



DICOMIST: An methodology for Performing Distributed Computing in Heterogeneous *ad hoc* Networks

Alejandro Velazquez-Mena, Héctor Benitez-Perez, Rita C. Rodríguez-Martínez,
Ricardo F. Villarreal-Martínez

Alejandro Velázquez-Mena

Department of Computer Engineering
School of Engineering at UNAM, México
mena@fi-b.unam.mx

Héctor Benítez-Pérez

Institute of Research in Applied
Mathematics and Systems (IIMAS)
at UNAM, Mexico
hector.benitez@iimas.unam.mx

Rita C. Rodríguez-Martínez

Institute of Research in Applied
Mathematics and Systems (IIMAS)
at UNAM, Mexico
rita.rodriguez@iimas.unam.mx

Ricardo F. Villarreal-Martínez

Institute of Research in Applied
Mathematics and Systems (IIMAS)
at UNAM, Mexico
ricardo.villarreal@iimas.unam.mx

Abstract

The Internet of Things (*IoT*) has emerged as a cornerstone technology, transforming how we interact with our surroundings. Despite their widespread adoption, *IoT* devices encounter challenges related to processing capabilities and connectivity, frequently necessitating the delegation of tasks to remote cloud servers. This offloading, essential for enhancing user experience, poses challenges, particularly for latency-sensitive applications. Edge-centric paradigms like *fog and mist computing* have emerged to address these challenges, bringing computational resources closer to end-users. However, efficiently managing task offloading in dynamic *IoT* environments remains a complex issue. This paper introduces D**ISTRIBUTED C**OMPUTING for M**IST** (*dicomist*), a methodology designed to facilitate task offloading in *IoT* settings. *Dicomist* utilizes wireless mesh networks to organize mobile nodes, employing clustering and classification techniques. Tasks are treated as consensus problems, enabling distributed computation among selected nodes. Real-world experiments demonstrate *dicomist's* effectiveness, underscoring its potential to enhance task offloading in *IoT* environments.

Keywords: Distributed Processing, Consensus, Fog Computing, Mesh Network.

1 Introduction

The Internet of Things (*IoT*) has evolved into an indispensable technology. The significance of *IoT* devices is unparalleled in today's interconnected world, revolutionizing the way we interact with technology and the environment around us. These devices have become essential components in various industries, ranging from healthcare and agriculture to transportation and manufacturing. The use of *IoT* devices grows exponentially, devices such as smartphones, laptops, and smartwatches used by the end-users are examples of this trend. Nevertheless, the functionality of these devices is limited by factors such as battery life, storage capacity, and processing power. As a result, the tasks they handle are typically transferred to the cloud to ensure a superior quality of service for users. This process of transmitting a fraction of a local task to another device for its remote computation is called offloading, and the amount of offloaded task is called the offloading fraction [96]. Cloud computing has long been the backbone upon which *IoT* devices heavily depend. The cloud provides on-demand pay-per-use resources of servers, storage, and networks following several paradigms such as Infrastructure-as-a-Service (IaaS), Platforms-as-a-Service (PaaS), and Software-as-a-Service (SaaS). However, offloading tasks to cloud services presents hurdles for latency-sensitive applications like healthcare and fraud detection, mainly due to the physical distance between the cloud and end-user devices.

Internet of Things-based technologies with an edge-centric focus, such as fog and mist computing, provide distributed and decentralized solutions to address the limitations of cloud-centric models [97]. Fog computing represents a paradigm shift, bringing computational resources and services closer to end-user devices at the edge of the network, thus reducing latency and facilitating connection with cloud computing resources. It also lowers the device's energy consumption significantly. Unlike pure cloud computing, fog resources rely on constrained and heterogeneous nodes, which may experience unstable connectivity [95]. As the number of *IoT* devices grows, the challenge is to maintain appropriate *QoS* parameters including throughput, performance, response time, and resource utilization. Additionally, the selection of a set of fog resources and the offloading fraction can vary drastically depending on the application scenario. Thereby, the offloading process involves a series of decision-making stages. In these complex scenarios, efficient task-offloading algorithms are central to the success of *fog computing*.

Efficient task-offloading is a non-trivial task due to the nature of the *IoT* wireless networks. For instance, *fog* resources are dynamic, and wireless medium channel conditions are prone to variation over time due to fading, interference, and many other factors. Moreover, the workload on fog resources changes often due to the diversity of the applications. As a consequence, it's crucial to develop algorithms capable of dynamically adjusting network parameters, allowing smart devices to select a reliable fog resource for service provisioning. Over the last decade, significant progress has been made in the field of optimizing task offloading in *IoT* environments. Advances in machine learning, optimization techniques, and heuristic strategies have led to more sophisticated algorithms capable of efficiently managing dynamic fog resources and adapting to changing wireless channel conditions. However, there is still room for improvement. In contrast to the previously proposed algorithms, the fog node selection and task offloading scheme presented in this work is implemented in a real-life testbed.

This paper presents DIstributed COmputing for MIST (*dicomist*), a flexible methodology to perform task offloading in *IoT* environments. *Dicomist* is a general methodology since it assumes an unorganized set of mobile nodes to perform a task offloading. To achieve this end, in the initial phase, the *dicomist* deploys a wireless mesh network. Then, organize the nodes based on their capabilities following clustering and classification strategies. Subsequently, a task is modeled as a consensus problem, in this way the selected nodes calculate some function of specific parameters in a distributed manner. Once the selected nodes reach an agreement on the value of the performed task, the response is retransmitted to the originator of the request in the recovery phase. Real-life results presented in this work demonstrate the effectiveness of *dicomist*.

The remainder of the paper is organized as follows. In Section 2, we present the related works on task offloading algorithms. Section 3 details the proposed methodology through the phases. Section 4 presents experiments and results with a testbed environment and real hardware. Finally, Section 6 concludes the paper.

2 Related Work

This section describes the most relevant works developed for the *Fog Computing* paradigm and consensus scenarios. With an increased research interest in *Fog Computing*, numerous applications and data management platforms, and strictly consistent. In [68] performed the analysis of state of the art and identifying and scrutinizing all the five major domains where task allocation has a crucial role, i.e., Internet of Things (*IoT*), sensor and actuator networks (SANs), multi-robot systems (MRS), mobile crowdsensing (MCS), and unmanned aerial vehicles (UAVs). In these five domains, scenarios can be generated in the *Fog Computing* platform, where there is a relationship between tasks and devices [25], [51], and [47], the main of which are: interconnection of tasks and the same tasks are assigned to multiple devices.

The process of selecting fog nodes is both crucial and demanding. While a multitude of fog nodes may exist within the network, the primary aim of selecting fog resources is to dynamically choose the most suitable nodes that ensure low latency while being location-aware. As the quantity of *IoT* and *fog* devices grows, the complexity of the problem escalates to an NP-hard level, making conventional solution methods challenging to implement. To this end, the authors have focused on heuristic alternatives. For instance, the authors in [98] propose a bio-inspired ant colony algorithm adapted for task offloading to *fog* nodes. These *fog* nodes process tasks in a manner that optimizes average response time and distributes tasks evenly across the fog network. In [100], the authors introduced a fruit-fly-based approach for task offloading, prioritizing minimal energy consumption. This method comprises three key components: a device manager, a broker, and a task tracker. The device manager oversees the capabilities of nodes within the cloudlet, while the broker assigns tasks to nodes based on their capabilities. Subsequently, the task tracker executes the assigned tasks. In [99], a novel approach utilizing smart ant colony optimization is introduced for task offloading within fog environments. This technique effectively manages task offloading and resource utilization. The proposed algorithm considers parameters such as latency and bandwidth, although the authors have not explicitly tackled aspects related to energy, power, and cost.

When performing the tasks, the factors that impact them are *heterogeneity* [82, 86, 89], *devices resource availability* [22, 26, 71, 88, 93], *mobility* [6, 27, 73, 81], *communication range* [43, 66], and *application complexity* [75].

Consensus problems have a long history in computer science and form the foundation of distributed computing. In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of specific parameters; for example, in *IoT*, nodes in sensor networks may be tasked with calculating all the sensor's average measurement values [30]. Similar concepts include state agreement [37], rendezvous [38, 39], and observer design [50] in control theory, and gossip algorithms [17] in computer science.

In [16] represent, the *IoT* is a heterogeneous network consisting of different physical objects such as sensors, actuators, RFID, smart devices, and servers. These objects must cooperate dynamically and utilize their resources effectively and distributed, which is ideal for using consensus algorithms. Consensus algorithms are widely applied for synchronization, resource allocation, and security, and this is due to their simple execution, robustness to topology changes, and to be distribution. The Laplacian matrix is applied for r-regular and q-triangular networks.

In [67] described, most cloud-based *IoT* platform implementations rely on a virtualization layer,

which is used to implement some functions that augment the capabilities of the physical devices. Specifically to use of the concept of Virtual Object (VO), which consists of the virtual counterpart of the physical devices (i.e., the Real World Objects, RWOs) to achieve consensus-based resource allocation. In [80], work shows that the edge dynamics in the complex network may also be of great importance [49]. For example, friendships in social networks, synapses in neural networks, natural systems [30, 41, 94], cars [19, 48, 63, 90], and networked multi-agent systems [53, 70, 84] may be changing over time.

In [72] aims at solving challenges by proposing a distributed consensus algorithm for the decision-making of services at edge nodes in the service-oriented *IoT*. Specifically, a service provider framework is proposed, where services representation, discovery, detection, and composition are investigated. In [65] was a proposed consensus algorithm inspired by gossip[40] and asynchronous iteration[52] to solve the average consensus problem in *IoT* environments. The proposed protocol in [65] can deal with packet losses, delays, asynchronism, and multi-rate behavior, all of which are pervasive in any *IoT* deployment. In [64] the algorithm is implemented using gossip with the rust programming language. In the search for information today do not exist an article needs to describe *IoT* networks with infrastructure and how to solve them dynamically and temporarily.

3 The Proposed Methodology

When reviewing the literature, the consensus has yet to be addressed in the *Fog Computing* paradigm that solves various scenarios with mobile devices by bringing together various techniques applied in other domains (wireless mesh network deployment, cluster formation, graph theory applied to consensus).

The proposed Distributed Computing for MIST (*dicomist*) methodology, implements a consensus in the paradigm of *Fog Computing* where the following operations (processes, send/receive data, and perform actions) at a given time Δt through seven phases: initial, configuration, consensus, re-configuration, distribution, execution, and recovery. Figure 1 shows the phases of the *dicomist* time algorithm, which is executed in a Δt time, while Figure 2 shows the flow diagram of the methodology.

The main contribution of *Dicomist* is to have a flexible methodology for distributed processing about the *Fog Computing* paradigm through a set of nodes to solve complex tasks. The algorithm consists of seven phases. The initial phase solves the problem of the set nodes with a temporary network. Once the network generates, the scenario has many nodes because one person has several mobile devices [9]. Considering the cluster formation (second phase), the convenience in the consensus (fourth phase) is that they converge quickly due to the energy limitation due to the data communication of the above with all the nodes. The nodes classification (third phase) to compute the complex tasks. The graph theory for fast convergence in small networks is applied in the consensus when it has clusters [16].

Some factors to consider when executing the *dicomist* methodology:

- The network configuration does not exist originally.
- To suppose that the nodes are heterogeneous.
- The time interval is known, so there is a deadline such as Δt .
- Latency and location awareness.

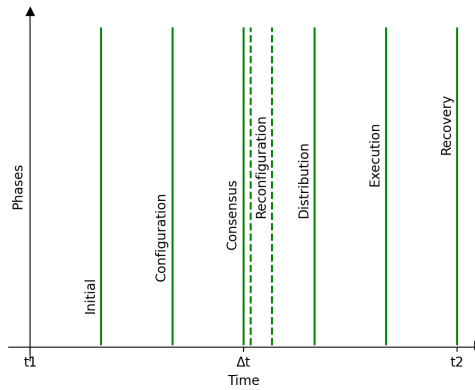


Figure 1: Phases of *dicomist* methodology

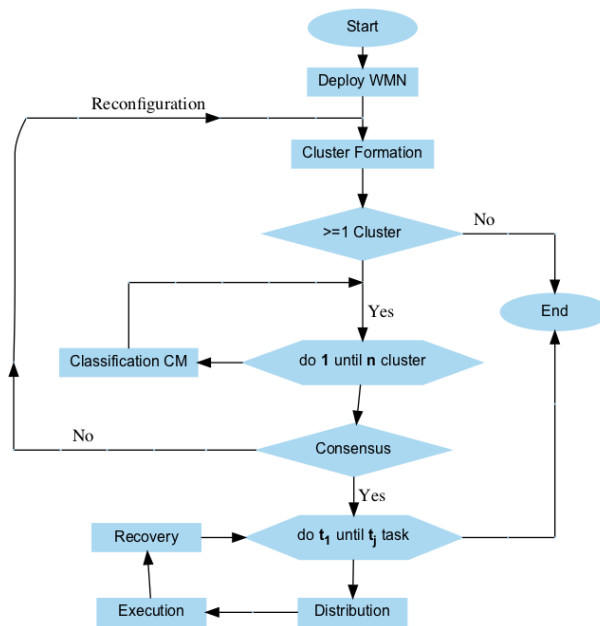


Figure 2: Flow diagram *dicomist* methodology

3.1 Phases of *dicomist* Methodology

3.1.1 *Initial* phase

The first phase, called **initial**, performs several steps to generate the two sets of nodes to be used later in the **configuration** phase. The initial phase is divided into two stages, deploying a wireless mesh network (WMN) and cluster formation. The stages are described below.

Stage A Deploy Wireless Mesh Network (WMN).

Set of heterogeneous devices without connection to one network, these devices have a wireless interface (Wi-Fi). Wireless networking technology is evolving as an inexpensive alternative for building federated and community networks. Besides its cost effectiveness, a wireless network brings operational efficiencies, namely mobility, and untethered convenience, to the end user. A wireless network can operate in the ad-hoc mode, where users are self-managed, and in the infrastructure mode. An *ad-hoc* network generally supports multi-hopping, where a data packet may travel over multiple hops to reach its destination [3].

The *ad-hoc* networks are unorganized and without infrastructure networks. The nodes in such types of networks perform many operations along with the routing functionalities. In Wireless Mesh Network (WMN), the nodes processing power and energy are always assumed to be very limited [29].

One of the main advantages of WMNs is that they do not need an infrastructure: the deployment phase, thus, is faster, less expensive, and less invasive for networks (either wired or wireless) that operate in a centralized way and need infrastructures [14].

One of the conditions described in the scenario is the lack of infrastructure for connecting the devices. For that reason, the WMN is ideal for these conditions. Let be a set of m devices to be connected with an algorithm mesh of type $\mathcal{A}(m)$, where \mathcal{A} is the algorithm's function, and m is the set of nodes to be interconnected. Therefore $N = \mathcal{A}(m)$. N is a vector containing the m nodes interconnected with the function \mathcal{A} WMN algorithm.

$$N = \begin{bmatrix} n_1 \\ n_2 \\ \dots \\ \dots \\ \dots \\ n_m \end{bmatrix} \tag{1}$$

Figure (3a) shows the set m of devices, and when applying function \mathcal{A} ; as a result, a vector N is generated, where the n interconnected devices are found, as shown in Figure (3b). For further details on the WMN deployment, see Appendix A.

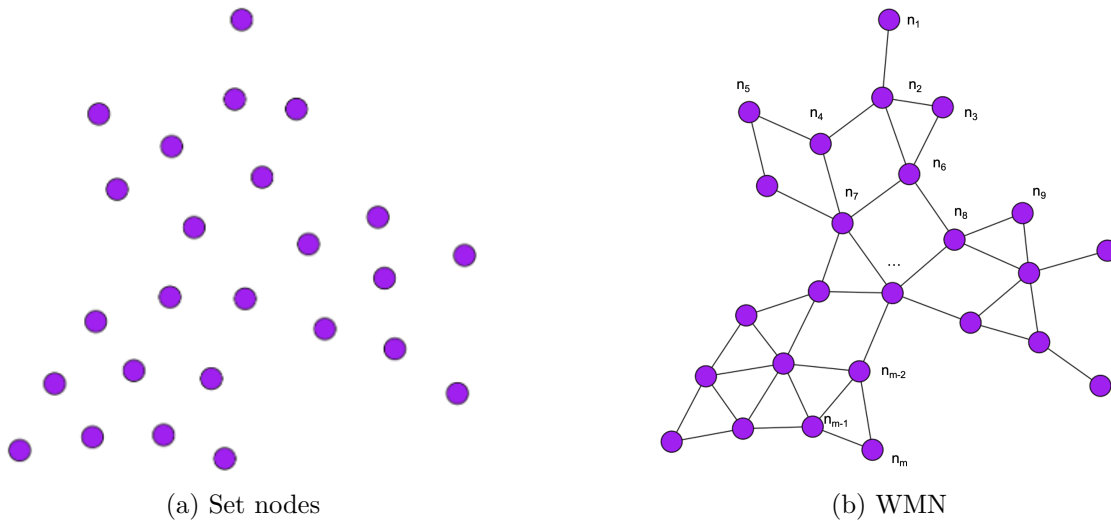


Figure 3: Deploy WMN

Stage B Cluster generation. Once the WMN is deployed, using a proactive protocol with multi-hop and decentralized reduces the overhead on the network and the amount of information stored [46]. The criteria that the available devices in the WMN must have are to know the availability and to generate a set of clusters where a cluster is a group of interconnected devices that work together and perform computationally intensive tasks. In order to implement clustering in WMN, it is necessary to form clusters using a clustering algorithm. Regarding sharing tasks between nodes, two categories of nodes have been defined: Cluster Head and Cluster Member. The Cluster Head (**CH**) is elected to be in charge of generating a transmission schedule, gathering data from all the **CM** in the cluster, and transmitting the assembled data back to the base station. The **CH** is a node with the best conditions for the variables. The Cluster Member (**CM**) is a node that belongs to a cluster but is not a **CH**, and its function is to iterate with the **CH**. A clustering algorithm has two main steps *CH Election* and *Cluster Formation* [87].

1. *CH Election*. The **CH** is the leader of a cluster and is in charge of gathering data and transmitting it to the BS. The **CHs** will consume more energy than typical **CMs**. Three **CHs election** schemes are adopted: Deterministic, Random, or Adaptive.

2. *Cluster formation.* After being elected, the **CHs** will advertise by broadcasting their information to other **CMs**. Several metrics can be used to determine the communication properties between a **CM** and a **CH**, such as communication cost, hop count, or physical distance. In some cases, the size of clusters is also considered when **CMs** join a cluster.

The *CH election*, in this case, will be *Adaptive* and is based on some particular parameters to design to adapt to the network and environment variations. The criteria to generate the cluster are defined:

- I. *Node processing capacity.* The metric used for processor performance is the concept of floating point operations per second (FLOPS). The theoretical calculation of the FLOPS is shown in the following equation [58]:

$$FLOPS = cores * \frac{cycles}{sec} * \frac{flops}{operations} * \frac{operations}{instruction} * \frac{instruction}{cycles} \quad (2)$$

- II. *Node storage.* The capacity of memory (RAM) to perform read and write operations by the processor. This metric is a node characteristic measured in Gigabytes (GB).
- III. *Distance between nodes.* Given a **CH** and a **CM**, you need to find a way to define the distance between them. The most common one is, of course, the Euclidean distance.
- IV. *Battery.* The alleged ability to deliver a variable voltage as its chemistry changes from charged to discharged sends the necessary power to the node—value range from 0 to 100% according to the use of the device.
- V. *Path loss.* The reduction in the power density (attenuation) of an electromagnetic wave as it propagates through space. To assign the received power, the following expression is used[4]:

$$P_{Rx} = P_{Tx} + K - 10\gamma * \log_{10}\left(\frac{d_i}{d_0}\right) + X_\sigma \quad (3)$$

Where:

P_{Tx} is the transmission power from source to destination. The signals of the WiFi is 20 dbm.

γ is the path loss exponent. This value is 2 or 2.4 depending on the obstruction in the medium:

d_i is the distance between a transmitter and a receiver.

d_0 is one constant value for outdoor scenarios. The values between are 1 to 10 meters.

X_σ is a zero-mean Gaussian random variable with σ representing the shadow fading factor [23].

K is a constant equal to:

$$20 \log_{10}\left(\frac{\lambda}{4 * \pi * d_0}\right) \quad (4)$$

Where:

λ . Speed of light/frequency. In WiFi signals is 2.4 Ghz.

d_0 is one constant value for outdoor scenarios. The values between are 1 to 10 meters.

The representation of the set nodes. From equation 1, an additional index applies to the vector N , which will be the dimensional variable of the node, renamed as node variables. These attributes are *node processing capacity*, *node storage*, *the distance between nodes*, *battery*, and *path loss*. The attribute declares with the index d . So the value n_m is now denoted $n_{m,d}$, and the vector N is transformed into a matrix. Let be a matrix N represented with two indices; the first represents the

population of nodes, and the range between them is from 1 to m . The second index has a range from 1 to d , where d are the dimensional variables of the node.

$$N = \begin{bmatrix} n_{1,1} & n_{1,2} & n_{1,3} & \dots & n_{1,d} \\ n_{2,1} & n_{2,2} & n_{2,3} & \dots & n_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ n_{m,1} & n_{m,2} & n_{m,3} & \dots & n_{m,d} \end{bmatrix} \quad (5)$$

A set of clusters generate from the set N (equation 5). Maybe one or more clusters configuration for the nodes used (better case) due to the heterogeneity of the nodes. When applying the function \mathcal{B} , to $\tilde{N}_{m,d}$ (N normalize matrix), the result is the set of **CH** (size w) that contains the nodes with the best characteristics, so the compliment is the **CM** set. $CH_{w,d} = \mathcal{B}(\tilde{N}_{m,d})$ and $\tilde{N}_{(m-w),d} = CM_{j,d} = \mathbf{CM}$ as shown in Figure 4. For further details on the Stage B, see Appendix B.

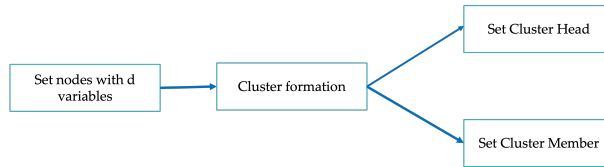


Figure 4: Step *Cluster Formation* of *dicomist* methodology

Where:

$$CH_{w,d} = \mathbf{CH}$$

$$CH = \begin{bmatrix} ch_{1,1} & ch_{1,2} & ch_{1,3} & \dots & ch_{1,d} \\ ch_{2,1} & ch_{2,2} & ch_{2,3} & \dots & ch_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ ch_{w,1} & ch_{w,2} & ch_{w,3} & \dots & ch_{w,d} \end{bmatrix}$$

$$\tilde{N}_{(m-w),d} = CM_{j,d} = \mathbf{CM}$$

$$CM = \begin{bmatrix} cm_{1,1} & cm_{1,2} & cm_{1,3} & \dots & cm_{1,d} \\ cm_{2,1} & cm_{2,2} & cm_{2,3} & \dots & cm_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ cm_{j,1} & cm_{j,2} & cm_{j,3} & \dots & cm_{j,d} \end{bmatrix}$$

3.1.2 Configuration phase

The configuration phase is necessary because it wishes to know which **CMs** can process the task τ . One solution is distributed classification if a centralized system is infeasible due to geographical, physical, and computational constraints. The objectives of the classification include robustness to changes in the network topology [92], efficient representation for high-dimensional, and good decision precision [44]-[20].

The classification problem is essential in data mining. The support vector machine (SVM) is one of the most popular classification algorithms in many fields [13], [18], [10], [36], [31]. For this reason, recent designs of distributed SVM classifiers rely on Support Vectors obtained from local training sets [20] and [91]. These Support Vectors obtained locally per node are incrementally passed on to neighboring nodes [21].

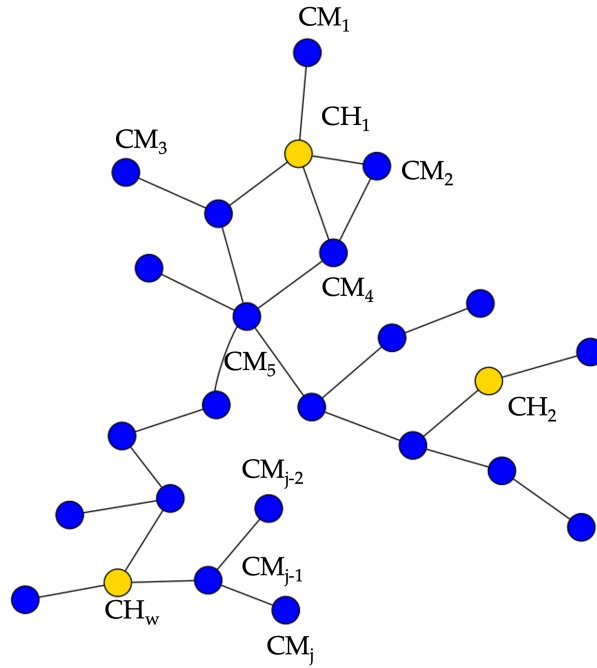


Figure 5: CH and CM sets

Once the **CH** and **CM** sets, the local conditions of **CM**'s are verified: processing, memory, battery, position, path loss, and link quality; the nodes will classify by Support Vector Machine (SVM).

Each **CH** requests the local conditions of the nodes for the devices to perform their classification using *SVM*. An example is calculating the *path Loss*, battery, and others. Three variables are used: S, D, and A.

- I. **Variable S.** The variable S value is binary and calculated (see appendix C); the classified nodes have the following deal: 1 or 2 and will have a binary value of 0. The nodes classified with zero values will have a binary value of 1.
- II. **Variable D.** The variable D value is binary, and the value is 1 (one) if the distance is more significant than x_{max} ; otherwise, the binary value is 0 (zero).
- III. **Variable A.** The variable A value is binary; if the value of *PL* is greater or equal to zero, the binary value of A is 00. Between $PL < 0$ and $PL \geq -44$, the value of A is 01. If $PL < -44$ and $PL > -90$, A is assigned the binary value of 10. If the $PL \leq -90$, then A is defined with the binary value of 11.

The relationship of $W_{v,j}$ will be between the CH_v and the CM_j . The binary value ($W_{v,j}$) is the weighted sum of the variables $W_{v,j} = A_{v,j} + D_{v,j} + S_{v,j}$. W can obtain values between 0 and 15, and $W_{v,j}$ obtain them with the following classification:

- a) 0-4 (0000-0100) is assigned the value two.
- b) 5-9 (0101-1001) is assigned the value one.
- c) 10-15 (1010-1111) is assigned the value of zero.

Given a set of nodes is represented by:

$$\mathbf{CH} = CH_{v,d} \text{ and } \mathbf{CM} = CM_{j,d}$$

$$CH_1 = ch_{1,1}, ch_{1,2}, \dots, ch_{1,d}$$

$$CH_2=ch_{2,1}, ch_{2,2}, \dots, ch_{2,d}$$

...

$$CH_v=ch_{v,1}, ch_{v,2}, \dots, ch_{v,d}$$

and

$$CM_1=cm_{1,1}, cm_{1,2}, \dots, cm_{1,d}$$

$$CM_2=cm_{2,1}, cm_{2,2}, \dots, cm_{2,d}$$

...

$$CM_j=cm_{j,1}, cm_{j,2}, \dots, cm_{j,d}$$

\mathcal{C} is the SVM function with the parameters (S, D, A) and the assigned kernel. Therefore $W = \mathcal{C}(ch, cm)$, where:

$$w_{1,1}=\mathcal{C}(ch_1,cm_1),$$

$$w_{1,2}=\mathcal{C}(ch_1,cm_2),$$

$$w_{1,3}=\mathcal{C}(ch_1,cm_3),$$

...

$$w_{1,j}=\mathcal{C}(ch_1,cm_j)$$

$$W_1 = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ w_{1,3} \\ \dots \\ \dots \\ w_{1,j} \end{bmatrix} \quad W_2 = \begin{bmatrix} w_{2,1} \\ w_{2,2} \\ w_{2,3} \\ \dots \\ \dots \\ w_{2,j} \end{bmatrix}, \dots, W_v = \begin{bmatrix} w_{v,1} \\ w_{v,2} \\ w_{v,3} \\ \dots \\ \dots \\ w_{v,j} \end{bmatrix}$$

Obtaining the vector W of all the nodes, SVM with kernel Radial Basis Function (RBF) is used for the classification, and the result is shown in the Figure 6:

$$K(cm, cm') = \exp(-\gamma \|cm - cm'\|^2) \tag{6}$$

Where: $S_1CM = \text{classified}(W_1, CM)$ and $S_1CM \subseteq CM$

$$S_1CM = \begin{bmatrix} s_1cm_{1,1} & s_1cm_{1,2} & s_1cm_{1,3} & \dots & s_1cm_{1,d} \\ s_1cm_{2,1} & s_1cm_{2,2} & s_1cm_{2,3} & \dots & s_1cm_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ s_1cm_{l,1} & s_1cm_{l,2} & s_1cm_{l,3} & \dots & s_1cm_{l,d} \end{bmatrix}$$

$1 \leq w$

$S_vCM = \text{classified}(W_v, CM)$ and $S_vCM \subseteq CM$

To finish the configuration phase, will to generate v subsets. $S_1CM \cup S_2CM \cup S_3CM \cup \dots \cup S_vCM = CM$

Rename subsets $S_1CM = S_1X, S_2CM = S_2X,$

$S_3CM = S_3X, \dots, S_vCM = S_vX$

For further details on the **Configuration phase**, see Appendix D.

3.1.3 Consensus phase

Consensus problems have a long history in computer science and form the foundation of distributed computing [7, 8, 15, 45, 78, 83]. A *consensus algorithm* (or protocol) is an interaction rule that specifies the information exchange between a node and all of its neighbors in the network [30]. In distributed systems and networks, it is often necessary for some or all of the nodes to calculate some function of

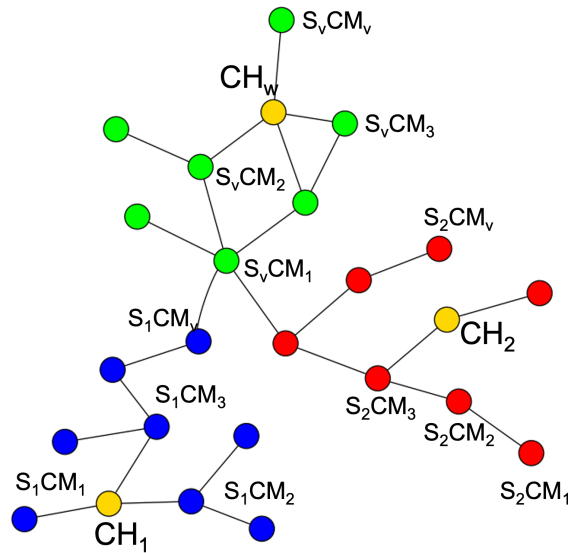


Figure 6: Classified set CM with SVM

specific parameters, for example, in the case of multi-agent systems, where all agents communicate with each other to coordinate their speed and direction. Another example is that *IoT* nodes in sensor networks may be tasked with calculating all the sensors average measurement values [30].

When considering a **CM** as an *IoT* device (due to its limited capabilities), the network generated by **CH** and **CM** (*Fog Computing* network) is similar to an *IoT*-type network. Consider a network with v nodes modeled by an undirected graph $G(V, E)$ with vertices $V := 1, \dots, v$ representing nodes, moreover, edges E describe links among communicating nodes. Potential application scenarios include but are not limited to the following ones: Wireless Sensor Networks, Distributed medical databases and Collaborative data mining [21], [2], [35] and [74].

Consensus phase is an agreement is achieved within the **CMs** minimum requirements. Let X one set of clusters to carry out a global consensus and information exchange, a *Fog Computing* type network must generate when necessary and local consensus must be carried out. A *Fog Computing* network can be modeled as a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where $V = V(G) = \{1, 2, \dots, v\}$ is the set of vertices and $E = E(G) = \{e_1, e_2, \dots, e_o\}$ is the set of edges of the graph. Change of variable $S_1CM_1 = X_1, S_1CM_2 = X_2, \dots, S_1CM_v = X_v$. Where $X_v = [x_1, x_2, \dots, x_v]$ (Figure 7a) is the node dimension. A function considers $x_v(t)$ where: v is the node in time t where the calculation for the consensus estimate performed at each node v , where ($v \in V$), given $x_v(0)$ denotes the initial measured value in v , which can update through iterative exchanges between neighboring nodes, and consensus or average can reach across all nodes with the following expression and Figure 7b:

$$x_k(t + 1) = x_k(t) - \mu \sum_{(j \in N_k)} (x_j(t) - x_k(t)) \tag{7}$$

Where:

$x_k(t)$ represents the state value (used to evaluate the matching value for new incoming services) at node k at time t , which can be intuitively understood as the estimate of the consensus value of k .

μ denotes the step size in each iteration.

N_k denotes the neighbor list of $k \in E$.

k where $k = 1, 2, \dots, v$

$x_k(0) \in R$ and $x(0) = [x_1(0), x_2(0), \dots, x_d(0)]$

The cluster in Figure 7b is used as an example to represent the adjacency matrix (AM) with rows and columns labeled by graph vertices, with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are adjacent or not.

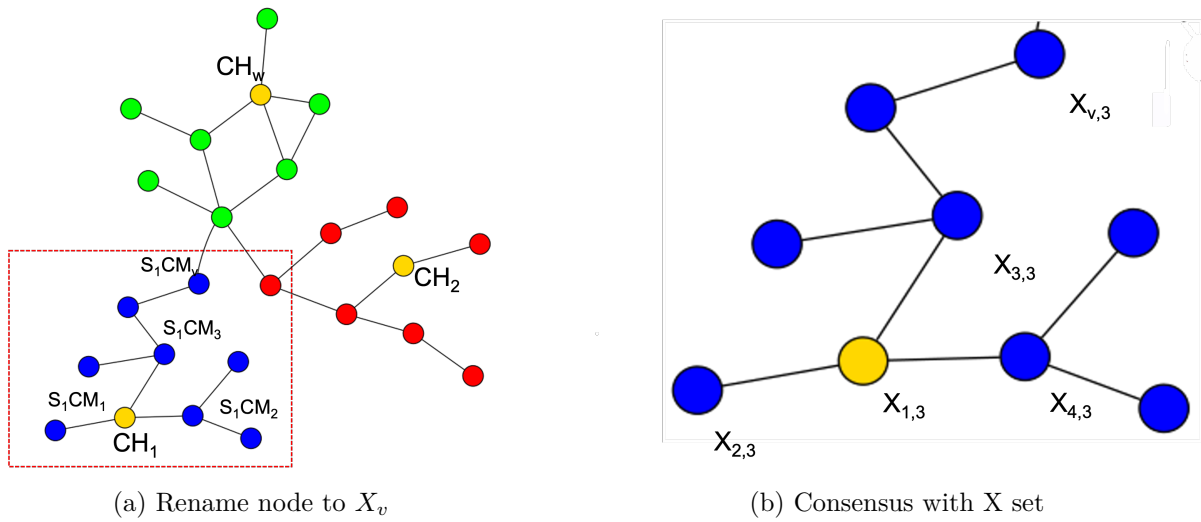


Figure 7: Consensus WMN

$$AM = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 1 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

The convergence properties of 7 are largely determined by the Laplacian matrix [16] L , thus

$$L(t + 1) = f(x) = \begin{cases} d_k, & k = j \\ -1, & (k, j) \in E \\ 0, & else \end{cases} \tag{8}$$

Where:

d_k is the degree at node k .

$$LP = \begin{bmatrix} d_k & -1 & 0 & \dots & 0 \\ -1 & d_k & -1 & \dots & 0 \\ -1 & 0 & d_k & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & d_k \end{bmatrix}$$

Let an $n \times 1$ vector $x(t)$ denote the states vector of all nodes at time t , which can be written as:

$$x(t + 1) = x(t) - \mu Lx(t) = (I - \mu L)x(t) \tag{9}$$

It can be simplified as:

$$x(t + 1) = Wx(t) \tag{10}$$

Where $W = I - \mu L$. For a graph, its Laplacian matrix L with eigenvalues $\lambda_0 \lambda_1 \dots \lambda_{(n-1)}$ and λ_0 is always 1 because every Laplacian matrix has an eigenvector. $\epsilon_0[1, 1, \dots, 1]$. The convergence time is [85]:

$$\tau = \frac{1}{\log(1/\rho)} \tag{11}$$

3.1.4 Reconfiguration phase

A change in the nodes triggers a reconfiguration, but it does not activate immediately. Nodes that join are held in a member state but are not considered in commit-level decisions or leader selection until a quorum of them is reached. This means they have seen all previous transactions up to joining the network. Similarly, nodes about to retire acknowledge when their state changes.

3.1.5 Distribution phase

CH sends tasks to CM.

3.1.6 Execution phase

CM executes tasks, and this tasks depends to the capability of the node.

3.1.7 Recovery phase

The results found from the CM sent to the CH and after to the leader.

4 Experiments and Results

In this section, we implement the proposed protocol in real hardware to demonstrate the effectiveness of the design methodology for one group of heterogeneous devices (m). The initial phase environment consists of a set of 13 heterogeneous devices. The hardware used is shown in Table 1

The Figure 8 is obtained by applying $N=\mathcal{A}(m)$ to the sets of nodes described in Table 1, resulting in the deployment of the WMN.

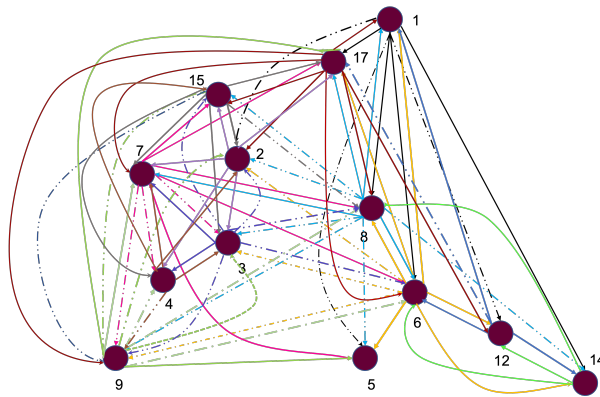


Figure 8: WMN Topology

The WMN deployment in the building is in Appendix A. The next stage in the **initial** phase is cluster generation. Delete the links with a weak connection; the topology of the mesh network is shown in Figure. 9a. From the previous Figure, the stage of cluster generation is applied with $CH_{w,d}=\mathcal{B}(\tilde{N}_{m,d})$, the dimension is $CH_{1,5}=\mathcal{B}(\tilde{N}_{13,5})$, and the result is one **CH** is shown Figure 9b and 10

The resulting vector in the **configuration** phase $W = \mathcal{C}(ch, cm)$ function, the vector is graphed to visualize it, resulting in the Figure 11.

In Figure 12, the representation of nodes is as follows. The nodes in purple are the **CMs** classified with the value 0 (zero), which cannot perform the processing. The nodes in blue are **CMs** that can perform the processing, and the yellow node is the **CH**.

The consensus phase has one set of nodes in Figure 12, the purple nodes are removed, and the two links become the bidirectional link, giving the result in a new set called X, which is shown in Figure 13. When executing the consensus algorithm, the topology generated with the nearest neighbor and the communication as shown in the Figures 14 and 15.





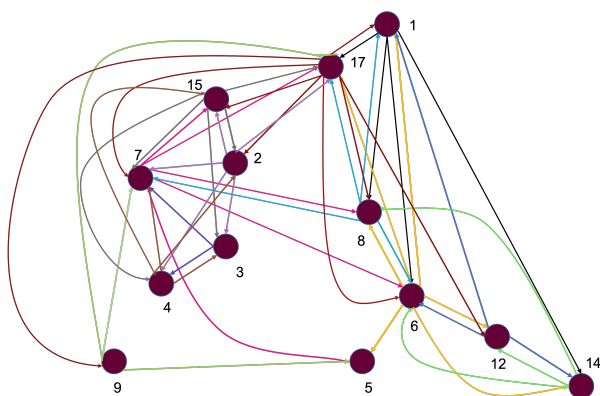
Device	Description
1.Seven Raspberries Pi 3B+ 	<ul style="list-style-type: none"> i) CPU Cortex-A53 (ARMv8) 64-bit SoC@1.4GHz. ii) 1GB LPDDR2 SDRAM/32GB micro-SD card iii) Fedora 33(3 devices), Ubuntu 20.04(3 devices) and Raspbian 10(1 device) iv) batctl 2021.4
2.Two Raspberries Pi 4 	<ul style="list-style-type: none"> i) Quad Core Cortex-A72(ARM)64-bit @1.5GHz ii) 4GB LPDDR4 RAM/32GB micro-SD card iii) Ubuntu 20.04 and Fedora 35 iv) batctl 2021.4
3.Two Arduino's Yun 	<ul style="list-style-type: none"> i) MCU Atmel AVR 8 bits ATmega32U4 and CPU MIPS Qualcomm ii) Atheros AR9331 @ 400 MHz iii) 16MB Flash Memory iv) OpenWRT (LEDE Yun 17.11)
4.Two TP-Link TL-WDR3600 	<ul style="list-style-type: none"> 1. CPU MIPS 74Kc i) 8 MB Flash Memory ii) OpenWRT 20.21.3 iii) batctl 2021.4

Table 1: Heterogeneous devices specification



(a) WMN without weak link

```

codigoAlgoritmo_v3 -- mena@estigja:~/minar/monero/mac -- vi salida18Fe...
Clasificando FogNode[11]=RAM:1.0 y FLOPS:44.79999923706055
clasR:0
clasF:0
Clasificador en 0, RAM:1.0 y FLOPS:44.79999923706055
S:1
D:0
PL:10
W:1001
Clasificando FogNode[12]=RAM:1.0 y FLOPS:46.33443107785982
clasR:0
clasF:1
Clasificador en 0, RAM:1.0 y FLOPS:46.33443107785982
S:1
D:0
PL:10
W:1001
Recibo los valores del CHX:40.0, y CHY:8.0
La posicion del CH esta en:11
Datos del CH normalizados son:
X:0.7838261963262275,y:0.8334689659466117,RAM:-0.43954766145495383,GFLOPS:-1.264
6021080547745,PL:-0.10100145146272392,Etiqueta:1
Datos del CH son:
X:40.0,Y:8.0,RAM:1.0,GFLOPS:44.79999923706055,PL:-30.4459970202808,Etiqueta:1
    
```

(b) Result of Cluster Formation

Figure 9: Cluster Formation

When carrying out another scenario with less complex tasks, more nodes that comply with the

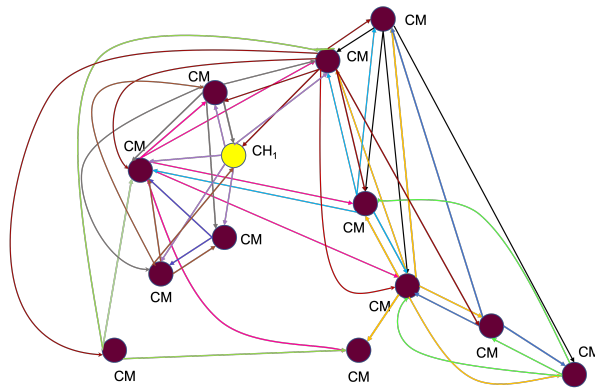


Figure 10: Result of CF with one CM

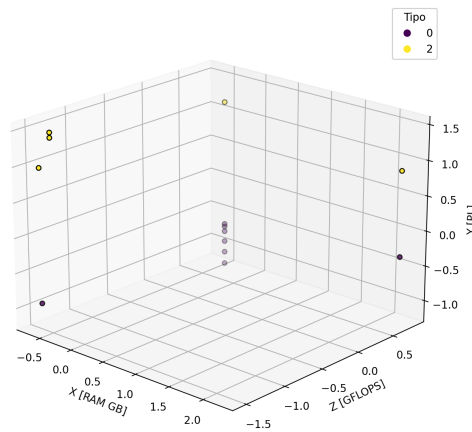


Figure 11: Result of SVM with CH_1

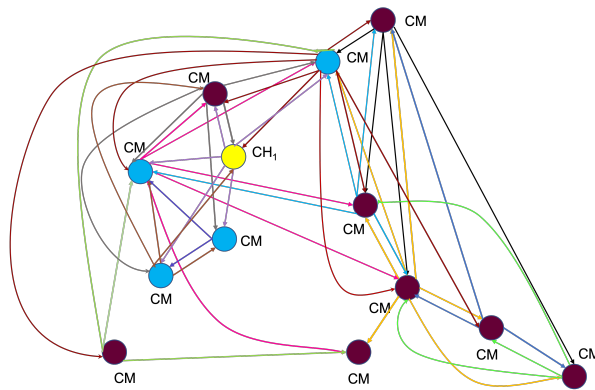


Figure 12: Topology with SVM

capacities to perform the task processing participate; the clusters can be seen, and how they agree to carry out the tasks later, as shown in Figures 16a, 16b and 17.

Figure 16b shows different topologies based on a ring structure. The idea behind the chosen topologies is to test the protocol’s performance for the minimum quantity of links, i.e., the ring structure, and then evaluate its behavior. The idea behind these topologies is to emulate networks with mobility where clusters have environments with multiple interfaces. An agent bridges two subnetworks with different technologies at the physical layer of the protocol stack.

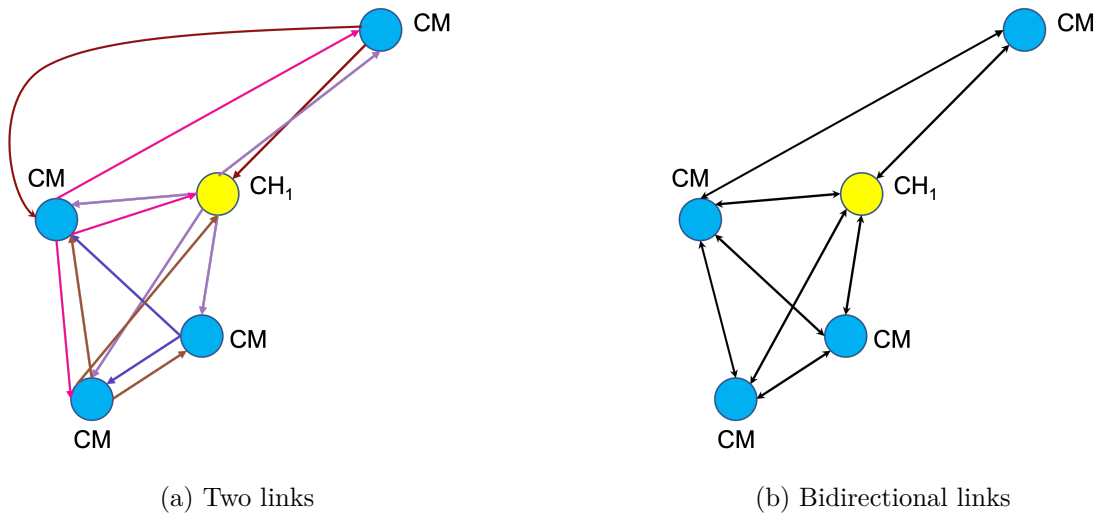


Figure 13: Set X

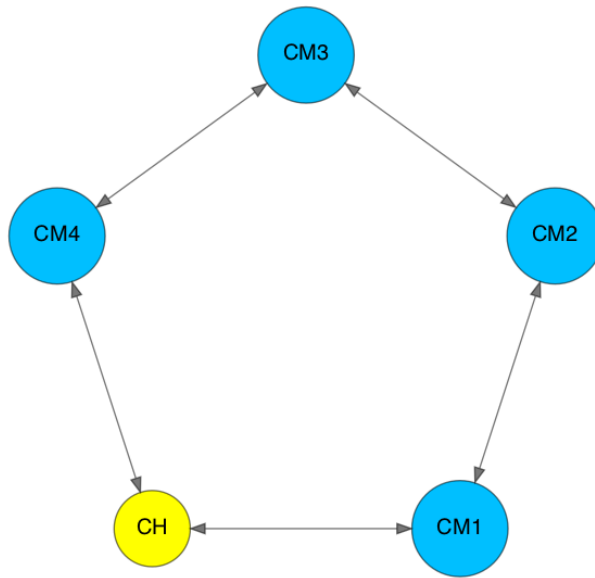


Figure 14: Pure ring topology

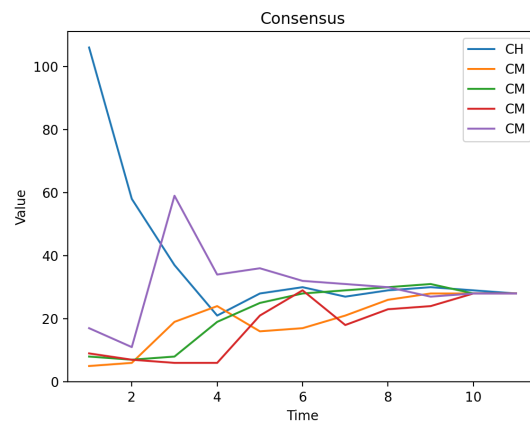


Figure 15: Results of the communication topology shown in Figure 14

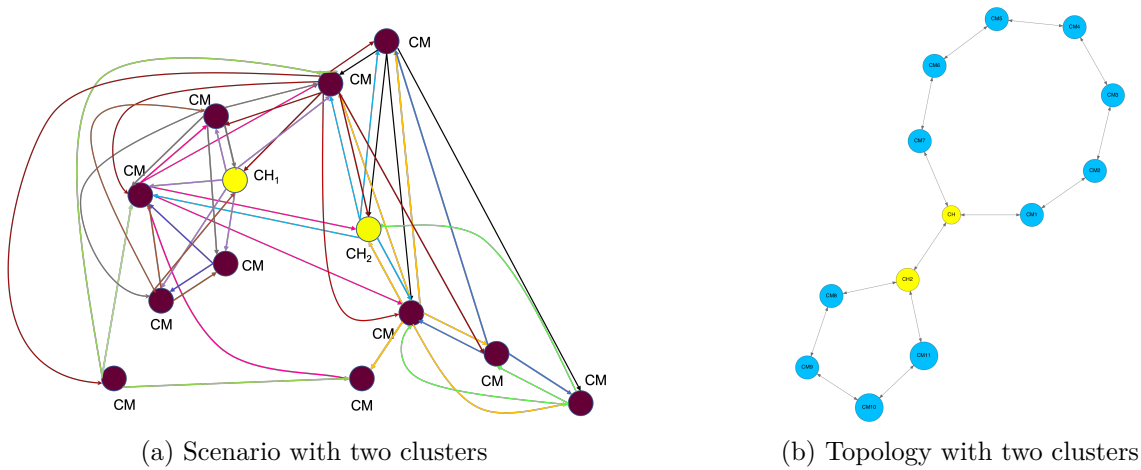


Figure 16: Results with two clusters

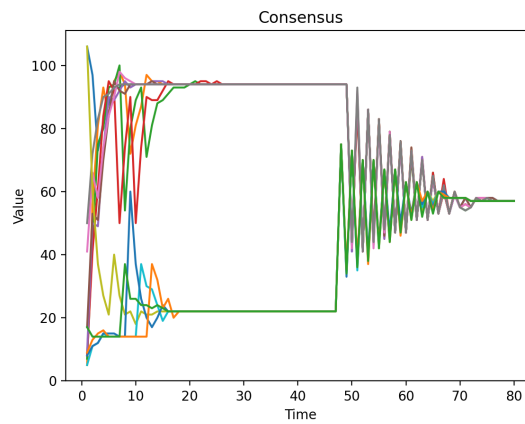


Figure 17: Consensus with two clusters

5 Future work

Future research will focus on expanding the scope of real testbed scenarios and increasing the number of wireless nodes to enhance the robustness and applicability of the findings. By incorporating more real-world environments and scenarios, the research aims to provide a more comprehensive understanding of the practical implications and effectiveness of *dicomist*. Additionally, increasing the number of wireless nodes will enable a more thorough exploration of scalability and performance under varied network conditions, further refining the efficacy of the proposed solutions in *Fog Computing* environments.

6 Conclusion

The manuscript outlines a strategy for *Fog Computing*, delineating five main stages crucial for creating an effective distributed cluster computing system. These stages encompass *Federated Computing*, configuration consensus, reconfiguration distribution, and processing, with a focus on utilizing low-cost processors. Establishing a stable computing system relies heavily on accurately configuring the cluster and selecting a *Cluster Head* within a precise time interval. Validation of the consensus algorithm ensures feasible agreement among nodes, facilitating timely processing of information. While the manuscript primarily explores processing tasks already installed at nodes, it acknowledges the potential for task migration, albeit not explored due to its time-consuming nature. The study showcases promising results for a heterogeneous platform but underscores the need for further investigation into additional stages for *Fog Computing*. This framework enhances computing capabilities among nodes

with varying processing abilities, yet addressing configuration, reconfiguration, and migrations remains a challenge in an open network environment. Performance was evaluated on a device benchmark with a low scale in a realistic environment. Experimentally, the results were both for static communication and variable topology in time.

A Deploy WMN

Create the virtual interface, and then the mesh network will use the B.A.T.M.A.N. protocol [56], [55], [54], [1], [32], [42], [12] and [79]. Each B.A.T.M.A.N. packet is encapsulated in a single UDP data packet. A B.A.T.M.A.N. packet consists of one OriGinator Message (OGM) and zero or more attached HNA extension messages, since its function is to find other B.A.T.M.A.N. nodes and define the best neighbor to reach others nodes. It also keeps track of new neighboring nodes and informs neighbors of their existence, OGM packets are used for this purpose, headers of this packet are shown in Table 2.

Concept	Description
Originator Address	Identify the node that created the B.A.T.M.A.N. package
Sequence Number	Used to measure link quality and detect duplicates.
Transmit Quality	Describe the link quality of the total route to the originator.
Address of the previous sender	Used to discard the OGMs already transmitted by the receiving node.
Time to Live (TTL)	Limits the number of nodes an OGM can traverse.
Host Network Announcement (HNA)	Describe the number of non-batman nodes capable that can be accessed through the creator. Information about each DNA is attached to the GMO message.
Gateway Flags	Used if the node offers a connection to another network

Table 2: Headers of B.A.T.M.A.N. protocol

To estimate the link quality to an originator, the transmission quality (TQ) in an OGM is changed during its passage through the network. The TQ describes the probability that one packet will reach to destination and is calculated. The value is stored in an eight bits value between 0 and 255 (decimal), which provides greater granularity without increasing the packet size. The quality of the link is measured with the help of three packages Figure 18.

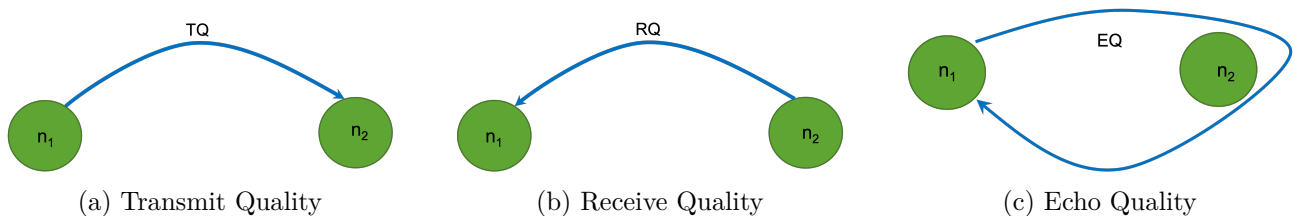


Figure 18: The three link qualities used in B.A.T.M.A.N

The transmission quality on a route can be computed using Equation 12, as follows:

$$EQ = TQ * RQ : TQ = \frac{EQ}{RQ} \quad (12)$$

Where:

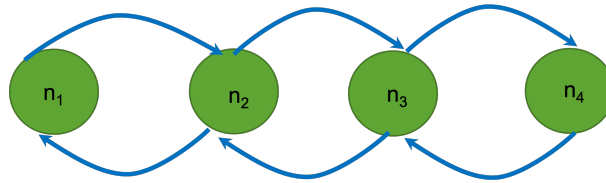


Figure 19: Path to calculate to transmission quality

$$TQ(n_1, n_2) = P_T(n_1, n_2) * P_R(n_1, n_2)$$

P_R is represented as:

$$TQ(n_1, n_2) = P_T(n_1, n_2) (1 - (1 - P_R(n_1, n_2))^3)$$

Now the quality of the transmission is expressed in a general way:

$$TQ(\mathcal{P}) = (1 - \alpha)^{k-1} \times \prod_{i=1}^{k-1} TQ(n_i, n_{i+1})$$

Where: \mathcal{P} is path.

$$\text{HOP PENALTY} = (1 - \alpha) \text{ where } 0 < \alpha < 1$$

At the end of the generation of the mesh network, the number of nodes involved in the mesh network is of size m , called the set N . Applying $N = \mathcal{A}(m)$.

The logical distribution of the WMN topology of Figure 8 is shown in the following Figure 20:

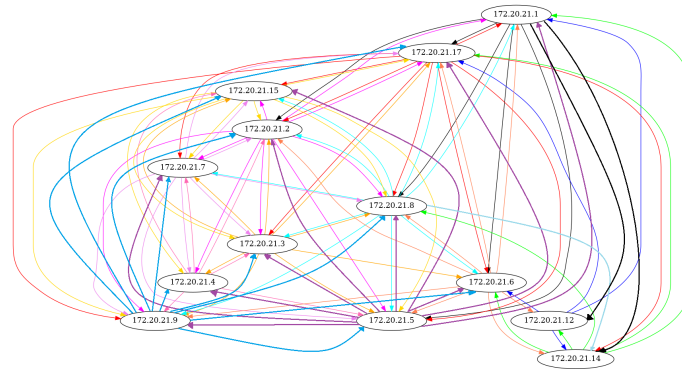


Figure 20: Deploy WMN

B Cluster Formation with crow search algorithm

The ultimate goal for clustering is to divide devices in a WMN into different clusters. After joining a cluster, a device typically only needs to communicate with **CH**. The routing scheme is either a single hop or multi-hop because it has one WMN included. They selected the **CHs** with one of the following three schemes in this part [87].

Deterministic: In this scheme, the **CHs** are preset and placed at fixed locations in a network. This case is different when super nodes (with powerful processing ability and high energy storage) exist since the devices in fields are usually homogeneous, and the super nodes can die for unexpected reasons.

Random: **CHs** can be selected among devices based on randomly generated values. A random **CH** election scheme is a simple and beneficial strategy if a network is homogeneous.

Adaptive: The adaptive **CH** election is based on parameters such as remaining residual energy or distance. With different system objectives, a clustering algorithm can use a specific combination of these parameters. An adaptive **CH** election approach is designed to adapt to the network and environment variations.

The *CH election*, in this case, will be Adaptive and is based on some particular parameters. To generate the clusters, we must know how many **CHs** are going to be needed; for this, the bio-inspired algorithm called Crow Search Algorithm (CSA) is used; below are the steps to execute the algorithm [5]:

- Step 1. Initialize the problem and adjust the parameters.
- Step 2. Initialize positions and memories.
- Step 3. Evaluate the fitness position.
- Step 4. Generate new positions.
- Step 5. Verify the feasibility of the new positions.
- Step 6. Evaluate fitness function for new positions.
- Step 7. Update memory.
- Step 8. Verify the termination criteria.

Each node variable has your measurement units, normalization to the set N is performed in order to eliminate sensitivity and the influences effect. Used to normalize the standard score, which is reflected in the following expression:

$$N_{norm} = \bar{N} = \frac{X - \mu}{\sigma} \tag{13}$$

Apply to the N matrix

$$x_{m,d} = \bar{n}_{m,d} = \frac{n_{m,d} - \mu_{m,d}}{\sigma_{m,d}} \tag{14}$$

$$\bar{N} = \begin{bmatrix} \bar{n}_{1,1} & \bar{n}_{1,2} & \bar{n}_{1,3} & \dots & \bar{n}_{1,d} \\ \bar{n}_{2,1} & \bar{n}_{2,2} & \bar{n}_{2,3} & \dots & \bar{n}_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \bar{n}_{m,1} & \bar{n}_{m,2} & \bar{n}_{m,3} & \dots & \bar{n}_{m,d} \end{bmatrix}$$

Rename $\bar{N}=X$:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ x_{m,1} & x_{m,2} & x_{m,3} & \dots & x_{m,d} \end{bmatrix} \tag{15}$$

$X_{m,d}$ where m is the population and d is the dimension node. X_m is the vector described as $X_m = [x_{m,1}, x_{m,2}, \dots, x_{m,d}]$

When executing the CSA algorithm for each iteration, the following expression is evaluated to find the nodes with the best conditions:

$$X_m^{iter+1} = \begin{cases} X_m^{iter} + r_i * f_m^{iter} * (M_j^{iter} - X_m^{iter}) & r_j \leq AP_j^{iter} \\ random\ value & else \end{cases} \tag{16}$$

Where:

r_i is a random number with uniform distribution between 0 and 1.

AP diversification parameter in optimization algorithms. Value is 0.1.

fl is the intensification parameter for diversification algorithms. Value is 2.0.

M is the memory matrix that is updated with the fitness function.

$iter$ is the number of iterations that the execute algorithm.

$$M_m^{iter+1} = \begin{cases} X_m^{iter} & \text{if } f(X_m^{iter+1}) \text{ is better } f(M_m^{iter}) \\ M_m^{iter} & \text{else} \end{cases} \quad (17)$$

Where: $f(m)$ is fitness function.

The fitness function. This function will optimize the variables of the node m and the best characteristics will be obtained, these nodes will be selected how **CH**. The *fitness* function is shown below:

$$f(m) = \frac{RAM_m + FLOPS_m + (X_m^{-1} + Y_m^{-1})}{PL_m + B_m^{-1}} \quad (18)$$

Where:

RAM_m . The *RAM* metric is the amount of memory the m node.

$FLOPS_m$. Defines the floating point operations performed by the m node.

X_m^{-1} . The random position on the X axis of the node m in a scenario, where X_m^{-1} is equal to $\frac{1}{X_m}$

Y_m^{-1} . The random position on the Y axis of the node m in a scenario, where Y_m^{-1} is equal to $\frac{1}{Y_m}$

PL_m . The *PL* metric is the reduction in power density from the m node towards the destination to be transmitted.

B_m^{-1} . The percentage of the battery charged to send power to the node. Where B_m^{-1} is equal to $\frac{1}{B_m}$

The *fitness* function generates the following behavior in the variables, by having a quotient, where the metrics that need to be increased are placed in the numerator and the desired metrics in the denominator to reduce.

The maximum number of iterations defined by $iter_{max}$ is determined. The iterations are the number of times the algorithm performs in the following steps (4 to 7). In this case, the maximum iteration will have a direct relationship with the number of **CHs** that are going to be used to generate a set of clusters, generally ranging between 5-15% of the population[77]; this percentage is the quantity CH to find, the value w is $w = \frac{m}{10}$. A continuation is shown as the result of the execution of cluster formation with the crow search algorithm; we change the value w and obtain two **CM**.

C Appendix Classification of the two variables nodes

Classification of the two variables nodes: RAM and FLOPS.

- I. It is labeled with 0 (zero) if the device has characteristics less than 2 Gb RAM. In this case, the characteristics of the FLOPS do not matter since they do not exceed 50 GFLOPS.
- II. It is labeled with 1 (one) if the device has the characteristics: Memory is between 2 to 4 Gb in RAM. FLOPS ranges from 44.8 to 100 GFLOPS.
- III. The label with value 2 (two) is assigned if the device has the following characteristics: Memory greater than 3GB in RAM. FLOPS greater than 80 GFLOPS, the average of these devices is 104.13.

```
MacMena:codigoAlgoritmo_v3 mena@estigia:~/minar/monero/mac -- bash -- 80x24
MacMena:codigoAlgoritmo_v3 mena$ java mx/unam/pcio/fog/EscenarioTest mobileData.
csv mobileData13jo.csv
```

(a) Stage 1

```
MacMena:codigoAlgoritmo_v3 mena@estigia:~/minar/monero/mac -- vi salida18Fe...
Genero el escenario
El numero total de lineas es:303
El tamaño del arreglo es:302
ff:302
Archivo de salida original:
Archivo de salida normalizado:
El numero total de lineas es:14
Ingreso en la configuración limas
El tamaño del arreglo limas es:13
-----mobileData13jo.csv
ff limas:13
Archivo para la predicción LIMAS:
Lower[0]:-1.4431982911069143
Upper[0]:1.768702253503311
Lower[1]:-1.0188290195545573
Upper[1]:2.46714308090673
Lower[2]:-1.506824553868106
Upper[2]:0.7839808281551534
Lower[3]:-0.4637465183020646
Upper[3]:2.212488394343549
Lower[4]:-1.1228698692114372
Upper[4]:1.52171971144982068
"salida18Feb023.txt" 289w, 8001B
```

(b) Stage 2

```
MacMena:codigoAlgoritmo_v3 mena@estigia:~/minar/monero/mac -- vi salida18Fe...
Lim inf X:-1.4431982911069143 y sup X:1.768702253503311
Maxima iteración y la Población:13
La dimension es de:5
El vector MEM es 15
La matriz X es de longitud X[13][5]
La matriz MEM es de longitud MEM[13][5]
La matriz XNDW es de longitud XNDW[13][5]
El vector fitness[13]
El vector fitness[13]
El vector mm[13]
El vector VALS[13]
El valor AP: 0.1 y el de fl:2.0
Se acaba el constructor
Ejecuto el metodo init(FogNode[])
La matriz de los nodos:
X, Y, RAMGB, FLOPS, PL
X[0]:-1.4432 2.4871 -0.6637 -1.3666 1.2337
X[1]:-1.1883 0.9289 -0.6637 -1.3666 1.4271
X[2]:-0.9844 -0.6293 -0.2529 0.5970 1.2683
X[3]:-0.8314 0.5394 -0.6637 -1.5068 0.9885
X[4]:-0.6785 -0.6293 2.2125 0.7840 0.8608
X[5]:-0.2196 0.5394 2.2125 0.7840 -0.3625
```

(c) Stage 3

```
MacMena:codigoAlgoritmo_v3 mena@estigia:~/minar/monero/mac -- vi salida18Fe...
Ejecuto la funcion fitness con el valor X
fitness[0]:-0.0223713909843714
fitness[1]:-0.6721624580387349
fitness[2]:0.575947902644045
fitness[3]:-1.593919850296434
fitness[4]:4.285076954015983
fitness[5]:-0.873030134989946
fitness[6]:-1.618776187274073
fitness[7]:-2.463542537065888
fitness[8]:-2.59243804966517
fitness[9]:-1.2879594591804937
fitness[10]:0.858393263091739
fitness[11]:-1.520486214789569
fitness[12]:-1.5271971144982068
Asigno los valores de X a MEM
-----1-----
Valor dep[0]:9.873030134989946 me da la funcion fitness
Valor dep[1]:15.0 da la posicion del nodo
-----2-----
Valor dep[0]:9.873030134989946 me da la funcion fitness
Valor dep[1]:15.0 da la posicion del nodo
-----3-----
```

(d) Stage 4

Figure 21: Execution CSA

```
MacMena:codigoAlgoritmo_v3 mena@estigia:~/minar/monero/mac -- vi salida18Fe...
Clasificando FogNode[11]=RAM:1.0 y FLOPS:44.79999923706055
clasR:0
clasF:0
Clasificador en 0, RAM:1.0 y FLOPS:44.79999923706055
S:1
D:0
PL:10
W:1001
Clasificando FogNode[12]=RAM:1.0 y FLOPS:46.33443107785982
clasR:0
clasF:1
Clasificador en 0, RAM:1.0 y FLOPS:46.33443107785982
S:1
D:0
PL:10
W:1001
Recibo los valores del CHX:40.0, y CHY:8.0
La posicion del CH esta en:11
Datos del CH normalizados son:
X:0.7838261963262275, Y:0.8334689659466117, RAM:-0.43954766145495383, GFLOPS:-1.264
6021080547745, PL:-0.10100145146272392, Etiqueta:1
Datos del CH son:
X:40.0, Y:8.0, RAM:1.0, GFLOPS:44.79999923706055, PL:-30.4459970202808, Etiqueta:1
```

Figure 22: Result of Cluster Formation

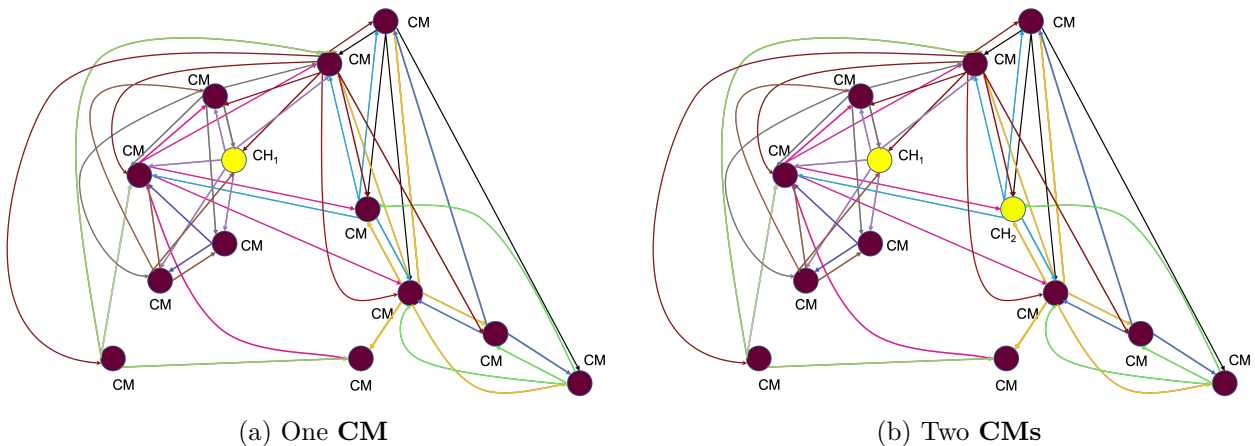


Figure 23: Results of the Cluster Formation

D Appendix Classified set CM with SVM

Support Vector Machine (SVM) is one of the most effective supervised machine learning algorithms currently available. It is possible to get good results without much adjustment to the problems it wishes to solve. Figure 24 shows the ACM's computer classification system (CSS) [57] for the SVM level.

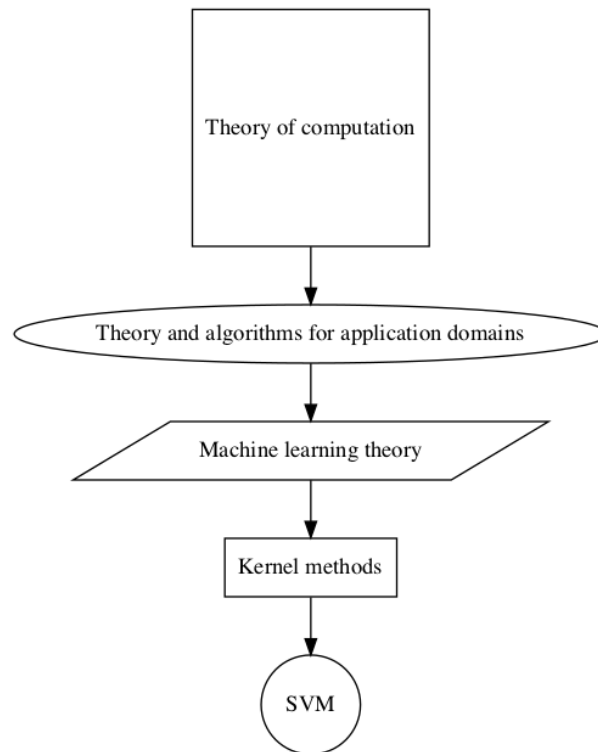


Figure 24: CSS for the SVM level

Once the **CH** and **CM** sets, the local conditions of CM's are verified: processing, memory, battery, position, path loss, and link quality; the nodes will classify by Support Vector Machine (SVM).

Sometimes polynomial or linear kernels need to be more sophisticated to work with a complex data-set. This case uses another, more complicated, kernel: the Gaussian kernel. It is also named RBF kernel, where RBF stands for Radial Basis Function. A radial basis function is a function whose value depends only on the distance from the origin or some point. The RBF kernel function is:

$$K(cm, cm') = \exp(-\gamma \|cm - cm'\|^2) \quad (19)$$

Each **CH** requests the local conditions of the nodes for the devices to perform their classification using *SVM*. An example is calculating the *path Loss*, battery, and others. Three variables are used: S, D, and A.

- I. **Variable S.** The variable S value is binary and calculated (see appendix C); the classified nodes have the following deal: 1 or 2 and will have a binary value of 0. The nodes classified with zero values will have a binary value of 1.
- II. **Variable D.** The variable D value is binary, and the value is 1 (one) if the distance is more significant than x_{max} ; otherwise, the binary value is 0 (zero).
- III. **Variable A.** The variable A value is binary; if the value of *PL* is greater or equal to zero, the binary value of A is 00. Between $PL < 0$ and $PL \geq -44$, the value of A is 01. If $PL < -44$ and $PL > -90$, A is assigned the binary value of 10. If the $PL \leq -90$, then A is defined with the binary value of 11.

The relationship of $W_{v,j}$ will be between the CH_v and the CM_j . The binary value ($W_{v,j}$) is the weighted sum of the variables $W_{v,j} = A_{v,j} + D_{v,j} + S_{v,j}$. W can obtain values between 0 and 15, and $W_{v,j}$ obtain them with the following classification:

- a) 0-4 (0000-0100) is assigned the value two.
- b) 5-9 (0101-1001) is assigned the value one.
- c) 10-15 (1010-1111) is assigned the value of zero.

Generate the SVM graphs, the parameters C and $gamma$ of the RBF kernel must be adjusted. Therefore, a heat map is generated to obtain the parameters as shown in Figure 25a. The Figure 25b is the set nodes training about SVM and yours classification.

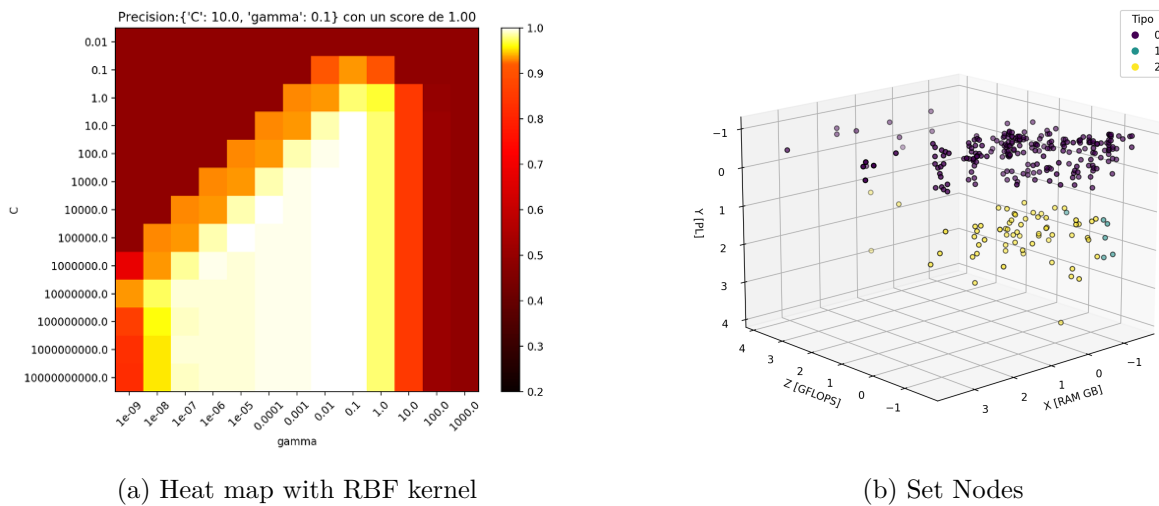


Figure 25: SVM Results

The prediction was made with data in original and normalized scales, the comparison with different performance indices and the results are shown in Table 3.

Performance indexes	Originals values	Normalized values
Jaccard index	0.7930	1.0
Confusion matrix	$\begin{bmatrix} 29 & 0 & 6 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix}$	$\begin{bmatrix} 35 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 24 \end{bmatrix}$
F1 Score	0.8777	1.0
Log Loss	0.2750	0.0321

Table 3: Performance indexes

Others concepts to measure are the sensitivity indexes with original and normalized values, where the results are compared in Table 4.

E Appendix Consensus

Consensus (also called common consensus) is specified in terms of 2 primitive functions: *propose* and *decide*. Each process (or node) has an initial value for the agreement through the primitive *propose*.

Concept	Originals values	Normalized values
Precision	0.8833	1.0
Sensitivity value 0	0.82857	1.0
Sensitivity value 1	0.0	1.0
Sensitivity value 2	1.0	1.0

Table 4: Sensitivity indexes

Proposed values are private of the process, and the action of proposing is local. This action launch broadcast events on which the processes exchange the proposed values to reach the deal eventually. All correct processes must decide on a value via the primitive *decide*. This value would be one of the proposed values [24], [33], [34], [45] and [61]. Consensus must satisfy the following properties:

Termination. Every successful process eventually decides on some value.

Validation. If a process decides v , then v was proposed by some process.

Integrity. No process decides two times in the same round.

Agreement. Two correct processes do not decide a distinct value.

With the previous concepts, the consensus algorithm is programmed with the following scope. A *Fog Com-puting* network can be modeled as a graph $\mathbf{G}=(\mathbf{V}, \mathbf{E})$ where $V=V(G)=\{1,2,\dots,v\}$ is the set of vertices and $E=E(G)=\{e_1,e_2,\dots,e_o\}$ is the set of edges of the graph. Change of variable $S_1CM_1 = X_1, S_1CM_2 = X_2, \dots, S_1CM_v = X_v$. Where $X_v = [x_1,x_2,\dots,x_v]$ is the node dimension. Various executions of the algorithm depend on the task's complexity. The results have been similar and are shown below in the following Figure 26.

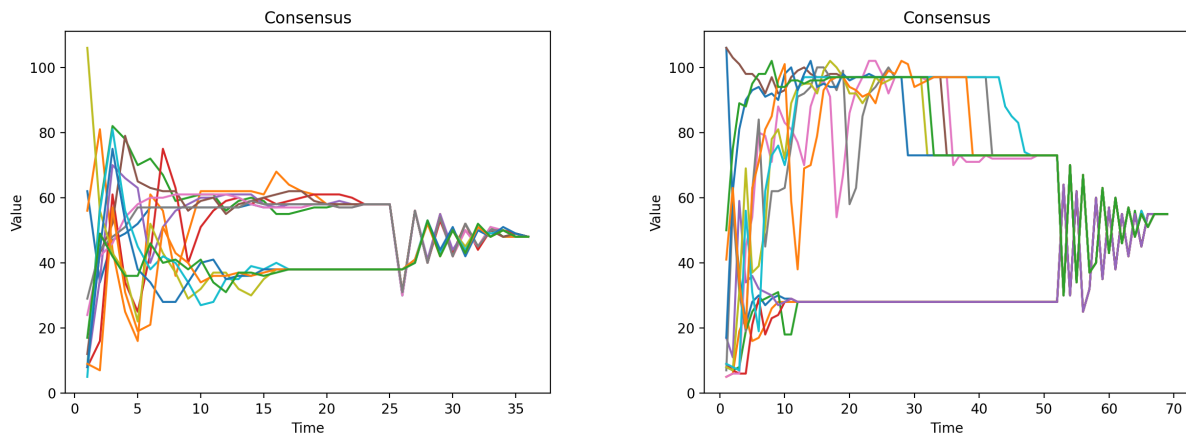


Figure 26: Results for topology with two clusters

Funding

This work was supported by the UNAM PASPA-DGAPA (Doctoral scholarship).

Author contributions

The authors contributed equally to this work.

Conflict of interest

The authors declare no conflict of interest.

References

- [1] Abolhasan, M. and Hagelstein, B. and Wang, J. C.-P.. In Search of an Understandable Consensus Algorithm, *Real-world performance of current proactive multi-hop mesh protocols*, 44-47, 2009.
- [2] Ian F. Akyildiz and Dario Pompili and Tommaso Melodia. (2005) Underwater acoustic sensor networks: research challenges, *Ad Hoc Networks*, 3(3),257-279, 2005.
- [3] Alotaibi, Eiman and Mukherjee, Biswanath. (2012) Survey Paper: A Survey on Routing Algorithms for Wireless Ad-Hoc and Mesh Networks, *Computer Networks*, 56(2), 940-965, 2012.
- [4] O. Arana and F. Garcia and J. Gomez.(2019) Analysis of the effectiveness of transmission power control as a location privacy technique, *Computer Networks*, 163, 1389-1286, 2019.
- [5] Alireza Askarzadeh. (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, *Computers & Structures*, 169, 1-12, 2016.
- [6] Atzori, Luigi and Girau, Roberto and Pilloni, Virginia and Uras, Marco. (2019). Assignment of Sensing Tasks to IoT Devices: Exploitation of a Social Network of Objects, *IEEE Internet of Things Journal*, 6(2), 2679-2692, 2019.
- [7] Benediktsson, J.A. and Swain, P.H. (1992). Consensus theoretic classification methods, *IEEE Transactions on Systems, Man, and Cybernetics*, 22(4), 688-704, 1992.
- [8] Borkar, Vivek and Varaiya, Pravin. (1982) Asymptotic agreement in distributed estimation, *IEEE transactions on automatic control*, 27(3), 650-655, 1982.
- [9] Byers, Charles C. and Wetterwald, Patrick. (2015) Fog Computing Distributing Data and Intelligence for Resiliency and Scale Necessary for IoT: The Internet of Things (Ubiquity Symposium), *Association for Computing Machinery, Ubiquity*, 14, 2015.
- [10] Cao, Li-Juan and Tay, Francis Eng Hock. (2003) Support vector machine with adaptive parameters in financial time series forecasting, *IEEE Transactions on neural networks*, 14(6), 1506-1518, 2003.
- [11] Carvin, Denis and Owezarski, Philippe and Berthou, Pascal, A generalized distributed consensus algorithm for monitoring and decision making in the iot *Proceedings of the 2014 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, 1-6, 2014.
- [12] Chissungo, Edmundo and Blake, Edwin and Le, Hanh. Investigation into Batman-adv Protocol Performance in an Indoor Mesh Potato Testbed, *Proceedings of the 2011 Third International Conference on Intelligent Networking and Collaborative Systems*, 8-13, 2011.
- [13] Cortes, Corinna and Vapnik, Vladimir. (1995) Support-Vector Networks, *Mach. Learn.*, 20(3), 273-297, 1995.
- [14] Luca Davoli and Antonio Cilfone and Laura Belli and Gianluigi Ferrari. (2019) Design and experimental performance analysis of a B.A.T.M.A.N.-based double Wi-Fi interface mesh network, *Future Generation Computer Systems*, 92, 593-603,2021.
- [15] Morris H. Degroot. (1974) Reaching a Consensus, *Journal of the American Statistical Association*, 69(345), 118-121, 1974.
- [16] Dhuli, Sateeshkrishna and Atik, Fouzul (2021). Analysis of Distributed Average Consensus Algorithms for Robust IoT network, *arXiv preprint arXiv:2104.10407*, 1, 2021.

- [17] Dimakis, Alexandros G. and Kar, Soummya and Moura, José M. F. and Rabbat, Michael G. and Scaglione, Anna. (2010) Gossip Algorithms for Distributed Signal Processing, *Proceedings of the IEEE*, 98(11), 1847-1864, 2010.
- [18] Drucker, Harris and Wu, Donghui and Vapnik, Vladimir N. (1999) Support vector machines for spam categorization, *IEEE Transactions on Neural networks*, 10(5), 1048–1054, 1999.
- [19] Chenyuan Feng, Howard H. Yang, Deshun Hu, Zhiwei Zhao, Tony Q.S., Geyong Min. (2021). Mobility-Aware Cluster Federated Learning in Hierarchical Wireless Networks, *IEEE Transaction on Wireless Communications*, 21(10), 8441-8857, 2022.
- [20] Flouri, K and Beferull-Lozano, Baltasar and Tsakalides, Panagiotis. Training a SVM-based classifier in distributed sensor networks, *Proceedings of the 2006, 14th European Signal Processing Conference*, 1–5, 2006.
- [21] Forero, Pedro A and Cano, Alfonso and Giannakis, Georgios B. (2010) Consensus-Based Distributed Support Vector Machines, *Journal of Machine Learning Research*, 163(5), 2010.
- [22] Funai, Colin and Tapparelo, Cristiano and Heinzelman, Wendi. (2020). Computational Offloading for Energy Constrained Devices in Multi-Hop Cooperative Networks, *Computational Offloading for Energy Constrained Devices in Multi-Hop Cooperative Networks*, 19(1), 60–73, 2020.
- [23] Goldsmith, Andrea (2008). Wireless Communications, *Cambridge University Press, 2005*
- [24] Guerraoui, Rachid and Rodrigues, Luis Introduction to reliable distributed programming *Springer Science & Business Media, 2006*
- [25] Guo, Wenzhong and Zhu, Weiping and Yu, Zhiyong and Wang, Jiangtao and Guo, Bin. (2019). A Survey of Task Allocation: Contrastive Perspectives From Wireless Sensor Networks and Mobile Crowdsensing, *IEEE Access*, 9, 78406-78420, 2019.
- [26] Guo, Chongtao and He, Wei and Li, Geoffrey Ye. (2021). Optimal Fairness-Aware Resource Supply and Demand Management for Mobile Edge Computing, *IEEE Wireless Communications Letters*, 18(3), 678-682, 2021.
- [27] Jiang, Zhiyuan and Zhou, Sheng and Guo, Xueying and Niu, Zhisheng. (2018). Task Replication for Deadline-Constrained Vehicular Cloud Computing: Optimal Policy, Performance Analysis, and Implications on Road Traffic, *IEEE Internet of Things Journal*, 5(1), 93-107, 2018.
- [28] Yuchuan Jiang and Zhangjun Wang and ZhiXiong Jin. Iot Data Processing and Scheduling Based on Deep Reinforcement Learning, *International journal of computers communications and control*, 18(3),1-8,2023
- [29] Muhammad Altaf Khan and et al. (2021) A Survey on the Noncooperative Environment in Smart Nodes-Based Ad Hoc Networks: Motivations and Solutions, *Security and Communication Networks*, 2021, 2021.
- [30] Kocarev, Ljupco (2013). Consensus and synchronization in complex networks, *Springer, 2013*
- [31] Kowalczyk Alexandre (2017). Support Vector Machines Succinctly *Synconfusion, 2017*
- [32] Elis Kulla and Masahiro Hiyama and Makoto Ikeda and Leonard Barolli. (2012) Performance comparison of OLSR and BATMAN routing protocols by a MANET testbed in stairs environment, *Computers & Mathematics with Applications*, 63(2), 339-349, 2012.
- [33] Lamport, Leslie. (1998) The Part-Time Parliament, *ACM Trans. Comput. Syst.*, 16(2), 133–169, 1998.
- [34] Lamport, Leslie. (2001) Paxos Made Simple, *ACM SIGACT News (Distributed Computing Column)*, 32(4), 51-58, 2001.

- [35] Koen Langendoen and Niels Reijers. (2003) Distributed localization in wireless sensor networks: a quantitative comparison, *Computer Networks*, 43(4), 499-518, 2003.
- [36] Li, Shutao and Kwok, JT-Y and Tsang, IW-H and Wang, Yaonan. (2014) Fusing images with different focuses using support vector machines, *IEEE Transactions on neural networks*, 15(6), 1555–1561, 2004.
- [37] Lin, Zhiyun and Francis, Bruce and Maggiore, Manfredi. (2007) State Agreement for Continuous-Time Coupled Nonlinear Systems, *SIAM Journal on Control and Optimization*, 46(1), 288-307, 2007.
- [38] Lin, J. and Morse, A. S. and Anderson, B. D. O. (2007) The Multi-Agent Rendezvous Problem. Part 1: The Synchronous Case, *SIAM Journal on Control and Optimization*, 46(6), 2096-2119, 2007.
- [39] Lin, J. and Morse, A. S. and Anderson, B. D. O. (2007) The Multi-Agent Rendezvous Problem. Part 2: The Asynchronous Case, *SIAM Journal on Control and Optimization*, 46(6), 2120-2147, 2007.
- [40] Liu, Ji and Mou, Shaoshuai and Morse, A. Stephen and Anderson, Brian D. O. and Yu, Changbin. (2011). Deterministic Gossiping, *Proceedings of the IEEE*, 99(9), 1505-1524, 2011.
- [41] Liu, Hui and Cao, Ming and Wu, Chai Wah. New spectral graph theoretic conditions for synchronization in directed complex networks *Proceedings 2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2307-2310, 2013.
- [42] Liu, Ligang and Liu, Jianpo and Qian, Hanwang and Zhu, Jun. (2018). Performance Evaluation of BATMAN-Adv Wireless Mesh Network Routing Algorithms, *Proceedings of the 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 122-127, 2018.
- [43] Liu, Jianhui and Zhang, Qi. (2018). Offloading Schemes in Mobile Edge Computing for Ultra-Reliable Low Latency Communications, *IEEE Access*, 6, 12825-12837, 2018.
- [44] Lu, Yumao and Roychowdhury, V. and Vandenberghe, L. (2008) Distributed Parallel Support Vector Machines in Strongly Connected Networks, *Trans. Neur. Netw.*, 19(7), 1167–1178, 2008.
- [45] Lynch, Nancy A. (1996) Distributed Algorithms, *Morgan Kaufmann Publishers Inc., 1996*
- [46] Magadán, Luis and Suárez, Francisco J. and Granda, Juan C. and García, Daniel F. (2022). Clustered WSN for Building Energy Management Applications *Proceedings of the Science and Technologies for Smart Cities*, 673-687, 2022.
- [47] Malik, Sehrish and Ahmad, Shabir and Ullah, Israr and Park, Dong Hwan and Kim, DoHyeun. (2019). An Adaptive Emergency First Intelligent Scheduling Algorithm for Efficient Task Management and Scheduling in Hybrid of Hard Real-Time and Soft Real-Time Embedded IoT Systems, *Sustainability*, 11(8), 2192-2203, 2019.
- [48] V. Muthukumaran and R. Sivakami and V. K. Venkatesan and J. Balajee. and T. R. Mahesh and E. Mohan and B. Swapna. Optimizing Heterogeneity in IoT Infra Using Federated Learning and Blockchain-based Security Strategies, *International journal of computers communications and control*, 18(6),1-21, 2023.
- [49] Nepusz, Tamás and Vicsek, Tamás. (2012). Controlling edge dynamics in complex networks, *Nature Physics*, 8(7), 568-573, 2012.
- [50] Newman, M. E. J. Networks: an introduction *Oxford University Press, 2010*

- [51] Ernesto Nunes and Marie Manner and Hakim Mitiche and Maria Gini. (2017). A taxonomy for task allocation problems with temporal and ordering constraints, *Robotics and Autonomous Systems*, 90, 55-70, 2017.
- [52] Felipe Núñez and Yongqiang Wang and David Grasing and Sachi Desai and George Cakiades and Francis J. Doyle. (2017) Pulse-coupled time synchronization for distributed acoustic event detection using wireless sensor networks, *Control Engineering Practice*, 60, 106-117, 2017.
- [53] Olfati-Saber, Reza and Fax, J. Alex and Murray, Richard M. (2007). Consensus and Cooperation in Networked Multi-Agent Systems, *Proceedings of the IEEE*, 95(1), 215-233, 2007.
- [54] [Online].Available: www.open-mesh.org/, Accessed on 3 February 2024.
- [55] [Online].Available: datatracker.ietf.org/doc/html/draft-wunderlich-openmesh-manet-routing-00, Accessed on 3 February 2024.
- [56] [Online].Available: www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept, Accessed on 3 February 2024.
- [57] [Online].Available: dl.acm.org/ccs, Accessed on 3 February 2024.
- [58] [Online].Available: mdotfernandez.wordpress.com/, Accessed on 3 February 2024.
- [59] [Online].Available: doi.org/10.6028/NIST.SP.500-325, Accessed on 3 February 2024.
- [60] [Online].Available: www.rfidjournal.com/articles/view?4986, Accessed on 3 February 2024.
- [61] [Online].Available: repositorio.unam.mx/contenidos/84523, Accessed on 3 February 2024.
- [62] [Online].Available: doi.org/10.6028/NIST.SP.800-145, Accessed on 3 February 2024.
- [63] Germán A. Montoya and Carlos Lozano-Garzón and Yezid Donoso. (2022). A Stochastic Mobility Prediction Algorithm for finding Delay and Energy Efficient Routing Paths considering Movement Patterns in Mobile IoT Networks, *International journal of computers communications and control*, 17(4),1-9, 2022.
- [64] [Online].Available: doi.org/10.48550/arxiv.2001.07704, Accessed on 3 February 2024.
- [65] Oróstica, Boris and Núñez, Felipe. (2019). Robust Gossiping for Distributed Average Consensus in IoT Environments, *IEEE Access*, 7, 994-1005, 2019.
- [66] Pilloni, Virginia and Atzori, Luigi and Mallus, Matteo. (2017). Dynamic Involvement of Real World Objects in the IoT: A Consensus-Based Cooperation Approach, *Sensors*, 17(3), 484-512, 2017.
- [67] Virginia Pilloni and Luigi Atzori. (2017). Consensus-based resource allocation among objects in the internet of things, *Annals of Telecommunications*, 72, 415-429, 2017.
- [68] Pilloni, Virginia and Ning, Huansheng and Atzori, Luigi. (2022). Task Allocation Among Connected Devices: Requirements, Approaches, and Challenges, *IEEE Internet of Things Journal*, 9(2), 1009-1023, 2022.
- [69] Preciado, Victor M and Verghese, George C, Synchronization in generalized erd ő sr é nyi networks of nonlinear oscillators, *Proceedings of the 44th IEEE Conference on Decision and Control*, 4628–4633 2005.
- [70] Ren, Wei and Beard, Randal W (2008). Distributed consensus in multi-vehicle cooperative control, *Springer*, 27(2), 2008
- [71] Sahni, Yuvraj and Cao, Jiannong and Zhang, Shigeng and Yang, Lei. (2017). Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things, *IEEE Access*, 5, 16441-16458, 2017.

- [72] Li, Shancang and Oikonomou, George and Tryfonas, Theo and Chen, Thomas M. and Xu, Li Da. (2014). A Distributed Consensus Algorithm for Decision Making in Service-Oriented Internet of Things, *IEEE Transactions on Industrial Informatics*, 10(2), 1461-1468, 2014.
- [73] Shaocheng Luo and Jonghoek Kim and Ramviyas Parasuraman and Jun Han Bae and Eric T. Matson and Byung-Cheol Min. (2019). Multi-robot rendezvous based on bearing-aided hierarchical tracking of network topology, *Ad Hoc Networks*, 86(3), 131-143, 2019.
- [74] Schölkopf, Bernhard and Smola, Alexander J and Bach, Francis and others (2002). Learning with kernels: support vector machines, regularization, optimization, and beyond, *MIT press*, 2002
- [75] Liu, Jianhui and Zhang, Qi. (2020). Jin Sun and Lu Yin and Minhui Zou and Yi Zhang and Tianqi Zhang and Junlong Zhou, *Journal of Systems Architecture*, 108, 101799, 2020.
- [76] Sunyaev, Ali. (2020). Internet computing: Principles of Distributed systems and emerging internet-based technologies, *Springer Nature*, October, 2020, 248-249, 2020.
- [77] Youcef Touati and Arab Ali-Chérif and Boubaker Daachi. (2017) 4 - Adaptive Routing for Large-Scale WSNs, *Energy Management in Wireless Sensor Networks*, 53-63, 2017.
- [78] Tsitsiklis, J. and Bertsekas, D. and Athans, M. (1986) Distributed asynchronous deterministic and stochastic gradient optimization algorithms, *IEEE Transactions on Automatic Control*, 31(9), 803-812, 1986.
- [79] Velázquez-Mena, Alejandro and Benítez-Pérez, Héctor and Rodríguez-Martínez, Rita C. and Villarreal-Martínez, Ricardo F. Design and evaluation of indoor wireless ad hoc network using BATMAN-adv with mobile robots, *Proceedings of the 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, 376-383, 2022.
- [80] Wang, Xiaofan and Wang, Xiaoling. Consensus of edge dynamics on complex networks, *Proceedings 2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1271-1274, 2014.
- [81] Wang, Liang and Yu, Zhiwen and Han, Qi and Guo, Bin and Xiong, Haoyi. (2018). Multi-Objective Optimization Based Allocation of Heterogeneous Spatial Crowdsourcing Tasks, *IEEE Transactions on Mobile Computing*, 17(1), 1637-1650, 2018.
- [82] Jianbin and Wang, Zesen and Zhang, Yonggang and Wang, Lu. (2020). Allocating Heterogeneous Tasks in Participatory Sensing with Diverse Participant-Side Factors, *IEEE Transactions on Mobile Computing*, 18(9), 1979-1991, 2019.
- [83] Weller, S C and Mann, N C. (1997) Assessing rater performance without a “gold standard” using consensus theory, *Med Decis Making*, 17(7), 71–79, 1997.
- [84] Yu, Wenwu and Chen, Guanrong and Cao, Ming and Ren, Wei. (2013). Delay-Induced Consensus and Quasi-Consensus in Multi-Agent Dynamical Systems, *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(10), 2679-2687, 2013.
- [85] Lin Xiao and Stephen Boyd. (2004). Fast linear iterations for distributed averaging, *Systems & Control Letters*, 53(1), 65-78, 2004.
- [86] Jianbin and Wang, Zesen and Zhang, Yonggang and Wang, Lu. (2020). Xiang Yin and Kaiquan Zhang and Bin Li and Arun Kumar Sangaiah and Jin Wang, *International Journal of Distributed Sensor Networks*, 14(8), 1550147718795355, 2018.
- [87] Xu, Lina and Collier, Rem and O’Hare, Gregory M. P. (2017) A Survey of Clustering Techniques in WSNs and Consideration of the Challenges of Applying Such to 5G IoT Scenarios, *IEEE Internet of Things Journal*, 4(5), 1229-1249, 2017.

- [88] Xu, Mengying and Zhou, Jie. (2020). Elite Immune Ant Colony Optimization-Based Task Allocation for Maximizing Task Execution Efficiency in Agricultural Wireless Sensor Networks, *Journal of Sensors*, 2020(1), 3231864, 2020.
- [89] Xue, Jianbin and Wang, Zesen and Zhang, Yonggang and Wang, Lu. (2020). Task Allocation Optimization Scheme Based on Queuing Theory for Mobile Edge Computing in 5G Heterogeneous Networks, *Mobile Information Systems*, 150-1403, 2020.
- [90] Zhengxin Yu, Jia Hu, Geyong Min, Wang Miao, Shamin Hossain M. (2021). Mobility-Aware Proactive Edge Caching for Connected Vehicles Using Federated Learning, *IEEE Transaction of Intelligent Transportation Systems*, 22(8), 5341-5351, 2021.
- [91] Lu, Yumao and Roychowdhury, Vwani and Vandenberghe, Lieven. (2008) Distributed Parallel Support Vector Machines in Strongly Connected Networks, *IEEE Transactions on Neural Networks*, 19(7), 1167-1178, 2008.
- [92] Yunmin Zhu and Li, X.R. and Zhisheng You. Unified fusion rule in multisensor network decision systems, *Proceedings of the Third International Conference on Information Fusion*, 2000.
- [93] Zhang, Guanglin and Zhang, Wenqian and Cao, Yu and Li, Demin and Wang, Lin. (2018). Energy-Delay Tradeoff for Dynamic Offloading in Mobile-Edge Computing System With Energy Harvesting Devices, *IEEE Transactions on Industrial Informatics*, 16(18), 4642-4655, 2018.
- [94] Li, Zhongkui and Duan, Zhisheng and Chen, Guanrong and Huang, Lin. (2010). Consensus of Multiagent Systems and Synchronization of Complex Networks: A Unified Viewpoint, *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(1), 213-224, 2010.
- [95] Breno Costa, Joao Bachiega Jr, Leonardo Rebouças de Carvalho, and Aleteia PF Araujo. Orchestration in fog computing: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 55(2):1-34, 2022.
- [96] Nidhi Kumari, Anirudh Yadav, and Prasanta K Jana. Task offloading in fog computing: A survey of algorithms and optimization techniques. *Computer Networks*, 214:109137, 2022.
- [97] Kimchai Yeow, Abdullah Gani, Raja Wasim Ahmad, Joel JPC Rodrigues, and Kwangman Ko. Decentralized consensus for edge-centric internet of things: A review, taxonomy, and research issues. *IEEE Access*, 6:1513-1524, 2017.
- [98] Kalimullah Lone and Shabir Ahmad Sofi. e-toalb: An efficient task offloading in iot-fog networks. *Concurrency and Computation: Practice and Experience*, 36(6):e7951, 2024.
- [99] Amit Kishor and Chinmay Chakarbarty. Task offloading in fog computing for using smart ant colony optimization. *Wireless personal communications*, 127(2):1683-1704, 2022.
- [100] Kai Lin, Sameer Pankaj, and Di Wang. Task offloading and resource allocation for edge-of-things computing on smart healthcare systems. *Computers & Electrical Engineering*, 72:348-360, 2018.



Copyright ©2024 by the authors. Licensee Agora University, Oradea, Romania.

This is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International License.

Journal's webpage: <http://univagora.ro/jour/index.php/ijccc/>



This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).

<https://publicationethics.org/members/international-journal-computers-communications-and-control>

Cite this paper as:

Velázquez-Mena, A.; Benítez-Pérez, H.; Rodríguez-Martínez R.; Villarreal-Rodríguez R.(2024). DICOMIST: An methodology for Performing Distributed Computing in Heterogeneous *ad hoc* Networks, *International Journal of Computers Communications & Control*, 19(4), 6526, 2024.

<https://doi.org/10.15837/ijccc.2024.4.6526>