Mobile Networks and Applications
https://doi.org/10.1007/s11036-020-01630-3

# *Neighborhood-aware Mobile Hub*: An Edge Gateway with Leader Election Mechanism for Internet of Mobile Things

**Marcelino Silva[1] · Ariel Teles[2] · Rafael Lopes[1] · Francisco Silva[1] · Davi Viana[1] · Luciano Coutinho[1] · Nishu Gupta[3] · Markus Endler[4]**



**Abstract**

Internet of Things (IoT) is the interconnection of thousands of heterogeneous addressable smart objects (i.e., devices embedded with sensors and actuators) with Internet connectivity. Internet of Mobile Things (IoMT) is characterized by considering the mobility of smart objects. For managing smart objects, it is necessary to provide a middleware. Mobile Hub (M-Hub) is an IoT middleware that collects, processes and distributes data from a large number of smart objects on the edge of the network. M-Hub runs on mobile devices, enabling them to be gateways. It represents an autonomous entity, able to detect a set of objects available in the neighborhood and to monitor them independently of other M-Hubs. Hence, in some situations it may happen that a same object is eligible to be monitored by several M-Hubs. In this context, this paper proposes Neighborhood-aware M-Hub (NAM-Hub), a leader election mechanism integrated to the M-Hub to determine a suitable gateway for each smart object discovered opportunistically. It considers context data gathered from the mobile device to dynamically elect leaders (i.e., a leader and a sub-leader). The proposed solution contributes to take advantage from the resources provided for the mobile gateway and avoids their wastage. The proposed leader election mechanism was tested and evaluated considering its performance and the results were promising, with short detection time and recovery time in the system.

**Keywords** Internet of mobile things · Middleware · Leader election · Mobile smart objects

## 1 Introduction

Internet of Things (IoT) can be defined as the interaction of technologies from different areas, such as ubiquitous computing, context awareness, communication protocols and technologies and computing devices with embedded sensors and actuators, also known as smart objects [8]. The interconnection of thousands of heterogeneous addressable objects with network connectivity allows them to collect and share data about the environment where they are located. These objects can be used to provide services in a variety of contexts [7, 27], such as public and private transportation management, power management and consumption, healthcare, public spaces for entertainment and social activities.

Due to memory and processing constraints, some smart objects do not have connectivity to medium and long range networks. They do not implement a Transfer Control Protocol/Internet Protocol (TCP/IP) stack. As a result, they cannot access the Internet through their own resources [15]. Additionally, IPv6 does not solve all issues present in the IoT environments [7], because management, addressing and routing are its major challenges. IP-based protocols are not supported by the vast majority of smart objects and their services are not suitable for most IoT applications. The reason is that IP-based protocols are intrinsically designed for intensive work cycles, large data flows and reliability while IoT communications involve small and frequent messages. In IoT communications, each message individually is not much important but the corresponding data flows carry relevant pieces of information [15]. Otherwise, smart objects are able to send their context data to an edge gateway [28], using short-range communication technologies such as Bluetooth and ZigBee. Upon receiving

✉ Ariel Teles
ariel.teles@ifma.edu.br

1 Federal University of Maranhão, MA, Brazil

2 Federal Institute of Maranhão, MA, Brazil

3 Vaagdevi College of Engineering, Warangal, India

4 Pontifícia Universidade Católica do Rio de Janeiro, RJ, Brazil

Published online: 01 September 2020

this data, the gateway is in charge of aggregating and filtering it, performing data fusion and forwarding useful information to the Internet, usually to the cloud side of the application [40]. Moreover, there is a trend to push the artificial intelligence (AI) frontier from the cloud to the network edge. Thus, AI techniques can be embedded in the gateway to generate knowledge [46, 48].

An IoT paradigm extension is the so-called Internet of Mobile Things (IoMT) [31, 39], which conceive situations where smart objects and IoT gateways can move or be moved with greater flexibility. Wearable and portable devices, robots and vehicles are some examples of moving objects. In IoMT scenarios, personal smartphones and tablets can act as mobile edge gateway, promoting discovery and opportunistic connections with smart objects around them.

Mobile Hub (M-Hub) [44] is a general-purpose middleware service that enables conventional mobile personal device (i.e., Android smartphones and tablets) to become propagator nodes (i.e., mobile edge gateways) for data produced by smart objects, either stationary or mobile (M-Obj). These objects are usually provided only with short-range wireless network interfaces. By considering this limitation, M-Hub opportunistically discovers, registers and enables communication to M-Objs.

M-Hub represents an autonomous edge gateway able to detect the presence of a set of objects available in its proximity and monitor them independently of other M-Hubs in its neighborhood. However, such condition can lead to wastage of communication, processing and energy resources since the same set of objects might be monitored by several M-Hubs. Hence, a M-Hub may be overwhelmed by being responsible for a large number of objects compared to other ones available in the environment.

To illustrate the problem addressed in this research, we define a scenario of mobile crowdsensing [22] in a smart city [41]. In such a scenario, individuals collectively share data and extract information to measure and map the phenomena of common interest [33]. Moreover, there may be concentrations of hundreds of devices at the same geographical area. Consequently, there might be a high density of M-Hubs and M-Objs establishing connections and forwarding data. For instance, people in a park with sensors in wearable devices sharing data in a paradigm of participatory or opportunistic sensing [32], and smartphones of pedestrians/bikers playing the role of M-Hubs. This is an example in which there can be more than one M-Hub monitoring the same M-Obj and at the same time, devices acting as M-Hub and M-Obj, both mobile.

By considering such situations, this paper proposes a mechanism that embeds a leader election algorithm into the M-Hub. This new version of the M-Hub is named *Neighborhood-aware M-Hub* (NAM-Hub). Proposed method enables the communication and negotiation between a group of co-located M-Hubs (i.e., its neighborhood) to determine the most suitable mobile edge gateways available in the IoMT environment for each opportunistically discovered smart object. The contribution of this paper is threefold: (1) we propose and describe the NAM-Hub. The NAM-Hub is an improvement of M-Hub that provides communication between co-located M-Hubs to orchestrate their actions; (2) we illustrate a test scenario to demonstrate the feasibility of practical use; and (3) we show results from a performance evaluation performed with the proposed solution regarding detection time and recovery time.

The remainder of this paper is structured as follows. Section 2 presents the original version of the M-Hub, details our research problematic, and discusses related works. In Section 3, we describe the proposed leader election mechanism, while in Section 4 we provide a practical demonstration in testing the NAM-Hub for IoMT applications. Section 5 shows the results obtained from experiments performed to evaluate the performance of the proposed mechanism. In Section 6, we discuss the proposed solution. At the end, Section 7 contains the conclusion and future research.

## 2 Background

### 2.1 The Mobile Hub

The knowledge about the concepts and functioning of the M-Hub components is fundamental to understand the solution proposed in this work. M-Hub is a general-purpose middleware service, executed in conventional personal mobile devices (e.g., smartphones, tablets) as a background application, allowing them to become IoMT edge gateways. M-Hub can discover, register, and enable unicast, broadcast and group-cast mode communication with many types of M-Objs through the *Scalable Data Distribution Layer* (SDDL) [16, 19] or the *Context Data Distribution Layer* (CDDL) [25] middleware solutions. The main components of the M-Hub are shown in Fig. 1, which is described next.

*Short-Range Sensor, Presence and Actuation* (S2PA) is responsible for the discovery, connection and communication with M-Objs. This component defines an Application Programming Interface (API) that provides an abstraction for communication with M-Objs that have short-range wireless network interfaces. For the exchange of data with smart objects, use of an object driver is required which is usually provided by the manufacturer. *Location Service* is responsible for registering the current position of the M-Hub and attaching it to messages transmitted to the cloud or other M-Hubs. *Connection Service* is implemented in two versions. The first one using *ClientLib*, a library that provides direct,

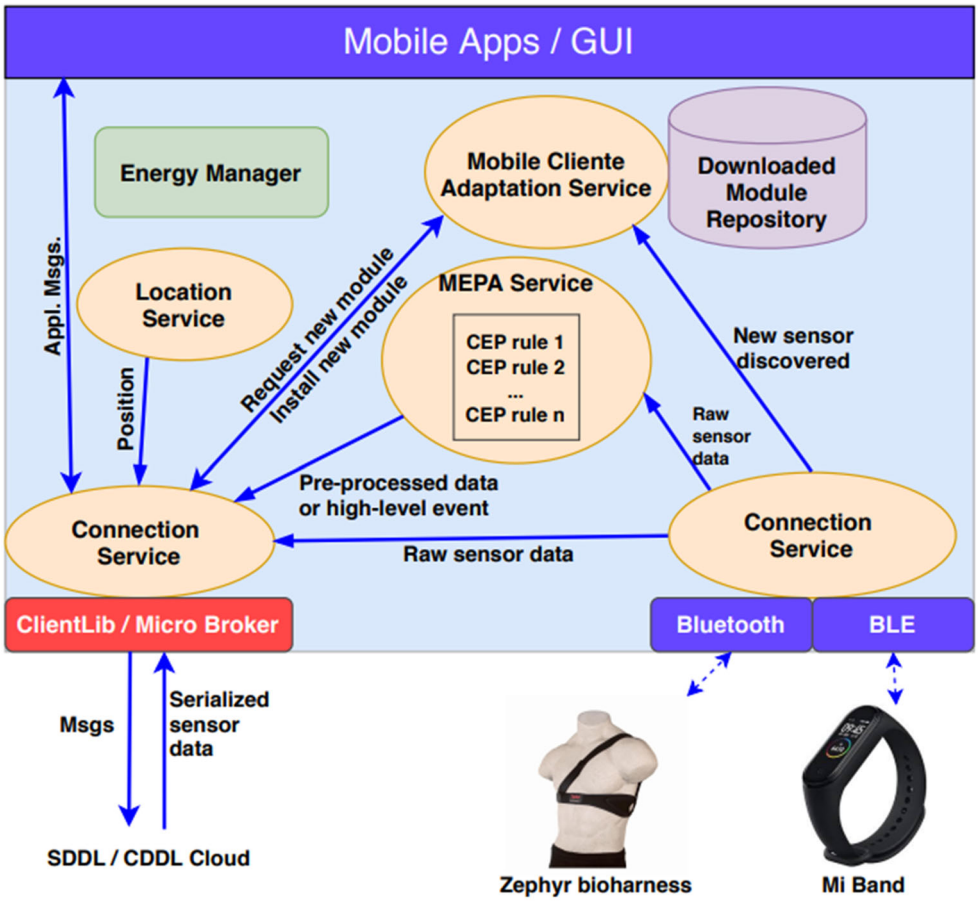


**Fig. 1** M-Hub Architecture

group and publish-subscribe communication paradigms for mobile nodes. It extends the *Mobile Reliable UDP* (MR-UDP) protocol [16] with mobility-tolerating features such as handover and Firewall/NAT transversal. The second one, as a *Micro Broker* of the CDDL [25], a soft version of a MQTT broker. *Mobile Event Processing Agent* (MEPA) service allows continuous analysis of data flow originated from smart objects to identify patterns of interest which can be defined by the mobile application. MEPA uses *Complex Event Processing* (CEP) [14] implemented with *Esper* [1] to process event flows in real time and find patterns through rules.

The device's energy level (classified as low, medium, high) influences the frequency and duration of actions performed by all the aforementioned components. This is set by the *Energy Manager* component, which collects samples of the mobile device's battery level and checks if it is connected to a power source.

Regarding Wireless Personal Area Networks (WPANs), M-Hub supports connections with smart objects via Bluetooth Classic (2.0 and 3.0) and Bluetooth Low Energy (BLE 4.0). Despite higher power consumption, Bluetooth Classic is still used by a wide variety of IoT devices [43], including healthcare such as Zephyr BioHarness 3 and Zephyr HxM [6]. BLE is emerging as a very promising technology because it is more energy-efficient and also enables fast discovery of peripheral devices. In addition, BLE supports about 2,500 concurrent connections; it is available in a growing number of smartphone models and is being incorporated into a growing variety of IoT [43] and wearable devices, such as Mi Band [4].

## 2.2 Problem Statement

Essentially, the IoMT paradigm considers any situation in which the relative position and speed among M-Objs and M-Hubs are variable and can change anytime. The variation of the device's position in the environment allows M-Objs to be simultaneously discovered by different M-Hubs. Therefore, IoMT encompasses mobility scenarios in which:

1. Smart objects are intentionally deployed in a stationary way and the M-Hub is mobile and can opportunistically interact with them;
2. M-Objs are mobile or attached to a mobile stuff (e.g., wearable devices, clothes, vehicles), while the M-Hub is purposely linked to a stationary environment;
3. M-Objs and M-Hub are both mobile.

In that last scenario, a M-Obj can establish a connection with a M-Hub by being co-localized or in co-movement for a certain period of time. These IoMT scenarios in which objects and edge gateways are essentially mobile, motivates the use of portable devices as natural candidates to act as edge gateways too.

The problem addressed in this paper involves the coordination of a group of co-located independent mobile devices (i.e., the neighborhood) to determine the most suitable edge gateways (i.e., a leader and a sub-leader) available in the IoMT environment for each smart object discovered opportunistically. The approach relies on avoiding overloading a M-Hub by monitoring various smart objects while other ones are also available to become a mobile edge gateway in the environment. Moreover, when using Bluetooth Classic an object may have more than one gateway, so the proposed solution avoids the redundancy of data collected from different smart objects by the same mobile edge gateways and, consequently, minimizing the consumption of communication, processing and energy resources of the mobile device [12].

### 2.3 Related Work

Election in distributed computing systems is a traditional topic of research [23], including proposals for mobile networks [45], but it is still one of the important problems faced by those systems [24], since paradigms (e.g., IoT and IoMT) with new requirements have been emerging (e.g., cluster head election in WSNs [18, 35] and MANETs [42]). Although there are several proposals for different application scenarios, in this section we focus on describing related works that proposed leader election mechanisms that can be applied to IoT.

Zhang et al. [47] initially showed why some traditional leader election methods cannot be originally used in IoT due to dynamic nature of things. They also proposed a leader election method to select nodes to coordinate other ones in IoT environments. The method performs the selection based on classification of nodes according to their features (e.g., ability of being identified or self-coordination) and weights attributed to these ones. Fernández-Campusano et al. [21] introduced a leader election algorithm for partially synchronous systems based on a punishment which is calculated for every candidate based on failures and disconnections.

De Masi [36] introduced a peer-to-peer load balancing scheme to distribute tasks to be processed by the leader in a smartphone cluster, a solution called *LoadIoT*. In a group of peer mobile smart devices, *LoadIoT* periodically elects a leader by using a list of input parameters such as residual energy (i.e., percentage of remaining energy), battery status (e.g., loading, discharging, critical), processing load, and number of running tasks. Drãgan et al. [17] proposed approaches to elect leaders in opportunistic networks in which each node announces its application that is used by interested receiver nodes to calculate a score. The node with the highest score becomes the leader. To define the score, the solution uses different characteristics such as trust, centrality, contact probability, and latency.

Bounceur et al. [9, 10] proposed the BROGO algorithms for leader election in wireless sensor and IoT networks. The proposal works with asynchronous networks without making any assumption on the topology and they require a root node (i.e., an initial reference node) to create a list of leaders. Faika et al. [20] presented a solution applied to battery management systems in which a leader election algorithm is proposed to select a battery module (i.e., the leader) that is in-charge of collecting the data from all other modules to know the overall status of the battery pack.

As turns out from the above, several parameters were used to determine the most suitable leader in the scenarios explored in each one of these described works. In addition, they contribute to different types of scenarios with specific requirements. However, despite the variety of functionalities presented by the analyzed solutions, they were not designed for providing capabilities that enable the election of leaders and sub-leaders considering characteristics of IoMT environments, as described in Section 2.2.

## 3 The Neighborhood-aware M-Hub

Our solution is an extension of the M-Hub, which is named *Neighborhood-aware M-Hub* (NAM-Hub). It aims at providing communication between co-located M-Hubs to coordinate their actions to optimize the gathering and distribution of data produced by M-Objs. NAM-Hub has a communication and coordination mechanism that implements a leader election distributed algorithm for selecting the most suitable leader M-Hubs based on a set of parameters related to their available computational resources and the wireless network signal strength with M-Objs.

### 3.1 Architecture

The architecture (Fig. 2) contains the proposed components for supporting the communication and coordination of M-Objs. The M-Hub original version has already been discussed in Section 2 and we detail here the new infrastructure components of the proposed solution.

– *Object Container* is a repository of objects discovered by the M-Hub. This module manages objects by updating data of an object that is already present in the container and inserting new ones if they are not

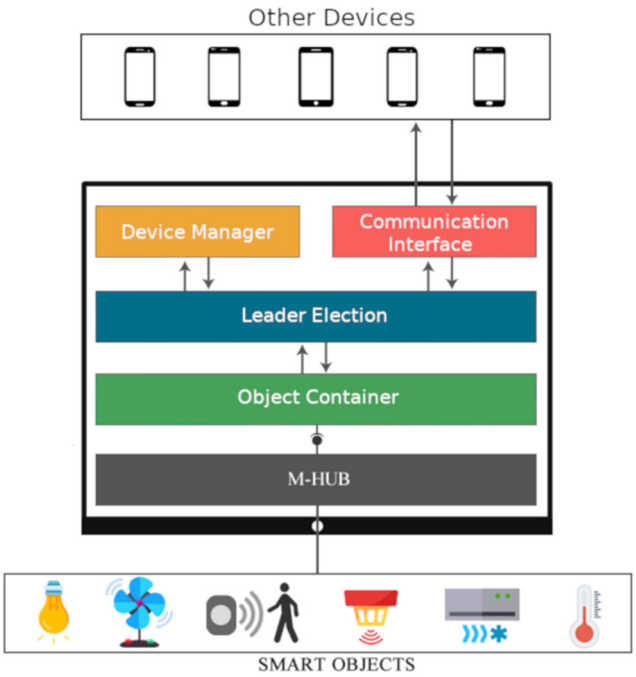


**Fig. 2** NAM-Hub Architecture

in the container. This component is also responsible for removing those objects whose data has not been received for a certain period of time (default time is 5 seconds) which implies that the M-Hub is no longer in the transmission area of an object in particular. It is important to note that new components proposed for the M-Hub are implemented as new software layers. For this reason, the original version of the M-Hub remains unchanged and all communications from it to the *Object Container* component are made via predefined interface;

- *Leader Election* is a module composed by several components that interact to allow the M-Hub to reach a decision about whether or not to participate in a leader election process and if it should start a process based on the current state of the system. The core of this module is the leader election algorithm. This module is responsible for performing actions such as requesting mobile device information from the *Device Manager* component and combining it with the object data to perform the calculation of the *Score* (described in Section 3.3) and communication with other M-Hubs via *Communication Interface* component using predefined messages (described in Section 3.4). In addition, it verifies if the *Object Container* has the objects contained in the messages exchanged with other M-Hubs;
- *Device Manager* provides information from the mobile device (e.g., battery remains, CPU load) where the M-Hub is running. The device manager aims to send data to the *Leader Election* module. Such data contains the parameters needed to calculate the *Score* used to define the leader during the election process;
- *Communication Interface* allows a M-Hub to communicate with other ones by exchanging a set of messages. Communication between M-Hubs is currently accomplished only through WiFi networks. This component is extensible to have added other communication channels in future developments.

### 3.2 Component Interactions

The interaction between architecture components is illustrated in the sequence diagram in Fig. 3.

Firstly, *S2PA* component, which belongs to the M-Hub original version (see Section 2.1), initializes the implemented technologies (i.e., Bluetooth technology). Then it starts to periodically trigger the M-Objs discovery process. It defines some basic methods and interfaces that should implement: 1) M-Obj discovery and connection; 2) discovery of services provided by each M-Obj; 3) reading and writing of service attributes (e.g., sensor values, actuator commands); and 4) notifications about disconnections of M-Objs. When a M-Hub receives a message from a M-Obj, S2PA extracts the information, such as the Universally Unique Identifier (UUID) and the Received Signal Strength Indicator (RSSI), and sends it to the *Object Container* (Step 1. - method *sensorData()*), which adds or updates the M-Obj information (Step 2. - method *addOrUpdate()*) in the container. *Leader Election* module periodically analyzes the objects contained in the *Object Container* (Step 3. - method *analizeSensorData(listening)*). Hence, it can decide which message should be transmitted to another M-Hub nearby (Step 4. - method *checkLeader(M-HubID)*). When it is required to calculate the *Score* for a new election process, *Device Manager* is requested and it sends device information (e.g., battery level and CPU load) to the *Leader Election* (Step 5. - method *checkDeviceStatus()*). Finally, messages are transmitted and received by the *Communication Interface* (Step 6. - method *sendBroadcast(M-HubID, M-ObjIDs)*).

### 3.3 Assumptions

Some assumptions are taken into account by the proposed solution. Fundamentally, M-Hubs must communicate by exchanging messages with each other. Regarding time, a partially synchronous system is assumed (i.e., there are upper limits on processing time and delay of message communication). Additionally, we consider the following assumptions about the nodes and the system architecture:

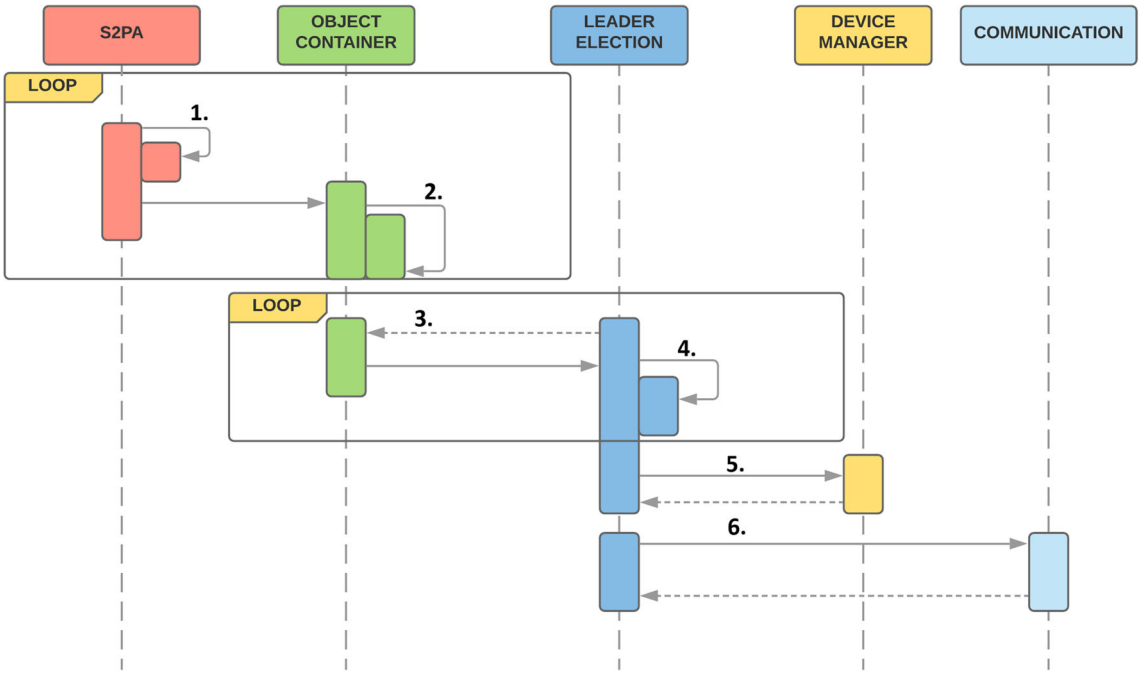


**Fig. 3** Component interactions in the NAM-Hub

- Unique Node IDs: all nodes have unique identifiers. They are used to identify participants (either M-Hub nodes or smart objects) during the election process;
- *Score*: each node has an associated priority value: the *Score*. The priority of a node indicates its attractiveness to become a leader and considers performance-related attributes such as battery level, computational capacities, and wireless network signal strength;
- Heterogeneous: nodes (M-Hubs and M-Objs) can be heterogeneous in relation to performance-related attributes and can be stationary and mobile (that is, M-Objs). Regarding the M-Hub, its performance-related configurations may vary because it runs on conventional personal mobile devices (e.g., smartphones, tablets) of different manufacturers and models. Likewise, M-Obj specifications may vary however they usually "may possess means to sense physical phenomena (e.g., temperature, light, electromagnetic radiation level) or to trigger actions having an effect on the physical reality (actuators)" [38];
- Connectivity: all M-Hubs are connected to a bidirectional communication link, the same wireless network, and communicate with each other via SDDL or CDDL. M-Objs are connected to M-Hubs using short-range communication technologies (Bluetooth Classic and BLE);
- Mobility: the mobility of nodes may result in arbitrary topologies, including partitioning and grouping. In addition, nodes and objects may fail at any time. Failures may be caused by: (1) communication problems (e.g., disconnection, weak or intermittent wireless connectivity), and (2) hardware components (i.e., communication antenna defects, power outages that cause shutdown). Faults may be permanent but M-Hubs and objects can also recover from a failure.

### 3.4 Description of Leader Election Process

There are three types of distributed messages used to execute the leader election process. They are described below.

- *ELECTION*: message is used for two purposes: to inform the start ($E_S$) of a new leader election process and to reply ($E_R$) an $E_S$ message. The $E_S$ message has two parameters: *MHubID*, a UUID string generated each time the M-Hub application starts; and *MObjList*, a list that contains the M-Obj identifier (*MObjID*). $E_R$ has an additional parameter: the M-Hub Score for each M-Obj;
- *ALIVE*: message sent by the leader node at predefined time intervals thereby detecting if the leader node has failed or lost connection with objects. It is also used to inform other M-Hubs who are the new leaders. It has the following parameter: *ListOfLeaders*, a list containing the identifier of the M-Objs and their respective leaders;
- *PENDING*: message used to request information about the leader. It is also directed to a particular leader of

a given object or set of objects to request responses through *ALIVE* messages that may not have been received.

The algorithm seeks to select the two most suitable gateways for each object: a leader and a sub-leader. To be considered as the two most suitable, these gateways should be the nodes with the highest *Scores* in a finite set of M-Hubs and M-Objs in an environment (i.e., the neighborhood). The *Score* is determined from the RSSI and parameters related to computational resources of the mobile device. The usage of a sub-leader is an important strategy to keep a second node as a candidate to become a leader. In cases when the leader is no longer accessible, the sub-leader becomes the new leader, avoiding a new election process.

The proposed algorithm is divided into three parts: (i) initialization and discovery of objects, (ii) periodic functions, and (iii) election of leaders. The first part of the algorithm consists of initializing a set of variables to store information about the node and discovered smart objects. The object discovery service is provided by the M-Hub through the S2PA component. Every time an object is found, it is inserted into the *Object Container* and tagged as "leaderless". All objects contained in that container receive a timer, that is, a predefined value that allows to control the arrival time of *ALIVE* messages. If an expected message from the leader of a given object is not received at the preset time, it means that the object does not have a leader or some failure may have occurred.

The second part of the algorithm includes three functions that are performed periodically and handle the reception of messages. The *onAlive* function deals with the sending of *ALIVE* messages. Received *ALIVE* and *PENDING* messages are processed in the *receiveAliveAndPending* function. First, it is checked whether the received *ALIVE* message contains any smart object discovered by the node that received it. If so, it updates the information about the objects. Finally, it restarts the timer associated with the objects.

The *receiveElection* function handles the $E_S$ messages. A M-Hub, after receiving this message, calculates its *Score* for each object contained in the message that is within the *Object Container*, and inserts it into the *MObjList*. The $E_R$ message is sent back to the node which started the process. If the node does not know any object contained in the message, it is ignored. Leader *Score* for each object is calculated using Eq. 1.

$$Score = (Wr * Vr) + (Wb * Vb) + (Wc * Vc) \quad (1)$$

The notation of the elements is as follows: *Wr*, *Wb*, *Wc* are weights for the RSSI, battery level, and processing level (CPU), respectively. The weights are used to define different degrees of importance for each parameter.

The connectivity level of a node normally varies according to its position, since objects are geographically distributed in the environment. For this reason, an important feature the leader should have is easy access to disputed objects. By taking this characteristic into account, RSSI is used to determine the most accessible nodes. It is represented by a negative value given in dBm: the closer it is to zero, the more accessible it is to the node. RSSI value of the channels between M-Hubs and M-Objs may oscillate significantly over time. Therefore, it is necessary to calculate the average RSSI data over time. To obtain the RSSI value, Eq. 2 is used

$$MRSSI = \alpha * VRSSI + (1 - \alpha) * MRSSI \quad (2)$$

where *MRSSI* represents the average of RSSI values, the *VRSSI* value indicates the instantaneous RSSI value (i.e., the last RSSI value obtained), and alpha ($\alpha$) is a constant that represents a weight value (i.e., a weight that indicates the relative importance of the instantaneous value compared to the historical average). The default alpha value is 0.7, since this value allows the average value to be projected more gradually in relation to the last RSSI value obtained. Equation 3 is used to calculate the *Vr* value. The default *Wr* value is 5, the highest weight because we consider it as the most important characteristic to choose a leader, since it is desired that nodes with greater accessibility to the object are more likely to become a leader. Hence, *Vr* is a value that varies between 0 and 5.

$$Vr = -30 * Wr * |MRSSI| \quad (3)$$

where -30 is the RSSI value used as a reference. *Vb* and *Vc* are technical characteristics related to the node: *Vb* is the battery level, obtained from the device, and *Vc* is related to the CPU load value, that is, the amount of free processing of the device (measured in percentage values). The algorithm uses these variables to prioritize nodes more charged and with free processing since they can have a longer lifespan as a leader/sub-leader. Equation 4 shows how the *Vb* value is calculated.

$$Vb = \frac{Wb * LevelBattery}{100} \quad (4)$$

in which the *Wb* is 3. The *Vc* value is calculated in Eq. 5, in which *Wc* is equal to 2:

$$Vc = \frac{Wc * CPULoad}{100} \quad (5)$$

M-Hub *Score* value is calculated for each object found and ranges between 0 to 10. It should be noted that a set of nodes elects a leader based on its current state and collected information. A node is considered leader if: (1) it is connected to the object; and (2) it has the highest *Score*. *Object Container* manages each object by assigning its respective leaders (i.e., main leader and sub-leader). At

the end of the leader election process, the leaders of the objects may be changed.

The third part of the algorithm deals with the process of electing a leader and a sub-leader. Before starting the election process, a M-Hub checks whether: (1) there is a connection established with the wireless network so that it is possible to exchange messages; and (2) elections are in progress, if so, to reach consensus, it only starts the process when all elections in progress have been finalized and it identifies objects without leaders. The election process summarized in Algorithm 1 contains the basic implementation of the election mechanism. As input, the algorithm uses the identifier of the initiator M-Hub of the leader election process and the identifier of the M-Objs in dispute.

**Algorithm 1** Leader election process.

```
1  INPUT: M-HubID, M-ObjIDs;
2  OUTPUT: ListOfLeaders;
3  E_S ← (M-HubID, M-ObjIDs);
4  sendBroadcast(E_S) → Connected M-Hubs;
5  while TimerElection >= 0 do
6  |   TempList = Receive(E_R);
7  end
8  ListOfLeaders = getLeaders(TempList);
9  ALIVE ← ListOfLeaders;
10 sendBroadcast(ALIVE) → Connected M-Hubs;
```

The procedure checks all objects without an elected leader and broadcasts $E_S$ messages to all connected M-Hubs (line 4 in Algorithm 1). If a M-Hub does not know any object contained in the message, the message is ignored. The initiator M-Hub waits for a predefined time for other nodes to send their $E_R$ messages (lines 5-7 in Algorithm 1). Therefore, an $E_S$ message is used to start an election, and $E_R$ to reply an $E_S$ message. A M-Hub, after receiving an $E_S$ message, calculates its *Score* for each object contained in the message that is within its *Object Container*, inserts it into the $E_R$ message, and finally sends it back to the initiator M-Hub. A temporary variable (*TempList*) is created to receive $E_R$ messages from other M-Hubs.

The function *getLeaders* (line 8 in Algorithm 1) is used to obtain the leader/sub-leader for each M-Obj based on the *Score* of all M-Hubs that sent their $E_R$ message and also the *Score* of the initiator M-Hub (i.e., the initiator M-Hub calculates its *Score* at this time). Therefore, the *ListOfLeaders* is composed by leaders and sub-leaders for all M-Objs participating in the election process. Finally, the message *ALIVE* containing the *ListOfLeaders* (line 10 in Algorithm 1) is propagated to all M-Hubs to notify them about the new leaders and sub-leaders of a set of objects.

Periodically, each leader broadcasts *ALIVE* messages. If an expected *ALIVE* message from the leader of a given M-Obj is not received in a predefined timer (e.g., 1200 ms), it means that some failure might have occurred with that leader. When this happens, the sub-leader of that given M-Obj becomes the new leader. If the new leader also falls (i.e., it stops sending *ALIVE* messages), any M-Hub that knows that given M-Obj can start a new election process.

## 4 Testing the NAM-Hub for IoMT

NAM-Hub was implemented as a new version of the M-Hub for the Android platform. Messages are exchanged using the *JavaScript Object Notation* (JSON) format, implemented with the Google Gson library [2]. The objective of testing the NAM-Hub was to show its feasibility of practical use and illustrate the results regarding the message exchange.

### 4.1 Tests Planning and Design

Tests were conducted in the Laboratory of Intelligent Distributed Systems (LSDi) at Federal University of Maranhão, Brazil, as illustrated in Fig. 4. This laboratory environment is a representative indoor site for the tests by having different devices connected to a 802.11n network in infrastructure mode. Moreover, devices are mobile (i.e., smartphones and beacons carried by people) and static (i.e., a beacon fixed on the wall). To perform the experiment, we developed a test application with the NAM-Hub, which was used in three mobile devices (Samsung Galaxy S8 with Android 8.0) carried by three people: A, B and C in Fig. 4. In addition, three objects (BLE Beacons) were used: 2 ones carried by the Person D and Person E (i.e., M-Objs), and another one fixed on the right wall (see Fig. 4).

MAC object identifiers were manually entered into the *Object Container* as a way of isolating the system from objects that were not part of the experiment. Mobile devices running the NAM-Hub were isolated in a WiFi network to avoid interference of external factors. Beacons were connected to the M-Hubs through Bluetooth LE. Table 1 shows how the beacons were configured while conducting the experiment. The protocol used was the *iBeacon* [3]. The advertising interval sets the frequency in which the advertising packages are sent. Transmission power is associated with the signal strength of the BLE beacons.

The time parameter related to the sending of *ALIVE* messages has been set to 600 ms. The timer that controls the arrival time of *ALIVE* messages has been set to 1200 ms. That is, if an expected message is not received in this time interval, the leader is considered to have failed.

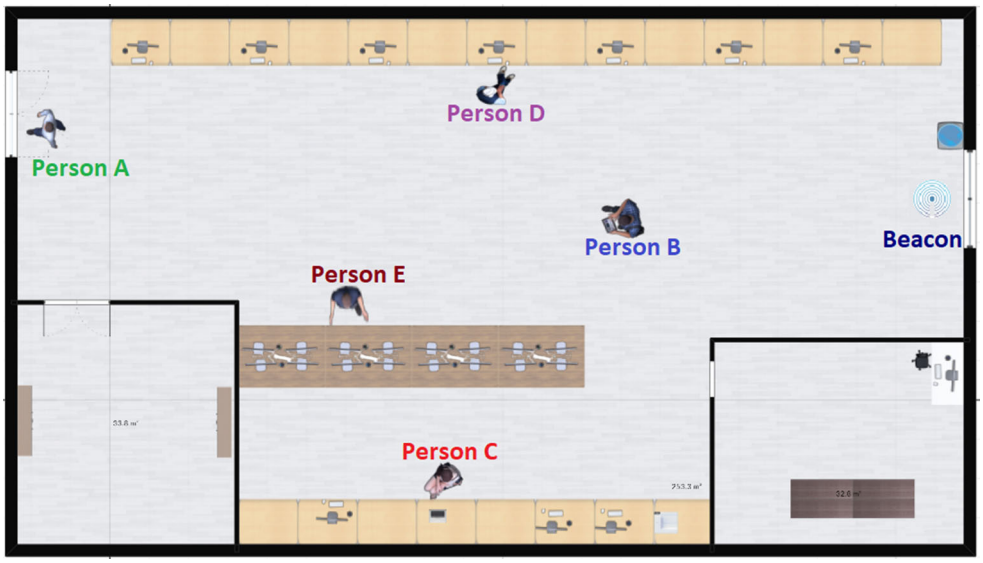


**Fig. 4** Testing environment

### 4.2 Conducting

We tested the NAM-Hub under two scenarios. Firstly, in Fig. 5, M-Hub carried by the Person A was the leader of the three objects. M-Hub with the Person B found the beacon with the Person E and broadcast a *PENDING* message (shown in Fig. 6), requiring information about its leader. M-Hub with the Person A, when receiving the message from M-Hub with the Person B, verified if it itself was the leader of the object listed in the message and, after confirming, it replied an *ALIVE* message (shown in Fig. 7). Hence, M-Hub with the Person B identified M-Hub carried by the Person A as the leader of the found M-Obj.

Secondly, in Fig. 8, the Person A leaves the laboratory, thus causing the device to disconnect from the wireless network, thereby M-Hub stops broadcasting *ALIVE* messages. Any M-Hub, in detecting the failure (i.e., the disconnection), could start an election process. In this experiment, M-Hub carried by the Person B detected and started the election by broadcasting $E_S$ message (shown in Fig. 9) and waited for two seconds. The M-Hub with the Person C, in receiving the message, verified if had a connection with the objects listed and calculated its *Score* for each one of them and sent back an $E_R$ message (shown in Fig. 10). In the end, M-Hub with the Person B selected the leaders and sub-leaders for the disputed M-Objs and broadcast this information.

**Table 1** Configuration parameters of the beacons.

| Configuration | Value |
|---|---|
| Transmission Protocol | iBeacon |
| Advertising Interval | 3 per second |
| Transmission Power | −3 dBm |

## 5 Experimental Evaluation

This section presents results from the performance evaluation performed in the same environment and configuration described in Section 4.1. Such as the second scenario of testing (Fig. 8), a failure was intentionally caused to the leader M-Hub. The version evaluated in experiments used CDDL middleware [25]. The experiment aimed at

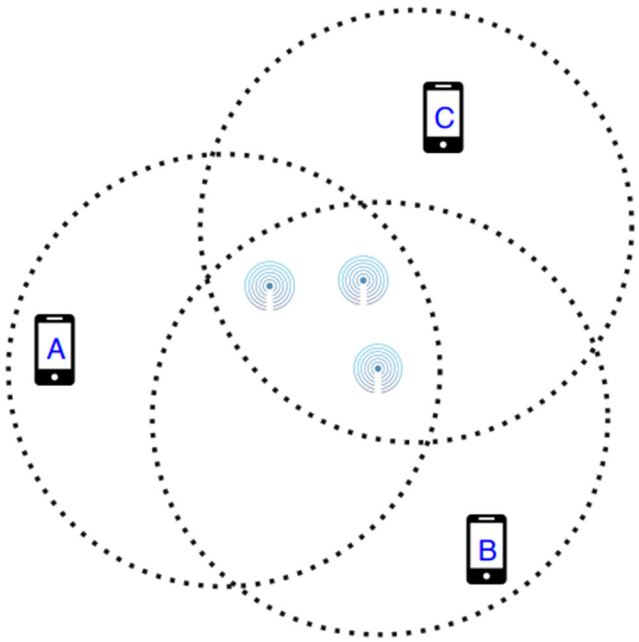


**Fig. 5** First test scenario

**Fig. 6** Example of *PENDING* message in JSON format

```
pending{"MHubID":"e5e6272b-403e-4daa-a461-bf72323199b0",
        "objectIDs":[{"MObjID":"0C:F3:EE:0E:34:9D"}]}
```

**Fig. 7** Example of *ALIVE* message in JSON format

```
alive{"MHubID":"babeaa18-6496-4bad-9b69-6fc373180547",
    "objectIDs":[{"MObjID":"0C:F3:EE:0E:34:9D",
                  "leaderID":"babeaa18-6496-4bad-9b69-6fc373180547'},
                 {"MObjID":"0C:F3:EE:0E:32:20",
                  "leaderID":"babeaa18-6496-4bad-9b69-6fc373180547"},
                 {"MObjID":"0C:F3:EE:0E:31:CE",
                  "leaderID":"babeaa18-6496-4bad-9b69-6fc373180547"}]}
```

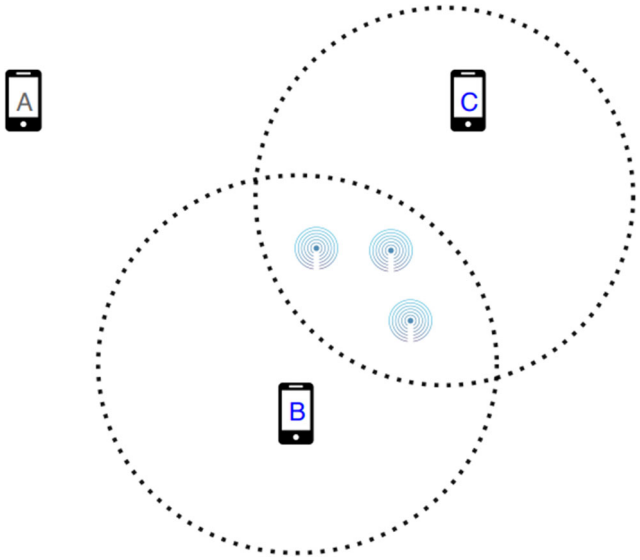


**Fig. 8** Second test scenario

**Fig. 9** Example of $E_S$ message in JSON format

```
election{ "MHubID":"e5e6272b-403e-4daa-a461-bf72323199b0",
          "objectIDs":[ { "MObjID":"0C:F3:EE:0E:34:9D"},
                        { "MObjID":"0C:F3:EE:0E:32:20"},
                        { "MObjID":"0C:F3:EE:0E:31:CE"}]}
```

**Fig. 10** Example of $E_R$ message in JSON format

```
election{ "MHubID":"80684afa-f4e6-4101-a082-ccd5de1a4c58",
          "objectIDs":[ {"MObjID":"0C:F3:EE:0E:34:9D","score":5.353914},
                        {"MObjID":"0C:F3:EE:0E:32:20","score":5.2899466},
                        {"MObjID":"0C:F3:EE:0E:31:CE","score":4.682632}]}
```

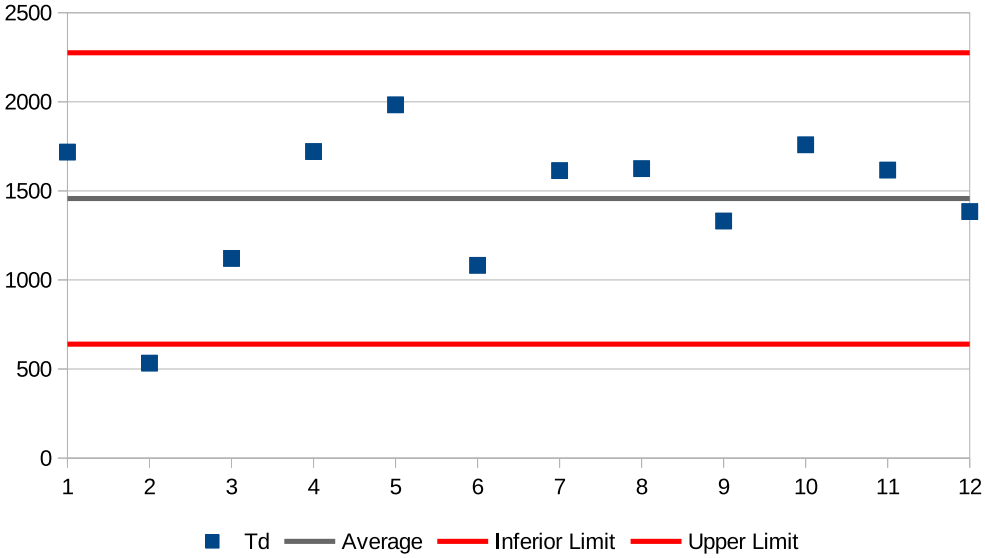


**Fig. 11** The leader crash detection time (TD)

assessing the performance of the proposed algorithm in case of crash-recovery.

### 5.1 Metrics

The analysis is made using the metrics *detection time* ($T_D$) proposed in [13] and recovery time ($T_{DR}$) proposed in [34] for the leader crash-recovery. That is, after causing the failure, we measured the time taken for each node (i.e., M-Hubs B and C) to detect the failure of the leader M-Hub A ($T_D$) and the elapsed time for each node to detect the presence of a new leader ($T_{DR}$), in which the elected leader could be the node itself.

Consider $t_c$ the timestamp of a leader crash and $t_d$ the timestamp of detection of that crash, the detection time ($T_D$) was obtained by calculating the difference of times: $T_D = t_d - t_c$. In a similar way, the recovery detection time ($T_{DR}$) was calculated considering the difference of recovery time $t_r$ and detection time of that recovery $t_{dr}$: $T_{DR} = t_{dr} - t_c$.

### 5.2 Results

The experiment consisted of 120 repetitions of a crash-recovery cycle, with intervals of 60 seconds between a complete recovery and a new crash. To obtain concise results, we excluded the highest and lowest values in each 10 repetitions [29].

Figure 11 shows the obtained $T_D$ results with each blue point representing the average value of 8 repetitions (discarded the highest and lowest values in each 10 repetitions). As we can see, the maximum value of 1983 ms and the minimum of 533 ms (AVERAGE = 1541.36 ms) are promising. Figure 12 displays $T_{DR}$ values with blue points representing the average value of 8 repetitions: maximum

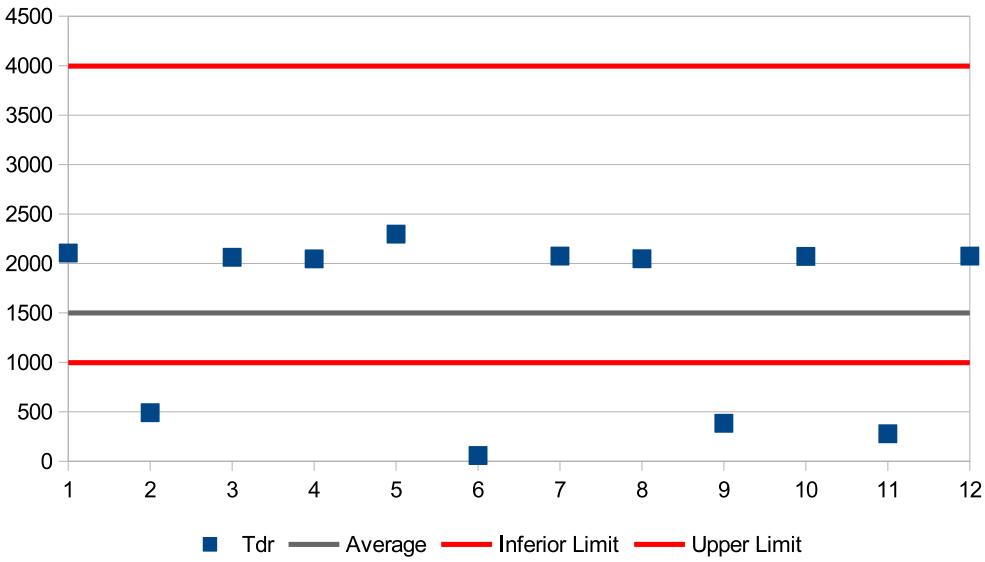


**Fig. 12** The the leader recovery detection time (TDR)

**Table 2** Summary of plot values

| | $T_D$ | $T_{DR}$ |
|---|---|---|
| Minimum | 533 | 58 |
| 1st Quartile | 1173.5 | 410.75 |
| Median | 1615.5 | 2055.5 |
| 3rd Quartile | 1720.25 | 2075 |
| Maximum | 1983 | 2297 |

time of 2297 ms and minimum value of 58 ms (AVERAGE = 1499.5), which are quick enough to avoid excessive data loss, thus proving the effectiveness of maintaining more than one node as a candidate to become a leader, thereby avoiding the need for a new election. Slight differences in mean values can be explained by interference in the wireless network signal and latency generated by the access point used in the tests. By considering that it is an opportunistic network scenario with mobile objects, we consider these values adequate to promptly react to failures and to keep the system stable. Values for descriptive statistic measures are described in Table 2.

## 6 Discussion

### 6.1 Analysis of the Proposed Solution

The number of election processes, given an arrangement with $n$ M-Hubs and $m$ M-Objs, up to $m$ elections may be performed concurrently and each M-Hub can start at most one election at a time. Thus, in the worst case, the number of elections generated is $m$, that is, an election for each M-Obj. The best execution case happens when an M-Hub starts a single election process for a set of $m$ objects, because the number of elections performed will be 1.

Regarding the number of messages exchanged, $n - 1$ messages are generated to propagate an election and disclose which M-Hub was elected. Thus, in the worst case scenario, the number of messages generated is $e(n - 1)$, where $e$ is the number of concurrent election processes. In the best case, the number of messages exchanged is $n - 1$. Therefore, the cost of the proposed algorithm is $O(n^2)$. It is important to emphasize that the leader election process in IoMT scenarios is characterized by being very dynamic and, for this reason, the cost regarding message exchange of the algorithm has a direct relation with the environment (i.e., the number of available devices) in a given moment.

### 6.2 Limitations

Our study has limitations that need to be discussed and, for this reason, we believe our work opens opportunities in this area for further research. Firstly, NAM-Hub cannot perform the exchange of leaders in cases where M-Hubs move very quickly, as there may be loss of connectivity, requiring a more quick execution of the election process. We also acknowledge that, although our solution considers dynamic device variables, a load balancing mechanism [30] for the NAM-Hub is required to avoid the excessive overhead in specific devices.

Secondly, this paper presents the NAM-Hub and an initial evaluation of interest to the scope of the study, however other experiments with NAM-Hub are required. Many experiments have been performed in the context of ContextNet [19, 37], SDDL [16] and CDDL [25] projects, and also with the first version of M-Hub [44]. For example, the latency in the discovery and connection processes between M-Hub and M-Obj (i.e., by measuring the connection time, service discover time, and enable notification time) is measured in [44], which does not differ from NAM-Hub. We have already evaluated the memory and battery consumption of the M-Hub with CDDL in [25], which is also the same for NAM-Hub in a static scenario without leader changes. However, since mobility will require re-elections, experiments could vary the mean time of mutual reachability among M-Hubs (i.e., a measure of stability), or the size of the sets of M-Hubs and M-Objs. Therefore, we acknowledge that additional experiments and metrics should be considered to identify the impact of the leader election mechanism to avoid wastage in the system and considering a greater number of nodes in the network.

Last, by considering security issues, both SDDL [26] and CDDL [25] have means to provide authentication and confidentiality in the exchange of messages. NAM-Hub inherits such resources when communicating with fog or cloud sides [37]. However, the leader election mechanism cannot deal with aspects related to information integrity during the leader election process. As the proposed algorithm uses a scoring system, on the one hand, a NAM-Hub could change values of its variables (i.e., memory, battery, and RSSI) to avoid becoming a leader/sub-leader, hence reducing resource consumption. On the other hand, a malicious NAM-Hub could alter these values to intentionally become a leader/sub-leader in order to access data produced by M-Objs.

## 7 Conclusion and Future Work

In this paper, we have presented an edge gateway provided with a leader election mechanism for IoMT. It is a solution for avoiding overloading a mobile edge gateway by monitoring several smart objects while there are other available ones. In addition, when using Bluetooth Classic, this solution avoids the wastage of communication,

processing and energy resources caused by the redundancy of mobile edge gateways in monitoring the same smart objects. NAM-Hub was proposed as a leader election mechanism to determine the most suitable edge gateways for each IoMT device discovered opportunistically. The proposed mechanism was conceived and implemented to select the two most suitable devices to become gateways: a leader and a sub-leader. Results showed its feasibility of practical use and reduced times to perform the whole process.

Future work initially involves considering security issues in the proposed mechanism. In addition, we would like to explore the use of CEP to process location and RSSI data of M-Hubs and M-Objs to determine their movement pattern, so enabling early detection of disconnections. Future research also includes the development of a load balancing mechanism for the NAM-Hub. We plan to develop it to be transparent and with minimal impact on system performance. Last, we would like to experiment our proposal in WiFi NANs [5, 11].

**Acknowledgements** The authors would like to thank FAPEMA (State of Maranhão Research Funding Agency) for supporting their research projects. This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9.

## Compliance with Ethical Standards

**Conflict of interests** The authors declare that they have no conflict of interest.

## References

1. Esper – EsperTech. http://www.espertech.com/esper/. Accessed 20 February 2020
2. Google Gson. https://github.com/google/gson. Accessed 20 February 2020
3. iBeacon – Apple Developer. https://developer.apple.com/ibeacon/. Accessed 20 February 2020
4. Mi Smart Band 4 – Mi Global Home. https://www.mi.com/global/mi-smart-band-4. Accessed 20 February 2020
5. Wi-Fi Aware – Wi-Fi Alliance. https://www.wi-fi.org/discover-wi-fi/wi-fi-aware. Accessed 25 February 2020
6. Zephyr. https://www.zephyranywhere.com/system/components. Accessed 20 February 2020
7. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutorial 17(4):2347–2376. https://doi.org/10.1109/COMST.2015.2444095
8. Atzori L, Iera A, Morabito G (2010) The internet of things: a survey. Comput Netw 54(15):2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010
9. Bounceur A, Bezoui M, Euler R, Kadjouh N, Lalem F (2017) Brogo: a new low energy consumption algorithm for leader election in wsns. In: 2017 10Th international conference on developments in esystems engineering (deSE), pp 218–223. https://doi.org/10.1109/DeSE.2017.11
10. Bounceur A, Bezoui M, Euler R, Lalem F, Lounis M (2017) A revised brogo algorithm for leader election in wireless sensor and iot networks. In: 2017 IEEE SENSORS, pp 1–3. https://doi.org/10.1109/ICSENS.2017.8234400
11. Camps-Mur D, Garcia-Villegas E, Lopez-Aguilera E, Loureiro P, Lambert P, Raissinia A (2015) Enabling always on service discovery: Wifi neighbor awareness networking. IEEE Wirel Commun 22(2):118–125. https://doi.org/10.1109/MWC.2015.7096294
12. Capra M, Peloso R, Masera G, Ruo Roch M, Martina M (2019) Edge computing: A survey on the hardware requirements in the internet of things world. Fut Internet 11(4). https://doi.org/10.3390/fi11040100
13. Chen W, Toueg S, Aguilera MK (2002) On the quality of service of failure detectors. IEEE Trans Comput 51(1):13–32. https://doi.org/10.1109/12.980014
14. Cugola G, Margara A (2012) Processing flows of information: From data stream to complex event processing. ACM Comput Surv 44(3):15:1–15:62. https://doi.org/10.1145/2187671.2187677
15. daCosta F (2013) Rethinking the Internet of Things: A Scalable Approach to Connecting Everything, 1st edn. Apress, Berkely
16. David L, Vasconcelos R, Alves L, André R, Endler M (2013) A dds-based middleware for scalable tracking, communication and collaboration of mobile nodes. Jo Internet Serv Appl 4(1). https://doi.org/10.1186/1869-0238-4-16
17. Dragan R, Ciobanu R, Dobre C (2017) Leader election in opportunistic networks. In: 2017 16Th international symposium on parallel and distributed computing (ISPDC), pp 157–164. https://doi.org/10.1109/ISPDC.2017.10
18. El-Refaay S, Azer MA, Abdelbaki N (2014) Cluster head election in wireless sensor networks. In: 10Th international conference on information assurance and security, pp 1–5. https://doi.org/10.1109/ISIAS.2014.7064625
19. Endler M, e Silva FS (2018) Past, present and future of the contextnet iomt middleware. Open J Internet Things (OJIOT) 4(1):7–23
20. Faika T, Kim T, Khan M (2018) An internet of things (iot)-based network for dispersed and decentralized wireless battery management systems. In: 2018 IEEE Transportation electrification conference and expo (ITEC), pp 1060–1064. https://doi.org/10.1109/ITEC.2018.8450161
21. Fernández-Campusano C, Larrea M, Cortinas R, Raynal M (2015) Eventual leader election despite crash-recovery and omission failures. In: 2015 IEEE 21St pacific rim international symposium on dependable computing (PRDC), pp 209–214. https://doi.org/10.1109/PRDC.2015.18
22. Ganti RK, Ye F, Lei H (2011) Mobile crowdsensing: current state and future challenges. IEEE Commun Mag 49(11):32–39. https://doi.org/10.1109/MCOM.2011.6069707
23. Garcia-Molina H (1982) Elections in a distributed computing system. IEEE Trans Comput 31(1):48–59. https://doi.org/10.1109/TC.1982.1675885
24. Gharehchopogh FS, Arjang H (2014) A survey and taxonomy of leader election algorithms in distributed systems. Ind J Sci Technol 7(6)
25. Gomes BDTP, Muniz LCM, Da Silva e Silva, FJ, Dos Santos DV, Lopes RF, Coutinho LR, Carvalho FO, Endler M (2017) A middleware with comprehensive quality of context support for the internet of things applications. Sensors 17(12). https://doi.org/10.3390/s17122853
26. Goncalves JF, Da Silva e Silva, FJ, Vasconcelos R, Baptista GLB, Endler M (2013) A security infrastructure for massive mobile data distribution. In: Proceedings of the 11th ACM international symposium on Mobility management and wireless access, pp 41–50. https://doi.org/10.1145/2508222.2508237

27. Goudos SK, Dallas PI, Chatziefthymiou S, Kyriazakos S (2017) A survey of iot key enabling and future technologies: 5g, mobile iot, sematic web and applications. Wirel Pers Commun 97(2):1645–1675. https://doi.org/10.1007/s11277-017-4647-8
28. Hassan N, Gillani S, Ahmed E, Yaqoob I, Imran M (2018) The role of edge computing in internet of things. IEEE Commun Mag 56(11):110–115. https://doi.org/10.1109/MCOM.2018.1700906
29. Jain R (1991) The art of computer systems performance analysis: Techniques for experimental design, Measurement, Simulation, and Modeling. Wiley, New York
30. Jiang Y (2016) A survey of task allocation and load balancing in distributed systems. IEEE Trans Parallel Distrib Syst 27(2):585–599. https://doi.org/10.1109/TPDS.2015.2407900
31. Kamilaris A, Pitsillides A (2016) Mobile phone computing and the internet of things: a survey. IEEE Internet Things J 3(6):885–898. https://doi.org/10.1109/JIOT.2016.2600569
32. Lane ND, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell AT (2010) A survey of mobile phone sensing. IEEE Commun Mag 48(9):140–150. https://doi.org/10.1109/MCOM.2010.5560598
33. Liu J, Shen H, Narman HS, Chung W, Lin Z (2018) A survey of mobile crowdsensing techniques: a critical component for the internet of things. ACM Trans Cyber-Phys Syst 2(3):1–26. https://doi.org/10.1145/3185504
34. Ma T, Hillston J, Anderson S (2010) On the quality of service of crash-recovery failure detectors. IEEE Trans Depend Sec Comput 7(3):271–283. https://doi.org/10.1109/TDSC.2009.35
35. Mao S, Zhao C, Zhou Z, Ye Y (2013) An improved fuzzy unequal clustering algorithm for wireless sensor network. Mob Netw Appl 18(2):206–214. https://doi.org/10.1007/s11036-012-0356-4
36. Masi AD (2015) Load balancing in p2p smartphone based distributed iot systems. Master's thesis, Luleå University of Technology
37. Meslin A, Rodriguez N, Endler M (2020) Scalable mobile sensing for smart cities: The musanet experience. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2020.2977298
38. Miorandi D, Sicari S, De Pellegrini F, Chlamtac I (2012) Internet of things: vision, applications and research challenges. Ad hoc Netw 10(7):1497–1516. https://doi.org/10.1016/j.adhoc.2012.02.016
39. Nahrstedt K, Li H, Nguyen P, Chang S, Vu L (2016) Internet of mobile things: Mobility-driven challenges, designs and implementations. In: 2016 IEEE First international conference on internet-of-things design and implementation (ioTDI), pp 25–36. https://doi.org/10.1109/IoTDI.2015.41
40. Salman O, Elhajj I, Chehab A, Kayssi A (2018) Iot survey: an sdn and fog computing perspective. Comput Netw 143:221–246. https://doi.org/10.1016/j.comnet.2018.07.020
41. Santana EFZ, Chaves AP, Gerosa MA, Kon F, Milojicic DS (2017) Software platforms for smart cities: concepts, requirements, challenges, and a unified reference architecture. ACM Comput Sureys 50(6):78:1–78:37. https://doi.org/10.1145/3124391
42. Sindhanaiselvan K, Mannan JM, Aruna SK (2019) Designing a dynamic topology (dht) for cluster head selection in mobile adhoc network. Mobile Networks and Applications. https://doi.org/10.1007/s11036-019-01283-x
43. Singh KJ, Kapoor DS (2017) Create your own internet of things: a survey of iot platforms. IEEE Consum Electron Mag 6(2):57–68. https://doi.org/10.1109/MCE.2016.2640718
44. Talavera LE, Endler M, Vasconcelos I, Vasconcelos R, Cunha M, Da Silva e Silva, FJ (2015) The mobile hub concept: Enabling applications for the internet of mobile things. In: 2015 IEEE International conference on pervasive computing and communication workshops (percom workshops), pp 123–128. https://doi.org/10.1109/PERCOMW.2015.7134005
45. Vasudevan S, Kurose J, Towsley D (2004) Design and analysis of a leader election algorithm for mobile ad hoc networks. In: Proceedings of the 12th IEEE International Conference on Network Protocols, pp 350–360. https://doi.org/10.1109/ICNP.2004.1348124
46. Véstias M. P., Duarte RP, de Sousa JT, Neto HC (2020) Moving deep learning to the edge Algorithms 13(5). https://doi.org/10.3390/a13050125
47. Zhang B, Liu G, Hu B (2010) The coordination of nodes in the internet of things. In: 2010 International conference on information, networking and automation (ICINA), vol 2, pp v2–299–v2–302. https://doi.org/10.1109/ICINA.2010.5636506
48. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J (2019) Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proc IEEE 107(8):1738–1762. https://doi.org/10.1109/JPROC.2019.2918951