Performance Improvement on Reception Confirmation Messages in Bluetooth Mesh Networks

Paola Pierleoni, Andrea Gentili, Marco Mercuri, Alberto Belli, Roberto Garello, *Senior Member, IEEE*, and Lorenzo Palma

***Abstract*—Bluetooth mesh is a recent technology built on the Bluetooth low energy protocol stack architecture, focusing on the Internet of Things. It represents an excellent solution for commercial and industrial lighting applications, though it is still evolving. One of the biggest challenges of the Bluetooth mesh network is the improvement of confirmation messages reception. In a Bluetooth mesh network, determining the Status of the received messages is a critical aspect that can generate unexpected issues when multiple devices respond simultaneously, as it may occur in some lighting applications. This behavior can reduce the probability of message delivery due to collisions, especially when the number of devices in the network increases. This article aims to improve the reliability in receiving confirmation messages in a Bluetooth mesh network by proposing a new technique of spreading Status overtime. To evaluate the proposed technique's performance, we compare our technique with a Bluetooth mesh network with standard configuration (SC) using real nodes experimental setup. We evaluated our results in terms of packet-loss rate, obtaining 98.84% of the Status received for the network with our optimized configuration and 96.98% for those with the SC. Finally, an in-depth performance evaluation method for the analysis of the lost Status was also conducted.**

***Index Terms*—Bluetooth mesh, Internet of Things (IoT), lighting application, packet-loss rate, status messages.**

I. Introduction

BLUETOOTH low energy (BLE) was introduced in the 4.0 version of the Bluetooth specification in 2010 [1], where a new technology and effective protocol that has achieved a strong impact on the market today was proposed. BLE uses the Bluetooth brand that mainly focuses on the Internet of Things (IoT) applications [2]–[5]. It requires a high reliability and scalability exchange of small amount of data between devices in sensor or beaconing applications [6], [7]. With each new version of the BLE technology, the achievable throughput, range, and energy efficiency have been improved significantly [8]. However, the limitation of the BLE design was that it is based on point-to-point interaction, thus the network size of the star-based topology, may be limited. Actually, it is a common trend in most recent applications to observe an important increase in the number of devices in networks with mesh topology. Moreover, BLE low-power competitors, such as ZigBee [9], Thread [10], or custom solution [11] offer mesh capabilities running on the IEEE 802.15.4 in the 2.4-GHz band. In these cases, meshing enables a many-to-many topology, where each node in the network can talk to every other node, directly or via multihop communication, using routing techniques [12]. Thus, the lack of standardized multihop models had prevented the BLE from occupying a key role in applications outside the short-range scenarios. Consequently, several attempts have been made to add meshing capabilities to BLE, resulting to different implementations of proprietary mesh networks [13], which makes interoperability between them impossible. Lack of standardization has been recently resolved when the Bluetooth SIG group released in July 2017, a mesh network specification built on top of the BLE protocol stack which makes use of some concepts and functionalities as its advertising capabilities [14].

The choice of one's specific technology, considering the heterogeneity of each in terms of protocols, performance, reliability, latency, cost and coverage, depends on the goals intended as both the services provided and application scenario are also factor to consider [15].

Bluetooth mesh with its simple structure, low implementation cost, absence of routing tables, and interoperability with the widely available BLE, is an attractive solution that provides unique features for a wide range of IoT applications [16]–[20]. Commercial and industrial lighting are great examples of deployment scenarios where Bluetooth mesh is a perfect solution. By including the features specifically for lighting systems and smart home network [21], Bluetooth mesh wireless network system has proven to be a flexible and low cost solution suitable for integrated lighting features. All mesh applications ensure interoperability across products from different vendors, and the behaviors are implemented using standard models [22], defined by the Bluetooth SIG to satisfy a product's distinct requirements [23]. In this scenario, wireless technology can support reliable, responsive, secure, and

Manuscript received April 20, 2021; accepted June 14, 2021. Date of publication June 18, 2021; date of current version January 24, 2022. This work was supported by the Project titled "SADABI-IT—Smart Awareness in Digital Automation and Business Intelligence with Integrated Tools," under Project B32C21000880005, call "Fabbrica intelligente, Agrifood e Scienze della vita." *(Corresponding author: Paola Pierleoni.)*

Paola Pierleoni, Andrea Gentili, Marco Mercuri, Alberto Belli, and Lorenzo Palma are with the Department of Information Engineering, Universitá Politecnica delle Marche, 60131 Ancona, Italy (e-mail: p.pierleoni@univpm.it; a.gentili@pm.univpm.it; m.mercuri@pm.univpm.it; a.belli@univpm.it; l.palma@univpm.it).

Roberto Garello is with the Department of Electronics and Telelcommunications, Politecnico di Torino, 10129 Turin, Italy (e-mail: roberto.garello@polito.it).

Digital Object Identifier 10.1109/JIOT.2021.3090656

scalable lighting systems, in which data transmission is urgent and its configuration changes in a very dynamical way.

To date, one of the biggest challenges comes from lighting applications, specially the lighting control for both commercial and industrial lighting. In particular, this is the case of a group of devices that receives simultaneously an On-Off light command. These devices reply with a confirmation message indicating that the command has been received. In principle, it is useful in several applications, because it ensures a good reception, and provides information on network operability. In fact, the collection and statistical analysis of the received messages furnish an estimation of the overall network reliability and the successful commands delivery over time. However, their usage, as reported in Bluetooth mesh specification [14], could generate unexpected issues when multiple nodes respond with this message simultaneously, thereby reducing the probability of message delivery due to message collisions, especially in some particular network topology configuration, and when the number of nodes in the network increases. This type of problem in Bluetooth technology has already been analyzed in the context of BLE advertising mode [24].

This article, aims to evaluate the reception probability of confirmation messages in a real Bluetooth mesh network and propose a procedure for packet-loss reduction for this type of messages. We present a performance comparison between the proposed procedure and that based on the standard configuration (SC). To provide a general assessment of the overall success networks performance in the command execution, we also propose an in-depth analysis of the results using only the information associated with the received confirmation messages.

In the mesh technologies' studies not many specific studies have been presented to evaluate the performance on a real application scenario, showing limitations and going beyond the general considerations currently provided from the literature on the mesh network behaviors. Most of them were concentrated on the performance analysis in terms of network delay and packet loss rate (PLR), at the varying of mesh configuration parameters and features, based on simulation environments.

Recently, a study has been carried out to perform an analysis of the simulated tests based on several sorts of flooding-based configuration parameters [25]. Some authors have identified configuration challenges, proposing network-tuning criteria and outlining possible standard improvement with an assessment within the implementation and execution in real devices with chipset limitations [26]. Pérez-Díaz-De-Cerio *et al.* [27] proposed a new feature to reduce the overall consumption in the Bluetooth mesh networks. Finally, other authors have evaluated the performance of the Bluetooth mesh technology, partially investigating its limitations for monitoring applications through real experiments [28]. However, this article presents choices at the experimental setup level leading to the creation of a network that is not completely compliant to the standard.

In this article we propose a detailed analysis, on the reliability of the reception confirmation messages in a Bluetooth mesh network, based on a standard firmware implementation on real devices. Then, we propose an optimized configuration (OC) achieving a significant packet loss reduction. We also carry out a general assessment of the network performance, improved with an offline analysis based on the results obtained in our experimental setup. The experimental setup consisted of tests carried out in two different scenarios. The first one was carried out in a controlled scenario with limited interfering elements. The second one was performed in a more challenging environment, as a typical real scenario. The remainder of this article is organized as follows. Section II briefly introduces some BLE mesh concepts and terminology. Section III describes the experimental setup used to evaluate the performance of a real Bluetooth mesh network. In Section IV, we describe the mesh test, whereas the analysis of the main results was carried out in Section V. Finally, Sections VI and VII, present the discussions and the conclusion, respectively.

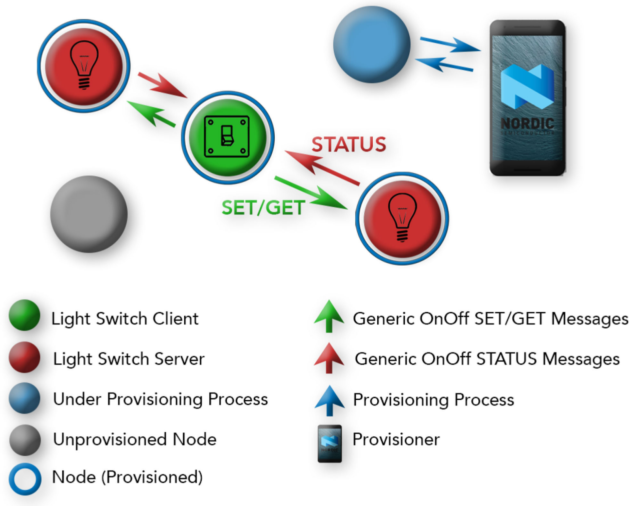

Fig. 1. Bluetooth mesh elements.

II. Bluetooth Mesh Background and Concepts

This section introduces some basic concepts and terminologies that characterize a Bluetooth mesh network [14]. In particular, we focus on some aspects that allow us to acquire a minimum background to understand this research article.

A. Bluetooth Mesh Background

First, a distinction must be made between devices that belong to the Bluetooth mesh network called Nodes from those with the necessary features to implement the standard but have not yet been associated with a Bluetooth mesh network, called *unprovisioned node*. An unprovisioned node becomes a Node through a particular process which is called *provisioning* Fig. 1. As will be seen in the following, this procedure is performed by specific nodes or external devices.

With the term Node, we generally indicate a device that is part of the mesh, but without going into the merits of its internal characteristics and functionality. Each one consists of at least one or more independent entity, called *elements*. An example is a light fixture (node) with different light bulbs (elements) that can be controlled separately. It is easy to understand that the elements need an indicator of their condition, e.g., on/off the light bulb, called *states* in the standard.

Every element is addressable through a *unicast* address to identify who sends or receives a message within the node. The standard defines two other types of addresses: 1) *group* and 2) *virtual*; both are multicast addresses, labeling one or more elements within one or more nodes. The difference between the two types lies in the number of addresses that can be assigned. In the groups, there are 256 fixed and 16128 dynamically assigned groups, whereas in virtuals addresses, many more nodes can be addressed using the 128-bit UUID Label than in the groups. The standard allows us to dynamically assign both group and virtual addresses to adapt the network to the physical structure in which it is installed.

The elements need to interact with each other to exchange information. This interaction is made possible due to the combination of the addresses described above and the introduction of a publish–subscribe paradigm. In this scheme, the sender (Publisher) merely publishes its message on the network without being aware of the identity of the recipient (Subscriber). The publisher can only infer the information if it is a single node or more, since the range of the address values for both cases is well defined. On the other hand, Subscribers, through the subscribing mechanism, can specify as precisely as possible which messages to subscribe. Even in this case, it is still difficult to know the nature of the Publishers or the number of nodes they publish on a group or virtual address.

The data structures used within the publish/subscribe paradigm to invoke operations between nodes are called *messages*. Messages are one of the key features of Bluetooth mesh, which is defined as message-oriented network technology. Inside the standard, we can identify three types of messages Fig. 1.

1) *Get* a message to request the Status from one or more nodes.
2) *Set* a message to change the state value of a node or a group of nodes. Set messages can be acknowledged or unacknowledged.
3) *Status* message used to respond to a Get request, in response to an acknowledged Set message, or is used by any element to indicate its Status independently; an example is a sensor that communicates periodically, its temperature value.

One important term defined in Bluetooth mesh specification is the *model* paradigm. A model brings with it all the previous concepts and is responsible for defining and implementing the characteristics of a given element [26]. Each element within a node must support one or more models Fig. 1 and can be grouped into three macrocategories.

1) *Server Model:* It defines the messages that the model can transmit and receive, the states in which an element may find itself and the behaviors (changes of state) that are triggered by the receipt of the appropriate messages.
2) *Client Model:* It defines the necessary messages to the node to request, modify or use corresponding states server. The client model does not have states.
3) *Control Model:* It has several functionalities, one or more of the models described above coexist, as well as a Control logic to coordinate the iterations between the models, for which it is composed.

To characterize each of the three macrocategories, Bluetooth SIG has provided a well-defined type of model: fundamental, generic, sensor, lightning, and vendor-specific Model; further subdivided according to their intended use. For the purposes of this article, we will give only a small hint about the Generic type. Any powered device has an on-off state, e.g., a fan, an air conditioning unit, a light, or a power socket, etc.; the standard to avoid characterizing each device with its on-off model, has generalized the concept by creating a Generic On/Off model with states and messages that can adapt to any situation. Within the Generic category there are also other models based on the same principle as level, transition, location, power, etc. [22] and the term Generic alongside these names indicates precisely the generality of use.

About this particular model, the Status messages, specifically the Generic On/Off Status, are the core of our study. Their structure and the data contained within them deserve special attention as they have allowed us to carry out all the necessary assessments and analyses. In all layered network architectures, including that of Bluetooth Mesh Standard, additional data is encapsulated and decapsulated at each layer. For the Generic On/Off Status, we can deduce the following packet field. For our purpose, given as follows.

1) *Present On/Off:* State in which the LED was before receiving the Set command.
2) *Target On/Off:* State in which the LED went after performing the Generic On/Off Set message.
3) *Remaining Time:* the time required from the moment in which the message is sent at the end of the transition to the Target state.

In a Bluetooth mesh network, all the nodes can receive and send messages: however, there are some particular features that a node can possess, which allow it to have special capabilities.

1) *Relay:* Messages are sent to the other nodes that are in direct radio range of the publishing node, but if the nodes to be reached are out of range, a "repeater" of messages is necessary to allow them to arrive at the destination. A relay node through advertising bearers can fulfill this role. In fact, it can receive and retransmit messages that are broadcasted by other nodes within the network [25]. This type of node is essential to extend network coverage allowing for multihop communication.
2) *Proxy:* The nodes of the network that support Proxy functionality, known as *Proxy Nodes*, are able to sent and receive messages between generic attribute profile (GATT) and Advertising Bearer. In this way any device with Bluetooth technology (above 4.0) that does not support advertising bearer, e.g., smartphones, tablets, can interface with the network and communicate with it.
3) *Low Power and Friend:* These two features allow the definition of two types of nodes, *low power node* (LPN) and *friend node* (FN), allowing to cope with all those situations in which a device, by necessity or choice, operates within the mesh network at significantly reduced receiver duty cycles. The relationship between LPN and FN is called Friendship [14].

4) *Provisioner:* This feature is the prerogative of all devices that, through the Provisioning process, can add a device to the mesh network. During *provisioning*, the *provisioner node* (PN) provides the unprovisioned node with the information and parameters necessary to interact with the entire network. The role of PN can be performed either by a device inside the Bluetooth mesh network, which then implements the entire protocol stack of the standard [29] or by an external device Bluetooth, e.g., smartphone, tablet. Through the use of the Proxy feature. In the latter case, the mobility and resources made available by a mobile device make the network configuration and its management much more versatile.

An example of the topology of the Bluetooth mesh network topology, in which all the types of nodes listed above are shown, is represented in [26].

B. Bluetooth Mesh Broadcast-Oriented Communication

The Bluetooth mesh works in a connection-less flooding mechanism, where each device broadcasts both its data and data that belongs to other devices. Though BLE operates both in advertising and in connection mode, BLE mesh devices use only the first one state, using three frequency channels 37, 38, and 39, as defined by Bluetooth Core Specification.

Thus, the nodes are not connected but communicate with advertising (ADV) packets through timing rules imposed by the BLE protocol [8], as will be explained below.

The same packet data unit (PDU) is transmitted sequentially over the three ADV channels, within a time slot defined as *Advertising Events*, and received from each device in radio range on the matching ADV channel during the corresponding *Scanning Event* that origins after a time slot defined as *scanInterval*. Moreover, the *scan window* parameter identifies the duration of each scan on one ADV channel. Note that, passive scanning should be set with a duty cycle as close to 100 percent as possible, to avoid missing any incoming mesh PDUs. Thus, the start of two consecutive *Advertising Events* is defined by the *AdvInterval* ($\geq$20 ms) plus the random delay (*advDelay*) added value between 0 to 10 ms. This is, according to BLE specification, and controlled by Link layer [1]. Furthermore, to lower the probability of packet collisions on all advertising channels, the standard recommends the gap between the start of two consecutive advertising PDUs within the same *Advertising Event* (T_{ChPDU}), which shall be less than 10 ms randomized (see Fig. 2).

C. Bluetooth Mesh Network Parameters

The efficiency and reliability of a mesh network depend on several features: the randomization on and within Advertising Events, discussed in the above section, and the control of redundant message repetitions, to ensure their reception, through the adjustment of several configuration parameters. However, the wrong choice of these features affects the possibility of increasing errors or collisions in the different phases of the communication flow. Bluetooth mesh network uses two main types of ADV packets: 1) network and 2) relay.

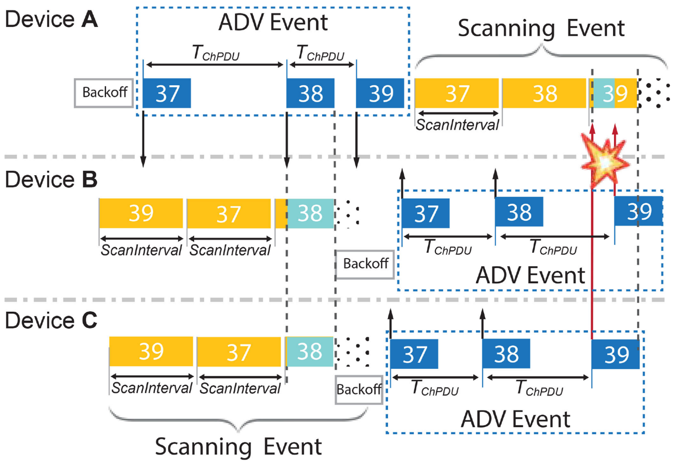

Fig. 2. Example of collision issue during communication flow between three devices in a Bluetooth mesh network.

Network transmit count and *Network transmit interval steps* (Nris) at network layer concern, respectively, the number of replicas of a PDU generated from a source node, and the space between these repetitions, with a transmission interval equal to (Nris + 1) $\times$ 10 ms. Like the previous *Network transmit*, the *Relay transmit count* and *Relay transmit interval steps* (Rris) are also parameters at the network layer, related to the relay features, when the message derives from another node and must be relayed. The transmission interval is equal to (Rris + 1) $\times$ 10 ms. As mentioned in the previous section, Link layer imposes a random delay value from 0 to 10 ms between each transmission, also for those managed by Nris and Rris. On the one hand, when activated, the relay functionality is particularly useful when the number of nodes that simultaneously retransmit the Status increases in the network, because it increases the possible routes for the PDU and its successful transmission through the mesh network, increasing the overall reliability performance. On the other hand, consecutive retransmissions could increase the collisions, hence generate network congestion. In this case, Status messages relayed, found in the middle, can be lost before arriving at their destination, or may not be correctly received, due to collisions or the unavailability of the receiving to scan on the advertising channels. This latter case is associated with the blind time [26], related to decoding or switching frequency functions, that may cause not continuous scanning, leading to a higher PDU loss rate than expected. In this regard, mesh specifications suggest to include a random time value, *backoff*, before the beginning of an *Advertising Event*, in particular, for all those networks PDUs received simultaneously by nodes with relaying functionality: it is necessary to apply a small random delay before relaying the network PDU to avoid collisions. More generally, a random time value is needed to be added before each first transmission to fulfill the random time requirement between each transmission and measurements performed with real devices seem to implement it, as declared in [26]. In any case, all consecutive advertiser transmissions should be affected by a random time.

D. Bluetooth Mesh Spreading Status Approach

Sometimes, the time interval between the different PDUs will not be sufficient in ensuring the correct reception of all messages. In particular, this occurs when messages are sent almost simultaneously, as shown in Fig. 2. The figure shows a simplified and general example of the case mentioned above. Devices B and C send two messages, which are received almost simultaneously by device A on channel 39, and the collision of the sent messages occurs. In this regard, Mesh specifications suggest that a further delay should be applied for a Status message initially sent in response to a received ACK message with a unicast address: the node should transmit the Status with a random delay (Rand_{ACK}) between 20 and 50 milliseconds. Moreover, if the Status is sent in response to a received message sent to a group address or a virtual address, the node should transmit the Status with a random delay (Rand_{ACK}) between 20 and 500 ms. This reduces the probability of the multiple nodes responding to this message at the same time and therefore increases the probability of message delivery rather than message collisions [14].

In this article, we analyze the reception Status issues in a Bluetooth mesh network. We adopt the Bluetooth mesh specification solution, implement it in real devices, remedy some shortcomings present in the firmware, and increase the probability of Status message delivery. Moreover, we try to provide a general assessment of network performances on the client side. Thus, we focus on the following.

1) The analysis of the Status messages packet success rate (PSR), in replying to the ACK messages with group addresses, trying to overcome some unexpected behaviors due to multiple packet collisions. More specifically, we experiment with the effect of using the Status overtime spreading technique, selecting an appropriate delay random value to spread in time Status messages, increasing the probability of message delivery, and reducing collisions over the different links.
2) The assessment of the network general performances, in terms of the overall success in executing the On/Off commands by the server nodes, from an analysis performed offline on the results, using only Status messages information client side.

III. Experimental Setup

In the proposed setup, Nordic nRF52 DK boards were adopted to implement BLE mesh network nodes, supporting Bluetooth Mesh Standard [30]. Starting from Nordic software development kit (SDK), we implemented into nRF52 DK boards ten Light Switch Server and one Light Switch Client. In the following, each type of node is briefly indicated with the term client and server. The server was board with a simple server role implementing one instance of Generic OnOff Server Model and identifying lamps to be switched on and off. In our setup, we assimilated the lamp to a LED, integrated into the nRF52 DK boards. The client was a board with a simple client role implementing one instance of a Generic OnOff Client Model and identifying a switch that drives all the lamps. In our setup, the client was connected to a PC in which the J-link real-time transfer (RTT) Logger software was able to take the data from the nRF52 DK board and store it in a log file. The J-link RTT Logger software, is part of the basic software package for J-Link. It took advantage of the RTT log technology developed by SEGGER Microcontroller, allowing two-way communication between the target board and PC. The technology is based on the J-Link's ability to read data into the microcontroller's memory while executing its code.

TABLE I
Configuration Parameters

Paramenter	**Value**
Number of devices	11
Servers $S\Delta$ (m)	0.40 ÷ 6
Client-servers $CS\Delta$ (m)	3 ÷ 16
Tx Power	0 dBm
PDU size	39 bytes
TTL	6
Network retransmit interval steps	5
Network retransmit count	3
Relay retransmit interval steps	5
Relay retransmit count	3
AdvInterval (ms)	20

To create the Bluetooth mesh network, a Provisioner must configure each node, allowing them to operate and exchange messages. In our case, both the server and the client we adopted were proxy nodes, and so we used a smartphone with the nRF Mesh Android App that allowed us to manage the provisioning process using its proxy features. Once provisioned, each node shows its capability, through the adoption of a set of configuration states, that describe its behavior within the mesh network. In addition to providing the essential information to associate the devices to the network,it was possible to assign publication and subscription addresses through the app. To obtain a one-to-many communication between client and server, a Group type publication address has been assigned to the Generic OnOff Client model. The same address was then set as the subscription address in the Generic OnOff model implemented in each server. In this way, with each Set message, all servers are affected by the same request simultaneously. As for the feedback messages directed from the servers to the client, even if we schematize the transmission with a many-to-one structure, the response of each server always takes place through the use of a Unicast address.

The configuration topology and the timings parameters chosen for the mesh network created in the proposed setup are summarized in Table I.

We have already discussed the choice of some parameters in the previous sections, regarding the network topology and the PDU size. We set the transmission power to 0 dBm, for all the nodes, more than enough for the type of test carried out and set the TTL equal to 6. Keeping in mind some fundamental considerations that emerged in [25], we adopted for both client and server *Network retransmit interval steps* = 3, and *Network retransmit count* = 5, to ensure the reception of the OnOff command, thus, increasing the PSR in general. Moreover, in this specific case, the maximum delay when using the replica retransmissions is an acceptable compromise for the improvement in reliability. Furthermore, the relay functionality is active on both client and server, with *Network retransmit interval steps* = 3, and *Network retransmit count* = 5, aware of the possible increase in lost packets and the congestion of the network, but necessary to compensate for the loss of the acknowledged Status messages sent back almost simultaneously by the servers. The relay functionality is particularly useful when the number of nodes that simultaneously retransmit the Status messages increases in the network, because it increases the possible routes for the PDU and its successful transmission through the mesh network, increasing the overall reliability performance. Note that the choice of the values of the parameters described so far are the result of decisions made on the preliminary tests carried out, to optimize the network performance.

Finally, it is crucial to consider the relationship between the duration of an *Advertising Event* and the *Scanning Event* to guarantee success in receiving messages thus, the matching of the channel through which the PDU is transmitted, and the equivalent one in the scan reception. This element is highly relevant because there are only three advertising channels.

In the proposed experimental setup, the behavior of the devices can be summarized as follows.

1) The client sends a *Generic OnOff Set* command with acknowledgment, e.g., requires that the LEDs of each server be turned on. Such a command can take only two values: a) 1 for the On and b) 0 for Off.
2) The servers receive the command, process it by changing the LED Status, and finally send their *Generic OnOff Status* message to the client. At the same time, keep in mind that the Relay functionality implemented in the network layer is active in each node of the server, they also take care of relaying the messages received from the other nodes and the same client.
3) The multiple Status messages reach the client.
4) J-link RTT Logger software takes the data from the client and stores it in the designated file.

The sequence of commands generated by the client consisted of an alternation of set On and set Off spaced by 10s and managed by a timer. The client's behavior allows to repeat cyclically what described from point 1 to 4 for the entire duration, approximately 15 h, of each trial in which a total of 5470 acknowledged message commands from the client to the 10 servers' nodes were sent. At the end of the sequence of commands, the generated log file was subjected to a careful review to find possible errors that can invalidate the trial. Comparing this trial to daily use, for example, inside a commercial premise, and assuming an average of six commands per day between On and Off, the simulated value of 5470 commands would correspond to about 911 days or equivalent to 2.5 years. During a trial the average current consumption measured on the client is about 5.52 mAh, whereas each server is about 2.08 mAh. The sequence of commands generated by the client and the related confirmation messages of the servers require a network energy consumption

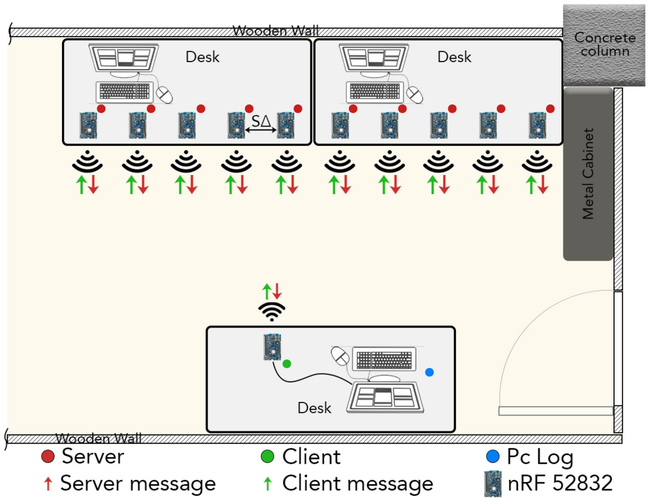

Fig. 3. Bluetooth mesh network implementation scheme of the controlled scenario.

of about 86.86 mWh. Such measurements of power consumption have been performed using Keysight N6705B DC Power Analyzer. In the proposed trial each confirmation message lost in the network requires command and responses retransmissions which involves an energy requirement of at least 0.24 mW. So even just a 1% reduction of the PSR of confirmation message leads to an increase in energy consumption of approximately 13.13 mW.

Given the importance of analyzing lost packets in the Bluetooth mesh network, we evaluate the performance based on results obtained in tests consisting of ten repetitions of the trial. In the proposed experimental setup, the trial was repeated ten times to have a statistically sufficient amount of data to analyze the results obtained. The tests were conducted in scenarios with two different environments and conditions in which the created Bluetooth mesh network operates. In particular, we conducted the tests in a controlled scenario and in a real scenario. The controlled scenario was arranged inside a laboratory with limited interfering elements, as shown in Fig. 3. Above a laboratory bench, the client and the PC were arranged while on another bench, at a distance between 3–3.6 m, the ten servers spaced about 40 cm apart from each other. For transmission power set and arrangement, all the nodes of the network are in complete visibility. Such a configuration is not the most favorable condition for exchanging messages within the mesh network. In fact, due to the relay functionality active on each node and complete visibility, the message traffic that the network must manage is high, generating collisions and packet losses. Note that it is not uncommon to find such a scenario in commercial environments, houses, industries, exhibition rooms, etc. However, this type of network represents one of the worst scenarios of smart lighting in terms of communication between devices. The client must manage multiple messages of feedback in reception, temporally spaced only by small delays, in most case insufficient.

The trials of the controlled scenario were carried out during the closing hours of the laboratory and in an environment

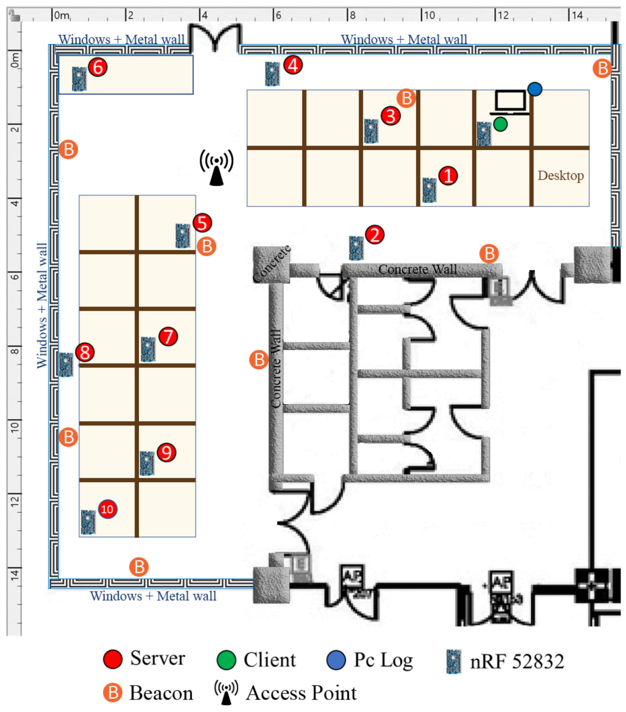

Fig. 4. Bluetooth mesh network implementation scheme of the real scenario.

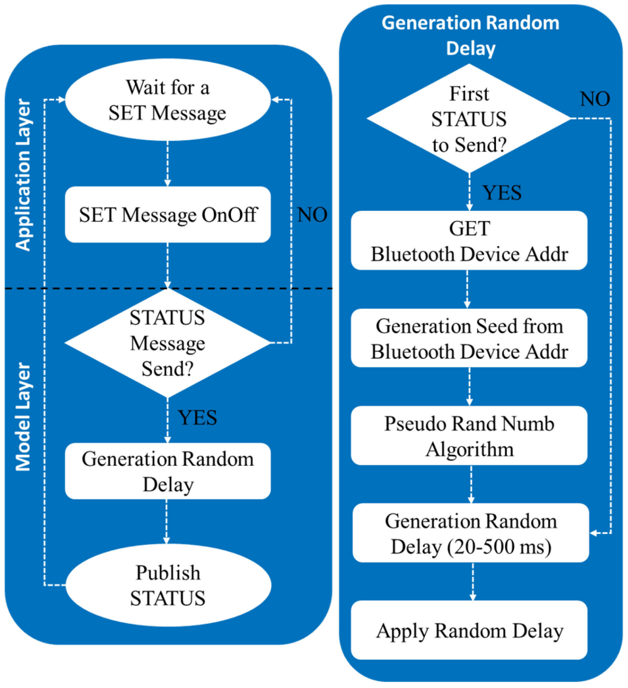

Fig. 5. Status randomization algorithm block diagram.

without obstacles and devices to minimize unpredictable interference to a minimum and ensure that a similar experimental condition for each test was implemented. This guarantees the reproducibility of the test and prevents the PLR measurements from being influenced by the co-channel interference due to obstacles and devices in the environment. The real scenario were arranged in a large office (Fig. 4) with two blocks made up of 12 desks inside. Each workstation is separated by wooden walls that rise about 70 cm above the desk top. The server nodes were arranged in such a way as to have a homogeneous distribution throughout the room and redundancy of paths for the exchange of messages. The distances between one node and the other vary between 2m and 6m. The trial of the real scenario was carried out close to office hours, considering a time span of 15 h. In contrast to the controlled situation, this scenario presents numerous interfering and attenuating sources for the signals transmitted by the Bluetooth mesh devices. In addition to the attenuation and reflections due to the structural and furnishing elements, there are also multiple RF disturbances and the presence of people occupying the environment. As shown in Fig. 4 we find a Wi-Fi access point placed on the ceiling in a central position, and a network of eight Bluetooth Beacons spread over the entire room. The latter transmit their information on the same advertising channels (Ch. 37, 38, 39) exploited by the Bluetooth mesh network. To these disturbances that we can consider fixed inside the room, we add all the random ones due to people and their personal devices, such as smartphones, tablets, PCs that access this place.

IV. Evaluation Tests Design

To evaluate the performance of reception confirmation messages in a Bluetooth mesh network, this study carried out two different tests in each scenario of the experimental setup illustrated in the previous section. The only difference between the tests consists of a substantial novelty in the management of Status messages, introduced at the server's model layer level. In particular, we conducted a test with SC of Bluetooth mesh network and another with an OC in terms of timing in sending Status messages. In the OC test, the change made at the server's model layer level mainly consists of introducing a delay between 20 and 500 ms before forwarding Status messages through the levels below. Otherwise, in the SC test, the servers do not implement any delay. All the other configuration parameters of the Bluetooth mesh network we chose are common to both types of tests, making it possible to directly compare the OC's performance with the SC one, starting from the same initial conditions for all the other setup variables. Therefore, the OC differs from the SC in the implementation of a technique of spreading Status overtime, but the setup and the general behavior of the entire network are identical, as described above in the Experimental Setup section.

The implemented technique consists in spreading the Status messages in time to increase the possibility of their reception, avoiding the issue of collisions. The OC's test was implemented in each server and the main steps involved, from the reception of a Set message to the reply with the publication of a Status one is shown in Fig. 5. First, for each On/Off command sent by the client each server receives, at the Application Layer, a new Set On/Off value derived from the Access Layer PDU transmitted by the underlying mesh layers. Then, through a function recalled at the Model layer, we checked if a Status message must be sent to the client, and published toward the underlying layers, after an imposed random delay time. Again, the process repeats for each new Set command received.

TABLE II
OVERALL PSR RESULTS FOR THE SC TEST OF THE CONTROLLED SCENARIO

ID Test	NODE										Avg. Single Test
	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	
SC_1	93.03	97.51	96.33	95.90	96.25	94.75	94.73	95.37	95.98	96.16	**95.60**
SC_2	96.07	97.88	96.63	97.68	97.07	96.73	96.42	96.36	96.45	98.17	**96.95**
SC_3	96.56	98.45	97.11	97.48	97.92	97.04	98.01	97.44	97.53	97.57	**97.51**
SC_4	95.70	97.90	97.22	97.48	97.61	96.65	97.61	96.89	96.78	97.09	**97.09**
SC_5	96.01	98.76	96.87	97.75	97.95	97.53	97.86	97.77	97.33	97.39	**97.52**
SC_6	95.78	98.68	96.91	95.72	96.86	96.98	96.91	96.18	97.39	96.20	**96.76**
SC_7	96.75	98.61	96.98	97.31	97.26	97.53	97.44	97.68	97.90	98.08	**97.55**
SC_8	97.35	98.57	97.62	97.15	96.76	97.57	97.57	96.97	97.71	98.03	**97.53**
SC_9	96.67	98.48	97.09	97.42	96.78	97.46	97.70	97.15	97.17	97.70	**97.36**
SC_10	94.46	97.77	96.12	96.07	96.80	95.01	95.85	95.72	95.61	95.72	**95.91**
Node Avg.	**95.84**	**98.26**	**96.89**	**97.00**	**97.13**	**96.73**	**97.01**	**96.75**	**96.99**	**97.21**	

In order to generate a random delay value for each Status sent, the first time a Status message is received, a pseudorandom number generator algorithm is used, based on the Bluetooth device address, chosen as seed. The Bluetooth device address is a unique 48-bit identifier assigned to each Bluetooth device by the manufacturer and is usually displayed as six bytes written in hexadecimal. Thus, after getting the Bluetooth device address, we generated a seed, represented as an integer, to use by the pseudorandom number generator algorithm. As a seed, we chose the first four digits of the device address. That is, we concatenated the digits, from right to left, from the least significant byte to the most significant byte, converting the whole four-bytes string generated in a decimal seed value.

Note that the steps described so far are performed only the first time each node receives a Status, as a sort of initialization. Thus, for each new Status to forward, every time a new random delay value is generated, between 20 and 500 ms, following the Bluetooth mesh specifications.

V. RESULTS

For both controlled scenario and real scenario, the network's performances were evaluated in terms of PSR Status (e.g., SC e OC test). From the analysis of the networks performance in executing the On/Off commands, a total of 54 700 commands sent by the client were considered for each test, respectively 5470 for each of the ten servers in the network. Hence, we computed the total of Status missed for each test, and the number of Status received. Then, we averaged the results obtained in each test for all the ten trials performed in each scenario. In order to assess how significant the difference between the two tests conducted is, student's t-test [31] among the averages along the 10 OC and SC tests were adopted, respectively.

Regarding the controlled scenario carried out in the laboratory, the results of the SC and OC tests shown in Tables II and III, respectively. Considering the overall results for each node (Table II) and then averaged for all ten tests,

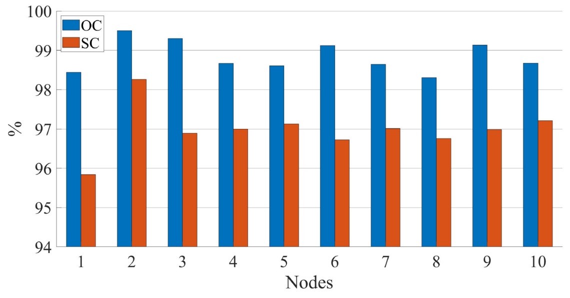

Fig. 6. Comparison of the Status PSR for each node, averaged for all ten tests of the controlled scenario, in both OC and SC cases.

we obtained, in the worst case of the SC test, an average of 95.84% Status messages received by the client from node one, whereas the best result was achieved by node two, with a 98.26% of Status messages correctly delivered. On the other hand, in the OC's test we obtained an average in the worst case of 98.44% for the Status messages received by the client from node one, and obtained the best result by node two, with 99.50% of Status messages properly delivered. Considering the overall results obtained for each test, on the average between all the nodes, the SC test shows, the worst result, with an average of 95.60% for the Status messages received by the client in test one, and the best result achieved in test seven, with 97.55% of Status messages correctly delivered. Whereas, for the OC test, as expected, in the worst situation, we obtained an average of 98.77% for the Status messages received by the client in test seven, with the best result achieved in test six, where 98.91% of Status messages was properly delivered.

In general, each node's performance was always higher for the OC test than its counterpart in the SC for all the ten tests' averages (see Figs. 6 and 7). Similar results can be obtained considering the average between all the nodes for each test.

As shown in Table IV, the resulting Status PSR in OC test was significantly different when compared to the SC one, with an average of 1.86% points of difference. As regards the

TABLE III
OVERALL PSR RESULTS FOR THE OC TEST OF THE CONTROLLED SCENARIO

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
OC_1	98.56	99.45	99.07	98.85	98.52	99.40	98.81	98.48	99.21	98.50	**98.88**
OC_2	98.32	99.47	98.96	98.56	98.48	99.20	98.78	98.39	99.16	98.57	**98.79**
OC_3	98.45	99.52	99.38	98.81	98.59	99.18	98.68	98.19	99.32	98.83	**98.90**
OC_4	98.63	99.31	99.36	98.63	98.70	99.10	98.48	98.03	99.12	98.78	**98.81**
OC_5	98.45	99.52	99.36	98.63	98.56	99.07	98.59	98.12	99.31	98.76	**98.84**
OC_6	98.23	99.63	99.52	98.76	98.78	98.98	98.63	98.46	99.29	98.79	**98.91**
OC_7	98.28	99.65	99.34	98.39	98.81	99.16	98.37	98.19	98.88	98.61	**98.77**
OC_8	98.56	99.65	99.51	98.72	98.68	98.90	98.74	98.34	98.94	98.52	**98.86**
OC_9	98.46	99.31	99.29	98.68	98.50	99.03	98.67	98.37	98.87	98.79	**98.80**
OC_10	98.50	99.49	99.21	98.63	98.45	99.21	98.72	98.52	99.25	98.57	**98.86**
Node Avg.	**98.44**	**99.50**	**99.30**	**98.67**	**98.61**	**99.12**	**98.65**	**98.31**	**99.14**	**98.67**	

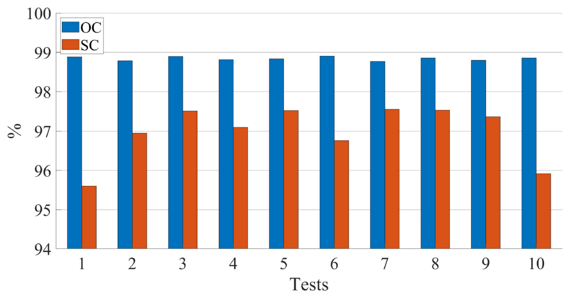

Fig. 7. Comparison on Status PSR for each test of the controlled scenario, averaged for all 10 nodes, in both OC and SC cases.

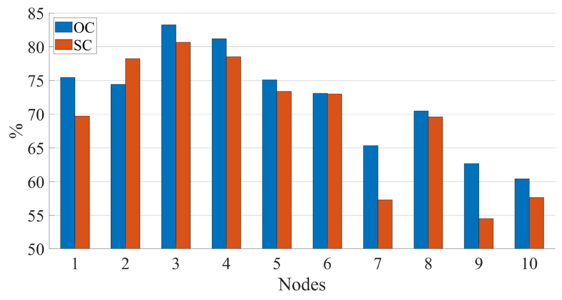

Fig. 8. Comparison of the Status PSR for each node of the real scenario, averaged for all ten tests, in both OC and SC cases.

TABLE IV
COMPARISON BETWEEN OC AND SC TESTS OF THE CONTROLLED SCENARIO

	Overall Average (%)	**Variance (%)**
SC	96.98	0.445
OC	98.84	0.002

OC, 98.84% of the Status was received, whereas 96.98% was received for the SC. Furthermore, the related average variance computed on both SC and OC tests was particularly significant, respectively, on 0.44% for the SC and 0.002% for the OC, as presented in Table IV. It results, as regards both the variance between the means on the individual nodes and the individual tests carried out.

As regards the real scenario carried out in the office, the results of SC and OC tests are shown in Tables V and VI. In the SC's test we obtained, considering the overall results for each node averaging all ten tests, the worst and the best result case, respectively, by node nine with 54.52% and by node three with 80.67% of Status messages correctly delivered, where, in the OC's test, the worst and the best results were achieved, respectively, by node ten and three with 60.41% and 83.25% of Status messages correctly delivered. Considering the overall results obtained for the SC test, on the average between all the nodes, we obtained the worst and best results, respectively, for test six with 68.36% and test four with 70.47% of Status messages correctly delivered. On the other hand, in the OC test, test eight and test seven achieved worst and better performances, respectively, with 71.19% and 73.65% of Status messages correctly delivered.

In the real scenario, each node's performance was almost always higher for the OC test than its counterpart in the SC (see Fig. 8), although only node two in the OC test performed worst than node two in the test SC. On the other hand, as the controlled scenario, each test's performance was always higher for the OC test than its counterpart in the SC's for all the ten tests' averages (Fig. 9).

As shown in Table VII, the resulting Status PSR in OC's test was significantly differ when compared to the SC one, with an average of 2.88% points of difference. As regards the OC, 72.14% of the Status was received, whereas 69.26% was received for the SC. Furthermore, the related average variance computed on both SC and OC tests was 0.62% for the OC and 0.46% for the SC, respectively, as shown in Table VII.

In conclusion, for the controlled scenario and the real scenario, respectively, about 15-h tests were carried out and t-test

TABLE V
OVERALL PSR RESULTS FOR THE SC TEST OF THE REAL SCENARIO

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
SC_1	67.41	77.59	80.65	76.94	72.96	72.41	58.52	69.54	54.17	55.65	**68.58**
SC_2	69.00	79.23	80.95	79.53	73,86	70.62	56.94	69.60	51.57	55.83	**68.71**
SC_3	68,98	77.04	79.54	78.33	73.70	70.37	55.93	67.31	60.74	62.78	**69.47**
SC_4	64.17	77.31	80.56	78.61	77.04	70.00	59.07	70.19	63.33	64.44	**70.47**
SC_5	70.37	77.50	78.61	78.15	69.17	74.26	57.69	70.56	52.69	56.11	**68.51**
SC_6	70.37	78.52	81.30	77.22	69.91	74.17	56.48	70.46	50.28	54.91	**68.36**
SC_7	70.19	81.67	83.52	79.35	72.69	74.35	58.80	6815	54.17	59.89	**69.68**
SC_8	74.17	77.50	80.37	80.19	72.69	73.70	57.69	73.15	52.59	56.20	**69.82**
SC_9	71.02	77.59	81.39	77.96	76.48	76.11	57.31	70.93	52.22	58.06	**69.91**
SC_10	71.51	78.45	79.83	78.91	75.39	73.91	54.58	66.05	53.47	58.65	**69.07**
Node Avg.	**69.72**	**78.24**	**80.67**	**78.52**	**73.39**	**72.99**	**57.30**	**69.59**	**54.52**	**57.65**	

TABLE VI
OVERALL PSR RESULTS FOR THE OC TEST OF THE REAL SCENARIO

	NODE										
ID Test	**n1**	**n2**	**n3**	**n4**	**n5**	**n6**	**n7**	**n8**	**n9**	**n10**	**Avg. Single Test**
OC_1	74.17	83.15	79.07	81.76	77.04	74.00	69.35	70.74	56.57	62.78	**72.86**
OC_2	74.26	81.57	78.89	80.19	76.48	72.04	65.37	66.67	56.30	63.15	**71.49**
OC_3	71.57	79.63	82.78	77.31	75.37	75.46	58.33	72.69	63.24	58.15	**71.45**
OC_4	73.70	77.22	83.61	81.39	76.20	74.17	60.74	68.98	64.17	58.98	**71.92**
OC_5	77.96	70.65	82.96	81.20	75.00	71.02	67.31	72.41	61.85	60.09	**72.05**
OC_6	77.69	72.41	84.35	81.67	75.09	70.56	70.09	73.70	63.24	60.37	**72.92**
OC_7	77.22	73.43	84.72	81.20	77.13	73.70	70.00	74.26	65.37	59.44	**73.65**
OC_8	73.80	67.96	84.44	82.31	72.78	73.33	64.07	68.43	64.54	60.28	**71.19**
OC_9	75.83	67.41	84.07	80.56	72.78	74.35	63.33	67.96	64.91	61.30	**71.25**
OC_10	78.33	70.74	87.50	83.98	73.15	72.50	64.81	68.89	66.67	59.54	**72.61**
Node Avg.	**75.45**	**74.42**	**83.25**	**81.16**	**75.11**	**73.11**	**65.34**	**70.47**	**62.69**	**60.41**	

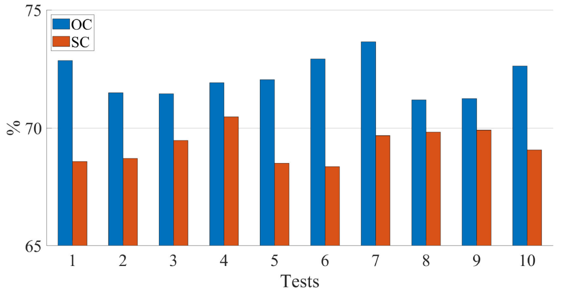

Fig. 9. Comparison of the Status PSR for each test of the real scenario, averaged for all ten nodes, in both OC and SC cases.

TABLE VII
COMPARISON BETWEEN OC AND SC TESTS OF THE REAL SCENARIO

	Overall Average (%)	**Variance (%)**
SC	69.26	0.46
OC	72.14	0.62

was computed for independent samples to determine whether there were statistically significant differences between the averages observed for the SC and OC. For both controlled and real scenario, preliminary analyses did not show the presence of outliers either for SC or for OC and the data made up of the averages on individual tests (Avg Single Test) were found to be distributed according to a normal one in both tests, as verified by the Shapiro–Wilk test ($p > 0.05$). For the controlled scenario, null hypothesis (difference in group means is zero) was assumed. That is, assuming that the averages of the two tests are the same, it turned out that the difference between the averages observed for the SC and OC test was $p = 1.48$E-07 ($p < 0.5$) therefore statistically significant, with *standard*

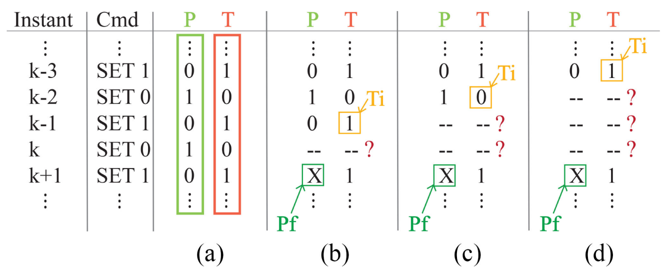

Fig. 10. Status lost analysis. (a) P/T status sequence in absence of anomalies for the selected SET series. (b) Anomaly with loss of isolated status messages. (c) Anomaly with loss of two consecutive status. (d) Anomaly with loss of three consecutive status.

deviation $\sigma = 0.59$ and $\sigma = 0.40$, respectively, whereas, for the real scenario, it turned out that the difference between the averages observed for the SC and OC test was $p = 1.46\text{E-}07$ ($p < 0.5$), therefore statistically significant, with *standard deviation* $\sigma = 0.71$ and $\sigma = 0.83$, respectively.

VI. Discussions

To provide an in-depth analysis of the results obtained in the performed experiments we also presented a method for evaluating the overall network's performance, processing the log files obtained from each trial of the tests and analyzing the lost Status from each server. This type of analysis is specific for our purposes and all those using binary values for the Present (P) and Target (T) fields in their models. Using models with Present and Target with more than two values requires a different analysis. The log files obtained from each trial of the SC and OC tests were processed in a MATLAB environment to evaluate implemented networks' overall performance. In particular, we implemented a simple code to count the total amount of Status received from each server node, out of which we performed a total of 5470 commands sent for each test. Moreover, we introduced the capability to identify the Status not received by the client, called anomalies, differentiating the cases in which these are lost only once, called isolated, and those who are lost more than once, called consecutive. In fact, the anomalies can be produced from the analysis carried out on Present On/Off (P) and Target On/Off (T) packet fields after a lost Status, using only the above information on the client side. The general idea is to observe the Status received before and after the anomaly to recover what happened inside it with a certain degree of uncertainty. The term uncertainty indicates the inability to univocally determine what happens within a sequence of n-Status lost consecutively and always referred to the same node. To give a correct contextualization of the concept of uncertainty and explain more clearly the logic used to achieve the proposed results, some examples of anomalies that can occur in a test are shown in Fig. 10.

As shown in Fig. 10(a), the reception of one Status ensures the delivery of the On/Off command for each server's single element, checking the value of the P and T packet fields in the next instant in which the Status was received. Therefore, we can formulate the following condition that should always occur within a *P/T* sequence of a k-Status message received

$$\text{Set}_k = P_{k+1} = T_k. \tag{1}$$

For the deterministic characteristic of the Set sequence and consequently of the *P/T* sequence, it is possible to uniquely determine what happened in instant k analyzing the *Set* value, the P field at the instant $k + 1$ *Final Present* (*Pf*), and the T field at the instant before the anomaly *Initial Target* (*Ti*).

The first case is the simplest one, representing an isolated Status in a certain instant k. As shown in Fig. 10(b), if $Pf \neq Ti$ and $Pf = \text{Set}_k$ then we can conclude that at the instant K, even if the client did not receive the Status, the Set_k command was received and executed by the server. Conversely, if $Pf = Ti$, it means that $Pf \neq \text{Set}_k \neq Tk$, then it can be said with certainty that at the instant k the command Set = 0 was not executed. Even if Status' loss concerns a Set_{k-1} command, therefore equal to 1, the reasoning remains the same. We can conclude that uncertainty is zero in this case. In fact, we can always recover the information associated with the lost Status.

The second case concerns the loss of two consecutive Status [Fig. 10(c)]. As in the previous case $Pf = x$ where $x = 0$ or 1. If $x = 1$, then, $Pf \neq Ti$, which leads to having $Pf \neq \text{Set}_k$. So we can say with certainty that the Set_k command has not been executed. With equal certainty, we can also say that Set_{k-1} has been performed instead of Set_k; in fact, the only hypothesis for both initial conditions to occur is that $Pf = \text{Set}_{k-1}$. Moreover, the uncertainty is zero.

If $x = 0$, then $Pf = Ti$. We cannot say univocally that all the commands have been executed or the Set_{k-1} has not, whereas the Set_k has, or no commands have been executed during the lost Status. Any conclusion is valid and there are no considerations that allow us to exclude one case rather than another. In this situation the uncertainty is maximum and equal to the number of Status lost.

Even for the case in which there is a loss of three consecutive Status as in Fig. 10(d), we can distinguish two instances and make the appropriate considerations. If $Pf \neq Ti$ then, it means that something between the Status at the instant $k - 3$ and $k + 1$ happened, surely one of the Set = 0 has been performed, but without further information, we cannot make considerations to reduce uncertainty further. In this instance we will conclude that the uncertainty is equal to $n - 1$, where n is equal to the number of consecutive lost Status. For the second instance, if $Pf = Ti \neq \text{Set}_k$, we can say with certainty that the server did not perform Set_k; otherwise, we would have the following equality *P/T*, $P_{k+1} = T_k = \text{Set}_k$.

The contents of Fig. 10 are only some possible combinations by way of example. There may be consecutive Status losses greater than three, although the chances of this happening are very small. To generalize the uncertainty $I(n)$ assessment for any anomaly present within the P/T sequence, we have formulated the following conditions (2):

$$I(n) = \begin{cases} n - 1, & \text{if } n \text{ is odd} \\ n - 2|Pf - Ti|, & \text{if } n \text{ is even} \end{cases} \tag{2}$$

remembering that n indicates the number of *Generic OnOff Status* messages lost consecutively, *Ti* (Initial Target) the target value of the last Status received before the sequence of Status lost consecutively and *Pf* (Final Present) indicates the present value of the first Status received after the sequence of Status lost consecutively. The uncertainty $I(n)$ refers to the

TABLE VIII
COMPARISON BETWEEN OC AND SC TESTS OF THE HE CONTROLLED SCENARIO, IN TERMS OF AVERAGE AND VARIANCE, FOR STATUS MISSED (CONSECUTIVE AND ISOLATED) AND STATUS RECEIVED

Test		Status Missed			Status Received
		Consecutive	Isolated	Consecutive Respect to the Isolated Ones	
OC	**Average (%)**	0.01	1.14	**1.29**	98.84
SC		0.25	2.77	**7.80**	96.98
OC	**Variance (%)**	2.93E-04	1.02E-03	**2.07**	1.29E-03
SC		0.03	0.48	**13.32**	0.73

TABLE IX
COMPARISON OF OC AND SC TESTS OF THE CONTROLLED SCENARIO, IN TERMS OF THE AVERAGE, IN COMPUTING THE MINIMUM ON/OFF COMMANDS EXECUTION, AND UNCERTAINTY

Test		Status Missed						Status Received	Minimum On/Off Commands Executed	Uncertainty
		Consecutive				Isolated				
		Even (n = 2)		Odd (n = 3)						
		I(n) = n	*I(n) = n-2*	*I(n) = n-1*		*I(n) = 0*				
		Pf=Ti	*Pf≠Ti*	*Pf=Ti*	*Pf≠Ti*	*Pf≠Ti*	*Pf=Ti*			
OC	**Average (%)**	0.01	0	0	1.05E-03	1.14	5.48E-04	98.84	99.99	0.01
SC		0.23	0	0	2.40E-02	2.77	0	96.98	99.75	0.25

calculation of the uncertainty for each anomaly. To calculate of the overall uncertainty within an entire test, it will be necessary to add up all the individual uncertainties and relate them to the overall number of Set sent.

Regarding the controlled scenario carried out in the laboratory, the results of the aforementioned performance evaluation method on the comparison between SC and OC, in terms of average and variance, are shown in Table VIII. Regarding the OC, 98.84% of the Status were received, hence the On/Off commands were executed with certainty. Moreover, more than 1.14% resulted isolated, whereas more than 0.01% resulted consecutive for the Status missed. However, we obtained 96.98% of the Status received for the SC, which means the On/Off commands were executed with certainty. Again, for Status missed, more than 2.77% resulted isolated, whereas more than 0.25% resulted consecutive. In particular, the percentage of consecutive Status compared to the isolated was 7.80% in the case of SC and 1.29% for the OC. As explained in the section above, for isolated Status we were always able to recover the information associated with them, therefore, whether the command was executed or not ($I(n) = 0$). In this case, as shown in Table IX, in our analysis, we are able to determine with certainty that most of the commands were executed even in the absence of reception of the Status, client side, when $Pf \neq Ti$. Only very few commands were not really executed, when $Pf = Ti$. Unlike the analysis of Status missed consecutively in the isolated Status there was no contribution to the increase or decrease in global uncertainty. However, it is possible for consecutive Status, to reach the same unambiguous conclusion as for the isolated ones. Nevertheless, more in-depth analysis is necessary for trying to reduce, where possible, the uncertainty associated with this kind of anomalies. That is, trying to recover any information inside the anomaly, for which it can certainly be concluded that the commands, related to one or more consecutive Status lost, have been received or not. The scope of what has been described above is to further increase the accuracy on acknowledging the total commands correctly executed, and the component relating to the totality of the commands for which we can say that the command has been received or not. In particular, as shown in Table IX, we experienced *even* anomalies with $n = 2$ and *odd* with $n = 3$, where n is equal to the number of consecutive Status lost. Consequently, and consistently with considerations made in the previous section, we found all the consecutive anomalies occurred for $n = 2$ with maximum uncertainty equal to n, when $Pf = Ti$. Only in a few cases were the uncertainty equal to $n-1$, for $n = 3$, and $Pf \neq Ti$. Only for the latter case, as explained in the section above, the reduction of uncertainty considered as contributing to the increase in the of the commands which we can certainly say has executed. In general, we obtained 99.99% of the commands certainly executed, with an uncertainty for the remaining commands of 0.01% in our OC analysis. Whereas, in SC analysis we achieved 99.75% of the commands certainly executed, with uncertainty of 0.25%.

Considering the real scenario, we conducted the same performance evaluation method on the comparison between SC and OC test as for the aforementioned scenario. As we

TABLE X
COMPARISON BETWEEN OC AND SC TESTS OF THE HE REAL SCENARIO, IN TERMS OF AVERAGE AND VARIANCE, FOR STATUS MISSED (CONSECUTIVE AND ISOLATED) AND STATUS RECEIVED

Test		**Status Missed**			**Status Received**
		Consecutive	**Isolated**	**Consecutive Respect to the Isolated Ones**	
OC	**Average (%)**	13.98	13.88	**100.72**	72.14
SC		16.10	14.64	**109.97**	69.26
OC	**Variance (%)**	0.62	0.13	**482.86**	0.62
SC		0.20	0.44	**45.03**	0.46

TABLE XI
COMPARISON OF OC AND SC TESTS OF THE REAL SCENARIO, IN TERMS OF THE AVERAGE, IN COMPUTING THE MINIMUM ON/OFF COMMANDS EXECUTION, AND UNCERTAINTY

Test		**Status Missed**						**Status Received**	**Minimum On/Off Commands Executed**	**Uncertainty**
		Consecutive				**Isolated**				
		Even		**Odd**						
		I(n) = n	***I(n) = n-2***	***I(n) = n-1***		***I(n) = 0***				
		Pf=Ti	***Pf≠Ti***	***Pf=Ti***	***Pf≠Ti***	***Pf≠Ti***	***Pf=Ti***			
OC	**Average (%)**	9.21	0.06	0.04	4.68	13.78	0.10	72.14	87.43	12.47
SC		10.69	0.06	0.04	5.31	14.53	0.11	69.26	85.27	14.44

would have expected and shown in Table X, for both the OC and the SC test we obtained a considerably lower Status correctly delivered, 72.14% and 69.26%, respectively. That means commands executed with certainty. Moreover, regard the Status missed, we experimented for the OC 100.72% of consecutive Status compared to the isolated (13.98% consecutive and 13.88% isolated) and for the SC 109 97% (16.10% consecutive and 14.64% isolated).

In particular, as shown in Table XI, we obtained 87.43% of the commands certainly executed, with an uncertainty for the remaining commands of 12.47% in our OC analysis. The remaining 0.10% represent the Status missed isolated certainly not executed. Whereas, in SC analysis we achieved 85.27% of the commands certainly executed, with an uncertainty of 14.44% and 0.11% of Status missed certainly not executed. Moreover, unlike the controlled scenario in which the only consecutive lost cases of Status were found with $n = 2$ and $n = 3$, in the real scenario the number of cases is much wider, reaching values with n = 12. As it results from (2), it is possible to group the values of the uncertainties generically in even and odd cases resulting from the sum of the uncertainties calculated for each case.

VII. CONCLUSION

This work aimed to evaluate reception confirmation messages (Status) in the Bluetooth mesh network. The main weakness in the usage of this type of Status messages, in reply to a group address On/Off command, is that they could generate unexpected issues when multiple nodes respond simultaneously, and therefore reducing the probability of message delivery due to message collisions. Notably, the standard firmware implementation of mesh Bluetooth node, to date, does not seem to address a particular solution aimed to resolve or, at least limit, the issue of Status messages congestion.

In this work, we propose a technique of spreading Status overtime to overcome the reception confirmation message issues in a Bluetooth mesh network. To evaluate the performance of the proposed technique in terms of Status PSR, we arranged an experimental setup consisting of a Bluetooth mesh network operating in two different scenarios. The controlled scenario with limited interfering elements and the real scenario with numerous interfering and attenuating sources. To show a comparison between the standard Bluetooth mesh network procedure with the proposed technique implemented, we used the experimental setups to carry out two specific tests. The first test used a network with SC test, whereas the second test implemented the proposed technique with OCs (OC test).

In the controlled scenario the obtained results in terms of the overall success on executing On/Off light commands show that our contribution adopted in the OC, positively influences Status PSR, with on average improvement of 1.86%, with respect to the SC one. To confirm this, we observed a difference in the calculated variance for both tests carried out by different orders of magnitude, with an average variance computed on 0.44% for the SC and 0.002% for the OC. Furthermore, we also provided an in-depth performance evaluation method that uses only the information associated with the confirmation messages to analyze on the lost Status. From the analysis carried out in executing the On/Off commands, we found that the percentage of consecutive Status missed in the SC is much higher than those in the OC and that obtained for the isolated Status, which turned out to be more than double of its initial value. All these show that our contribution has led to a significant decrease in the number of consecutive Status lost. Thus, the uncertainty in the unequivocal determination of the execution or otherwise of the command of On/Off, obtaining a knowledge of the execution Status of the On/Off command on average of 99.99% of the total commands sent, for each test carried out. Therefore, in the OC test, we obtained an overall improvement concerning the network's general behaviour, for the total amount of Status messages received by the client and a general uniformity in the number of Status received by all the nodes, with respect to the SC. Furthermore, the proposed OC lead to a significant decrease in the number of consecutive Status lost, and the uncertainty in the unequivocal determination of the On/Off command execution.

In the real scenario, we obtained an average improvement of 2.88% in OC's test compared to the SC one. In this scenario as well as in the real one, the performance improvement obtained in OC test respect the SC one, is in any case significant as shown by the t tests conducted. In general, the overall PSRs of SC and OC tests in the real scenario are lower than the same tests carried out in the controlled scenario. This may be due to the numerous interfering elements of the real scenario, such as co-channel interference from a WIFI access point and other Bluetooth devices, attenuation and reflections from structural and furnishing elements and the variable presence of people occupying the environment during the trials. These variables are clearly not controllable, configuring as a source of random interference.

This article proposes a technique for improving the reception of confirmation messages based on a pseudorandom number generator algorithm with seed obtained from the first four-bytes of the Bluetooth device address. This technique represented a good compromise between lost packets reduction and low computational load, obtaining a performance improvement in the adopted evaluation tests. However, considering the increase in the number of devices in the network, it would be advisable to use a hash function that maps the six bytes of the unique address onto a decimal value used as the seed of the pseudorandom number generator algorithm. Moreover, high-performance algorithms for collision avoidance could be adopted, but they would have the drawback of increasing the computational loads. This aspects will be investigated in future work, including a greater number of tests performed in several challenging environments.

REFERENCES

[1] (2016). *Bluetooth SIG, Bluetooth Specification Version 4.0*. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[2] K.-H. Chang, "Bluetooth: A viable solution for IoT? [industry perspectives]," *IEEE Wireless Commun.*, vol. 21, no. 6, pp. 6–7, Dec. 2014.

[3] S. Raza, P. Misra, Z. He, and T. Voigt, "Bluetooth smart: An enabling technology for the Internet of Things," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, 2015, pp. 155–162.

[4] K. Sornalatha and V. R. Kavitha, "IoT based smart museum using Bluetooth low energy," in *Proc. 3rd Int. Conf. Adv. Elect. Electron. Inf. Commun. Bioinformat. (AEEICB)*, 2017, pp. 520–523.

[5] D. Hortelano, T. Olivares, M. C. Ruiz, C. Garrido-Hidalgo, and V. López, ' 'From sensor networks to Internet of Things. Bluetooth low energy, a standard for this evolution," *Sensors*, vol. 17, no. 2, p. 372, 2017.

[6] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, "BLE beacons for Internet of Things applications: Survey, challenges, and opportunities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 811–828, Apr. 2018.

[7] K. E. Jeon, J. She, J. Xue, S. Kim, and S. Park, "LuxBeacon—A batteryless beacon for green IoT: Design, modeling, and field tests," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5001–5012, Jun. 2019.

[8] (2016). *Bluetooth SIG, Bluetooth Specification Version 5.0*. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification

[9] C. A. G. da Silva, E. L. dos Santos, A. C. K. Ferrari, and H. T. dos Santos Filho, "A study of the mesh topology in a ZigBee network for home automation applications," *IEEE Latin America Trans.*, vol. 15, no. 5, pp. 935–942, May 2017.

[10] W. Rzepecki and P. Ryba, "IoTSP: Thread mesh vs other widely used wireless protocols—Comparison and use cases study," in *Proc. IEEE 7th Int. Conf. Future Internet Things Cloud (FiCloud)*, 2019, pp. 291–295.

[11] P. Pierleoni *et al.*, "The scrovegni chapel moves into the future: An innovative Internet of Things solution brings new light to Giotto's masterpiece," *IEEE Sensors J.*, vol. 18, no. 18, pp. 7681–7696, 2018.

[12] S. S. Ahmeda and E. A. Esseid, "Review of routing metrics and protocols for wireless mesh network," in *Proc. IEEE 2nd Pac.–Asia Conf. Circuits Commun. Syst.*, vol. 1, 2010, pp. 27–30.

[13] S. M. Darroudi and C. Gomez, "Bluetooth low energy mesh networks: A survey," *Sensors*, vol. 17, no. 7, p. 1467, 2017.

[14] (2019). *Bluetooth SIG, Mesh Profile Version 1.0.1*. [Online]. Available: https:www.bluetooth.com/specifications/mesh-specifications

[15] *Silicon Labs. An1142: Mesh Network Performance Comparison*. Accessed: Oct. 14, 2020. [Online]. Available: https://www.silabs.com/documents/public/application-notes/an1142-mesh-network-performancecomparison.pdf

[16] K. Abboud, Y. Li, and S. Bermudez, "eSNAP: Enabling sensor network automatic positioning in IoT lighting systems," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10427–10436, Oct. 2020.

[17] S. M. Darroudi, C. Gomez, and J. Crowcroft, "Bluetooth low energy mesh networks: A standards perspective," *IEEE Commun. Mag.*, vol. 58, no. 4, pp. 95–101, Apr. 2020.

[18] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu, "A survey on Bluetooth 5.0 and mesh: New milestones of IoT," *ACM Trans. Sensor Netw.*, vol. 15, no. 3, pp. 1–29, 2019.

[19] P. Di Marco, P. Park, M. Pratesi, and F. Santucci, "A Bluetooth-based architecture for contact tracing in healthcare facilities," *J. Sensor Actuator Netw.*, vol. 10, no. 1, p. 2, 2021. [Online]. Available: https://www.mdpi.com/2224-2708/10/1/2

[20] M. Jörgens *et al.*, "Bluetooth mesh networks for indoor localization," in *Proc. 20th IEEE Int. Conf. Mobile Data Manag. (MDM)*, 2019, pp. 397–402.

[21] Q. Wan and J. Liu, "Smart-home architecture based on Bluetooth mesh technology," in *Proc. IOP Conf. Mater. Sci. Eng.*, vol. 322, 2018, Art. no. 072004.

[22] (2019). *Bluetooth SIG, Mesh Model Version 1.0.1*. [Online]. Available: https:www.bluetooth.com/specifications/mesh-specifications

[23] *Bluetooth SIG, Bluetooth Mesh Networking: Paving the Way for Smart Lighting*. Accessed: Jun. 24, 2021. [Online]. Available: https://www.bluetooth.com/bluetooth-resources/bluetooth-mesh-paving-the-way-for-smart-lighting

[24] M. Ghamari *et al.*, "Detailed examination of a packet collision model for Bluetooth low energy advertising mode," *IEEE Access*, vol. 6, pp. 46066–46073, 2018.

[25] R. Rondón, A. Mahmood, S. Grimaldi, and M. Gidlund, "Understanding the performance of Bluetooth mesh: Reliability, delay and scalability analysis," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2089–2101, Mar. 2020.

[26] D. Pérez-Díaz-De-Cerio *et al.*, "Bluetooth mesh analysis, issues, and challenges," *IEEE Access*, vol. 8, pp. 53784–53800, 2020.

[27] D. Pérez-Díaz-De-Cerio, J. Valenzuela, M. Garcia-Lozano, Á. Hernández-Solana, and A. Valdovinos, "BMADS: BLE mesh asynchronous dynamic scanning," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2558–2573, Feb. 2021.

[28] E. D. Leon and M. Nabi, "An experimental performance evaluation of Bluetooth mesh technology for monitoring applications," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2020, pp. 1–6.

[29] A. S. Brandão, M. C. Lima, C. J. B. Abbas, and L. J. G. Villalba, “An energy balanced flooding algorithm for a BLE mesh network,” *IEEE Access*, vol. 8, pp. 97946–97958, 2020.
[30] *Nordic Semiconductor. nRF5 SDK for Mesh*. Accessed: Jun. 24, 2021. [Online]. Available: https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK-for-Mesh
[31] E. Whitley and J. Ball, “Statistics review 3: Hypothesis testing and p values,” *Crit. Care*, vol. 6, no. 3, pp. 1–4, 2002.

Paola Pierleoni received the master’s degree in electronic engineering and the Ph.D. degree in electrical engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 1991 and 1995, respectively.

In 1991, she joined the Department of Information Engineering, Universitá Politecnica delle Marche, where she is currently working as an Assistant Professor of Telecommunications. Her research topics include network protocols, wireless sensor networks, Internet of Things, signal processing, and embedded devices development.

Andrea Gentili received the master’s degree in electronic engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 2016, with the thesis focused on the analysis of parameters of the gait based on sEMG, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering.

His research fields of interest are digital sensors networks for Internet of Things, in particular for industrial lighting, as well as data and biological signals processing.

Marco Mercuri received the master’s degree (*cum laude*) in electronic engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in February 2019, where he is currently pursuing the Ph.D. degree with the Department of Information Engineering.

His research interests are in Bluetooth mesh architectures for IoT applications based on the Bluetooth 4 protocols and later versions.

Alberto Belli received the master’s degree in telecommunications engineering and the Ph.D. degree in biomedical, electronic, and telecommunications engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 2012 and 2016, respectively.

He is a Technician with the Department of Information Engineering, Universitá Politecnica delle Marche. His research interests are wireless sensor networks for Internet of Things, data fusion algorithms for array sensors, and wearable sensors.

Roberto Garello (Senior Member, IEEE) received the Dottorato di Ricerca degree in ingegneria elettronica from the Politecnico di Torino, Turin, Italy, in 1994.

From 1994 to 1997, he was working with Radio Link Laboratory of Marconi Communications, Genova, Italy. From 1998 to 2001, he was an Associate Professor with the University of Ancona, Ancona, Italy. Since November 2001, he has been an Associate Professor with Politecnico di Torino, whereas from 2006 to 2008, he served as a Coordinator of the Communication Engineering degree. His research interests are communication systems (xDSL, LTE, space links, indoor positioning), cognitive radio networks, and channel coding.

Lorenzo Palma received the master’s degree (*cum laude*) in electronic engineering and the Ph.D. degree in information engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 2012 and 2017, respectively.

In 2017, he had a research grant from Consortium GARR to develop a new system for seismic and structural monitoring based on IPv6 sensors able to provide data for the creation of an earthquake early warning system. He is currently a Research Fellow with the Universitá Politecnica delle Marche. His research interests are IoT, wireless sensors networks, IMU sensors, and embedded systems, and communication network.