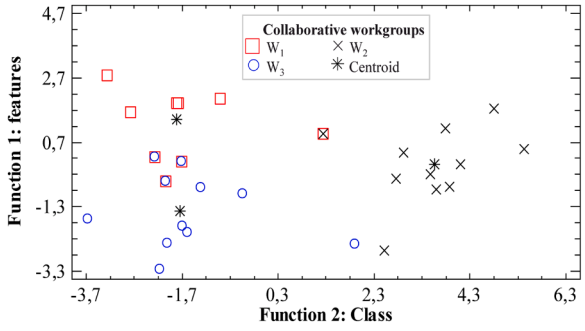

Fig. 8. Discriminant Analysis.

Fig. 8 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 Fig. 8 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. Fig. 9 shows that with values K = 3, greater precision is achieved, so it is selected for all tests.

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."

4.5. 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"

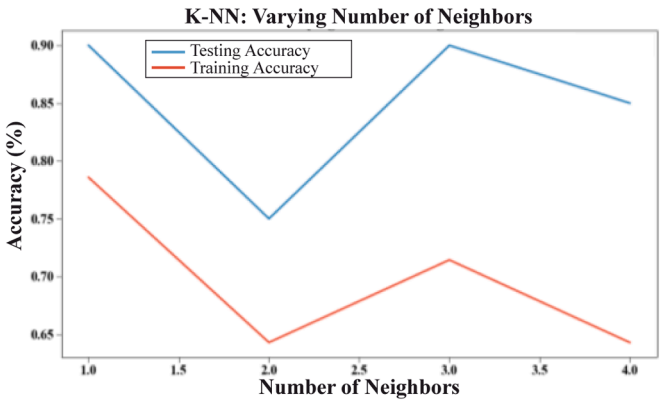

Fig. 9. Selection the best value of K.

class. On the other hand, with the "layer" class, each parameter's best result give using "StandardScaler."

Fig. 10 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 Fig. 10 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.

Fig. 10 compares the results of different distance metrics for each classification label. It is observed in the results that the accuracy $\geq$ 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.

Fig. 11 shows the error in the previous tests. Results show very low error for most of the distances evaluated.

As it is observed in Fig. 11 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 Fig. 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

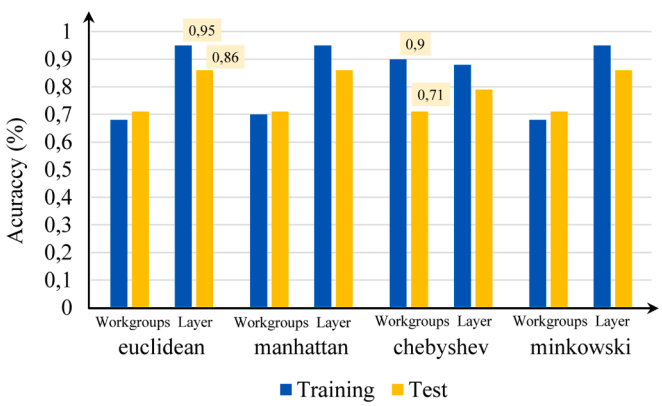

Fig. 10. Comparison of distance metric in a KNN classifier (K=3).

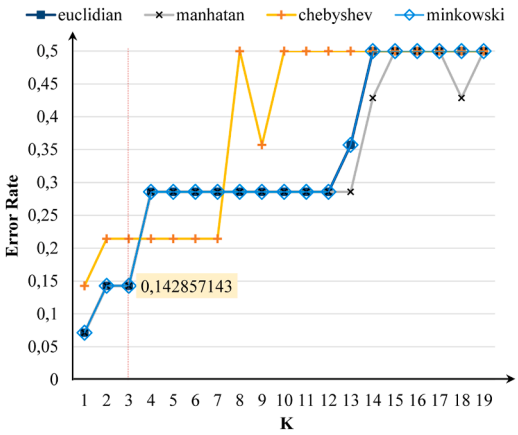

Fig. 11. Comparison of Error Rate in a KNN classifier (K=3).

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.

Fig. 12 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.

Fig. 13 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.

Fig. 14 shows a Neighborhood Components Analysis (NCA). It tries to find a feature space to improve accuracy.

In any case, none of the three figures shows a clustering of the data that is visually meaningful, like in HYPERLINK \l "fig0008" Fig. 8.

The results of these reduction techniques PCA, LDA, and NCA applied to the K-NN model are compared with those obtained in Fig. 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

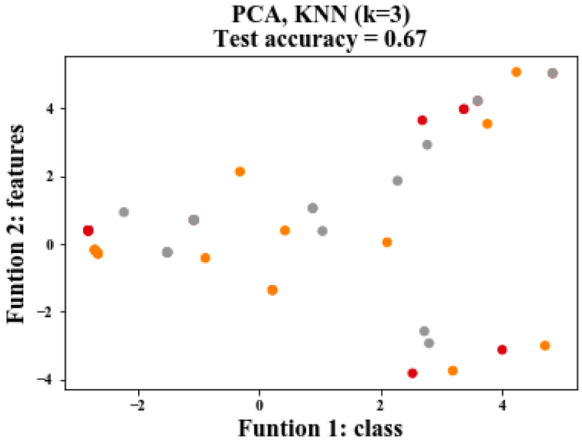

Fig. 12. PCA, K-NN (K=3) Test accuracy = 0,67.

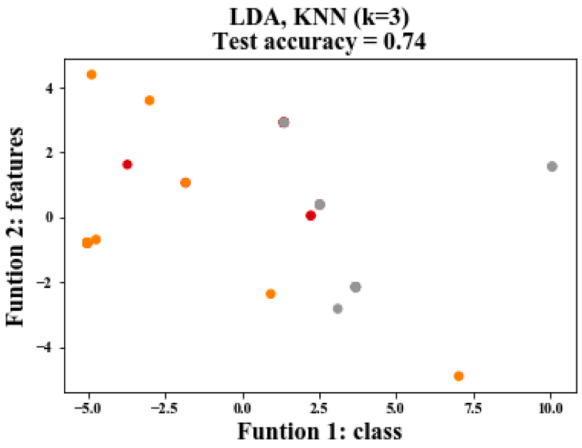

Fig. 13. LDA, K-NN (K=3) Test accuracy = 0,74.

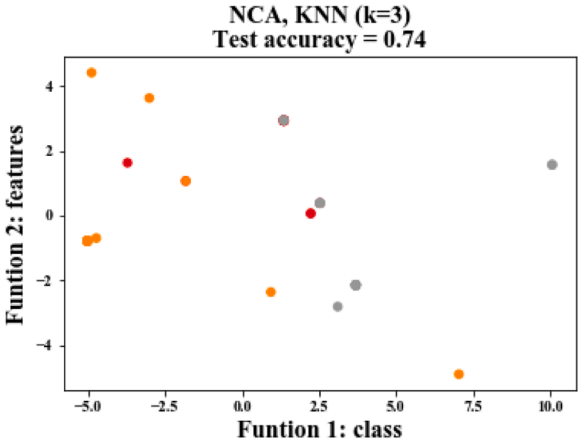

Fig. 14. NCA, K-NN (K=3) Test accuracy = 0,74.

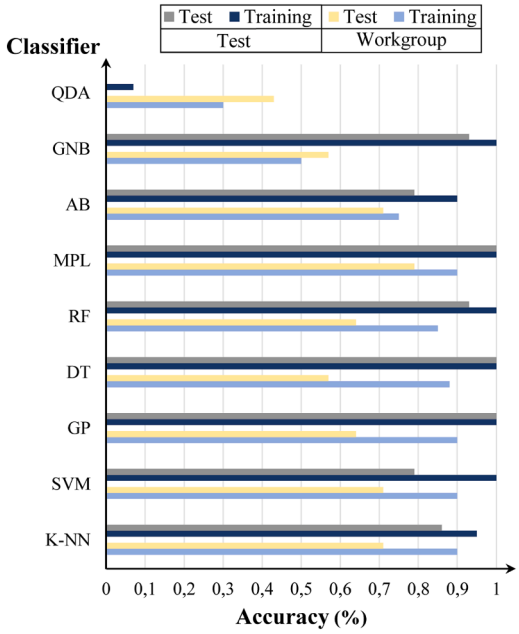

Fig. 15. Comparison of accuracy with different classifier.

use different techniques and metrics. Fig. 15 shows a comparative graph with the other classifiers.

Fig. 15 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.

Fig. 16 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.

Fig. 17 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.

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 data sets, which may be biased, so cross-validation is a better approximation. The difference between some values compared with those obtained in Fig. 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.

5. Conclusion

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 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

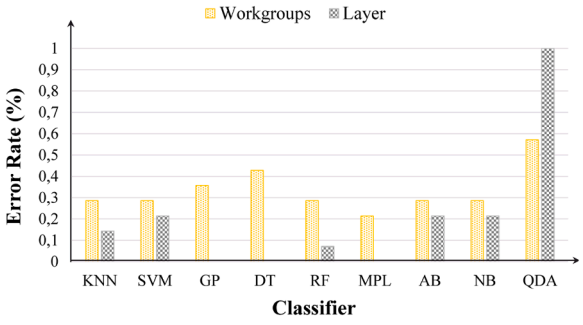

Fig. 16. Error Rate's Comparison each classifier.

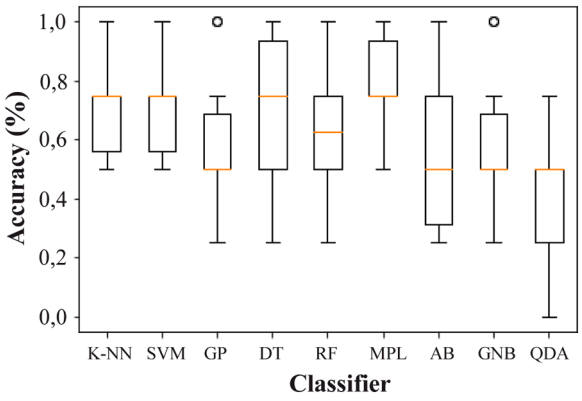

Fig. 17. Comparison of models with the cross-validation technique.

the clustering algorithm used in this simulation and observing its behavior when a new object joins the network. Thus, larger input data sets 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.

Declaration of Competing Interest

None.

Acknowledgments

This work has been partially supported by the "Ministerio de Economía y Competitividad" in the "Programa Estatal de Fomento de la Investigación Científica y Técnica de Excelencia, Subprograma Estatal de Generación de Conocimiento" within the project under Grant TIN2017-84802-C2-1-P. This work has also been partially supported by European Union through the ERANETMED (Euromediterranean Cooperation through ERANET joint activities and beyond) project ERANETMED3-227 SMARTWATIR. At the Univeridad Central (Colombia), to the Smart City work team of the MAXWELL research group, for their special interest in generating new contributions to the networks (IoT & IoE).

References

[1] M. Chen, U. Challita, W. Saad, C. Yin, M. Debbah, Artificial neural networks-based machine learning for wireless networks: a tutorial, IEEE Commun. Surv. Tutorials PP (c) (2019) 1, https://doi.org/10.1109/COMST.2019.2926625.

[2] K.A.P. da Costa, J.P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of Things: a survey on machine learning-based intrusion detection approaches, Comput. Netw. 151 (Mar.) (2019) 147–157, https://doi.org/10.1016/j.comnet.2019.01.023.

[3] Telecommunication Standardization Sector Of ITU, "Recommendation ITU-T Y.2060," Overview of the Internet of things, 2012. [Online]. Available: https://www.itu.int/itu-t/recommendations/rec.aspx?rec=Y.2060. [Accessed: 10-Jun-2020].

[4] V. Gazis, A survey of standards for machine-to-machine and the internet of things, IEEE Commun. Surv. Tut. 19 (1) (2017) 482–511, https://doi.org/10.1109/COMST.2016.2592948. Institute of Electrical and Electronics Engineers Inc.

[5] J. Serra, L. Sanabria-Russo, D. Pubill, C. Verikoukis, Scalable and flexible IoT data analytics: when machine learning meets SDN and virtualization, in: 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2018, pp. 1–6, https://doi.org/10.1109/CAMAD.2018.8514997, 2018-Septe.

[6] ETSI, "Machine-to-Machine communications (M2M) - functional architecture technical specification," TS 102 690 V2.1.1, 2013.

[7] F.A. Gomide, Redes neurais artificiais para engenharia e ciências aplicadas: curso prático, Sba Control. Automação Soc. Bras. Autom. 23 (5) (2012) 649–652, https://doi.org/10.1590/s0103-17592012000500011.

[8] S. Haykin, Neural networks and learning machines third edition, vol. 40, no. 6. 2001.

[9] K. Pardini, J.J.P.C. Rodrigues, O. Diallo, A.K. Das, V.H.C. de Albuquerque, S. A. Kozlov, A smart waste management solution geared towards citizens, Sensors (Switzerland) 20 (Apr. (8)) (2020) 2380, https://doi.org/10.3390/s20082380.

[10] P.L. Gonzalez Ramirez, J. Lloret, M. Taha, 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, https://doi.org/10.1109/CSCI46756.2018.00193.

[11] M.M. Hassan, M.R. Hassan, S. Huda, V.H.C. de Albuquerque, A robust deep learning enabled trust-boundary protection for adversarial industrial IoT environment, IEEE Internet Things J. 8 (8) (2020) 1, https://doi.org/10.1109/JIOT.2020.3019225. –1.

[12] D. Kimovski, H. Ijaz, N. Saurabh, 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, https://doi.org/10.1109/CFEC.2018.8358723.

[13] G.P. Miguel, A Group-based Architecture and Protocol for Wireless Sensor Networks, Universitat Politècnica de València UPV, 2013.

[14] C. Napoli, G. Pappalardo, E. Tramontana, R.K. Nowicki, J.T. Starczewski, M. Woźniak, Toward work groups classification based on probabilistic neural network approach, Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science) 9119 (2015) 79–89, https://doi.org/10.1007/978-3-319-19324-3_8.

[15] B. Yao, et al., Applying graph theory to the internet of things, in: Proceedings - 2013 IEEE International Conference on High Performance Computing and Communications, HPCC 2013 and 2013 IEEE International Conference on Embedded and Ubiquitous Computing, EUC, Zhangjiajie, 2013, pp. 2354–2361, https://doi.org/10.1109/HPCC.and.EUC.2013.339, 2014.

[16] K. Batool, M.A. Niazi, Modeling the internet of things: a hybrid modeling approach using complex networks and agent-based models, Complex Adapt. Syst. Model. 5 (1) (2017) 1–19, https://doi.org/10.1186/s40294-017-0043-1.

[17] G. Fortino, W. Russo, C. Savaglio, Agent-oriented modeling and simulation of IoT networks, in: Proc. 2016 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2016 8, 2016, pp. 1449–1452, https://doi.org/10.15439/2016F359.

[18] H. Yu, X. Xia, Adaptive consensus of multi-agents in networks with jointly connected topologies, Automatica 48 (8) (2012) 1783–1790, https://doi.org/10.1016/j.automatica.2012.05.068.

[19] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, C. Verikoukis, Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture, IEEE Internet Things J. 7 (May (5)) (2020) 4183–4194, https://doi.org/10.1109/JIOT.2019.2944695.

[20] M.G. dos Santos, D. Ameyed, F. Petrillo, F. Jaafar, and M. Cheriet, "Internet of things architectures: a comparative study," 2020.

[21] J. Lloret, S. Sendra, P.L. González, L. Parra, An IoT group-based protocol for smart city interconnection, Commun. Comput. Inf. Sci. 978 (2019) 164–178, https://doi.org/10.1007/978-3-030-12804-3_13.

[22] J. Lloret, M. Garcia, J. Tomas, J.J.P.C. Rodrigues, Architecture and protocol for intercloud communication, Inf. Sci. (Ny). 258 (2014) 434–451, https://doi.org/10.1016/j.ins.2013.05.003.

[23] P.L.G. Ramirez, M. Taha, J. Lloret, J. Tomas, An intelligent algorithm for resource sharing and self-management of wireless-IoT-gateway, IEEE Access 8 (2020) 3159–3170, https://doi.org/10.1109/ACCESS.2019.2960508.

[24] P. Gonzalez, J. Lloret, J. Tomas, O. Rodriguez, M. Hurtado, IoT-WLAN proximity network for potentiostats, in: 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2020, pp. 94–99, https://doi.org/10.1109/FMEC49853.2020.9144776.

[25] Jupyter Org, "Project Jupyter | Jupyter Software," 2020. [Online]. Available: https://jupyter.org/. [Accessed: 12-Feb-2020].

[26] F. Pedregosa, et al., Scikit-learn: machine learning in Python, J. Mach. Learn. Res. 12 (85) (2011) 2825–2830.

[27] "WIFI LoRa 32 (V2) – Heltec automation," 2019. [Online]. Available: https://heltec.org/project/wifi-lora-32/. [Accessed: 11-Mar-2020].

[28] 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.

[29] H. Gupta, A.V. Dastjerdi, S.K. Ghosh, 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. 47 (Jun. (9)) (2016) 1275–1296, https://doi.org/10.1002/spe.2509.

[30] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, H. Ijaz, A job scheduling algorithm for delay and performance optimization in fog computing, Concurr. Comput. Pract. Exp. 32 (Apr. (7)) (2020), https://doi.org/10.1002/cpe.5581.

PEDRO LUIS GONZÁLEZ RAMÍREZ received a degree in Electronic Engineering at Universidad de Pamplona, Colombia, in 2000. M.Sc.-Ing degree in Electronic Engineering at Pontifícia Universidad Javeriana, Colombia in 2012. Currently Ph.D. student in Telecommunication at the Polytechnic University of Valencia (Spain). He is a Cisco Certified Network Professional Instructor. He has a special interest in android mobile application development, research in Telecommunications Networks, Internet of Things (IoT), and Internet of Everything (IoE), design and implementation of Architectures, Protocols, and Algorithms for IoT networks. He has been a paper reviewer and session chair at some international conferences and is a Lecturer with the Department of Electronic Engineer, Universidad Central, Colombia. He is also a member of the Research Group Maxwell of Universidad Central and a Junior Researcher in MinCiencias, Colombia.

JAIME LLORET (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in physics, in 1997, the B.Sc. and M.Sc. degrees in electronic engineering, in 2003, and the Ph.D. degree (Dr.-Ing.) in telecommunication engineering, in 2006. He worked as a Network Designer and an Administrator in several enterprises. He is currently an Associate Professor with the Polytechnic University of Valencia. He is a Cisco Certified Network Professional Instructor. He is the Chair of the Integrated Management Coastal Research Institute (IGIC), and he is the Head of the Active and Collaborative Techniques and Use of Technologic Resources in the Education (EITACURTE) Innovation Group. He was the Director of the University Master "Digital Post Production,'' from 2012 to 2016. He is the Director of the University Diploma ''Redes y Comunicaciones de Ordenadores.'' He has authored 22 book chapters and has more than 480 research articles published in national and international conferences, and international journals (more than 220 with ISI Thomson JCR). He is an ACM Senior Member and IARIA Fellow. He was the Vice-Chair for the Europe/Africa Region of Cognitive Networks Technical Committee (IEEE Communications Society), from 2010 to 2012, and the Vice-Chair of the Internet Technical Committee (IEEE Communications Society and Internet society), from 2011 to 2013. He was the Internet Technical Committee Chair of the IEEE Communications Society and Internet society, from 2013 to 2015. He has been the General Chair (or Co- Chair) of 57 International workshops and conferences. He is currently the Chair of theWorking Group of the Standard IEEE 1907.1. Since 2016, he has been the Spanish Researcher with highest H-index in Telecommunications journal list according to Clarivate Analytics Ranking. He has been involved in more than 450 program committees of international conferences, and more than 150 organization and steering committees. He has led many local, regional, national, and European projects. He is the Editor-In-Chief of Ad Hoc and SensorWireless Networks (with ISI Thomson Impact Factor), the international journal Networks Protocols and Algorithms, and Interna- tional Journal of Multimedia Communications. Moreover, he is an advisory board member of International Journal of Distributed Sensor Networks (both with ISI Thomson Impact factor), and he is the IARIA Journals Board Chair (eight journals). Furthermore, he is (or has been) an Associate Editor of 46 international journals (16 of them with ISI Thomson Impact Factor). He has been the Co-Editor of 40 conference proceedings and a Guest Editor of several international books and journals.

JESÚS TOMÁS was graduated in Computer Science in 1993 at Polytechnic University of Valencia, getting the best ratings. He finished his Doctoral Thesis in 2003. He worked as software programmer in several enterprises and as a freelance. From 1993, he is an associate professor at Polytechnic University of Valencia. He is member of the Integrated Management Coastal Research Institute. His research focuses on statistical translation, artificial intelligence, pattern recognition and sensors networks. He has published multiple articles in national and international conferences and has multiple articles in international journals (more than 17 of these are included in the Journal Citation Report). He has been involved in several research projects related to public and private pattern recognition and artificial intelligence applied to multiple subjects (4 of them as principal investigator). He is the director of University Master Develop of mobile applications.

MIKEL FERNANDO HURTADO-MORALES: Chemist, Ms.c, Chemical Thermodynamics (Universidad Nacional de Colombia), Ms.c, Materials Science and Nanotechnology (Zaragoza University and Aragon Institute of Nanoscience), PhD, Solid State Chemistry (Universidad Nacional de Colombia and Helmholtz Zentrum Berlin), PostDoc in Nanomaterial Physics (Los Andes University and BASF). Associate Professor at Universidad Central. Group Leader SEM2 Surface, Energy and Modern Materials, Electronic Engineering Department Universidad Central Bogotá Colombia since 2014. Solid State Physics and Nanosystems, Nanomaterials and Modern Applications, and Flexible and transparent Nanoelectronics lectures. Senior Researcher by MinCiencias Colombia since 2017.