RouMBLE: a Sink-Oriented Routing Protocol for BLE Mesh Networks

Luca Davoli, *Member, IEEE,* Massimo Moreni, and Gianluigi Ferrari, *Senior Member, IEEE*

***Abstract*—In Internet of Things (IoT)-like contexts, there is often the need to leverage traffic routing mechanisms among heterogeneous devices, especially when classical (and well-known) addressing paradigms cannot be adopted or supported by constrained IoT devices deployed *on the field* (e.g., due to memory footprint, internal limitations, etc.). This is even more true (and necessary) when nodes interact in unstructured networks (e.g., mesh-like) lacking a specific topology (e.g., exploiting flooding approaches to transfer information) and external "smart" devices should be allowed to interact with these networks. To this end, in this paper a multi-sink routing protocol, denoted as Routing on Mesh Bluetooth Low Energy (RouMBLE), is proposed. Our implementation relies on BLE advertisement channels and allows sink nodes to control topology formation and data collection (with both unicast and broadcast communications), with nodes identified with compressed addresses. A relevant experimental application to environmental lighting management is presented.**

***Index Terms*—Wireless Mesh Network, Routing Protocol, Sink Node, Gateway, Topology.**

I. Introduction

ONE of the main aspects characterizing the Internet of Things (IoT) regards the need to let "things" interact through heterogeneous communication protocols, according to a "system of systems" paradigm. To this end, one of the most interesting and promising (due to its widespread adoption) technologies available in devices exchanging data everyday—in Human-to-Machine (H2M) and Machine-to-Machine (M2M) ways—is Bluetooth Low Energy (BLE). BLE has emerged as a major low-power wireless technology, allowing low-energy communication in heterogeneous environments and contexts (e.g., with sensors, actuators, smartphones, wearables, and so on). Moreover, even standardization entities—such as the Internet Engineering Task Force (IETF) [1] and the Bluetooth Special Interest Group (SIG) [2]—are defining innovative adaptation mechanisms to facilitate the interaction of BLE nodes within Internet of Things (IoT)-oriented scenarios—e.g., adaptation layers to support IPv6 over BLE [3].

BLE networks are traditionally organized with a star topology. Therefore, they support limited coverage range [4] and this can represent a limitation in some scenarios (e.g., urban, agricultural, industrial, etc.) [5]. Direct interaction between devices may allow to overcome this limitation. To this end, BLE-based mesh networks [6, 7] allow to overcome the coverage limitations of a classical star topology [8, 9].

Another challenge is related to the way in which BLE nodes can be addressed in such a network. This aspect is crucial in real deployments, especially with constrained IoT nodes (e.g., in terms of power constraints, memory footprint, etc.). The IPv6 protocol may support network integration: this holds true, in particular, for a BLE network, as proposed in the literature [3, 10, 11]. However, IPv6 adoption may be critical. Therefore, alternative addressing approaches are needed to support services on top of BLE networks.

In this paper, we present and discuss a BLE-oriented multi-sink routing protocol, denoted as Routing on Mesh BLE (RouMBLE), for heterogeneous Point-to-Point (P2P), MultiPoint-to-Point (MP2P), and broadcast-like communication scenarios, where Over-the-Air (OTA) operations and update mechanisms are needed. In particular, RouMBLE allows to route data collected by sensing nodes deployed in an environment (e.g., *on-field* Passive InfraRed, PIR, sensors) through BLE mesh nodes. As a reference scenario, a BlueLightLink (BLL) [12] network for environmental lighting management will be considered.

RouMBLE is inspired by the Better Approach to Mobile Ad-hoc Networking (B.A.T.M.A.N.) routing protocol [13] and is designed to be deployed on BLE devices composed of a host micro-controller and a BLE System-on-Chip (SoC) managing all the networking functionalities. Owing to its data packets' format, RouMBLE exploits BLE advertisement channels to transmit data with a flooding approach. The BLE nodes are assigned the following roles: (i) *on-field* devices, in charge of sensing the environment and performing tasks; (ii) management devices, denoted as *sink nodes*, in charge of managing the behaviour of *on-field* devices and requesting them to perform particular actions; and (iii) external nodes, denoted as *smart devices*, interested in obtaining information from the sink nodes for further actions (e.g., visualization, data analysis, etc.). Finally, it should be highlighted that even if RouMBLE has been designed to rely on BLE advertisement channels, it may be applied in a transparent way to other Wireless Mesh Networks (WMNs) which could benefit from such a traffic routing mechanism (e.g., protocols relying on flooding mechanisms).

The remainder of this paper is organized as follows. In Section II, a discussion on existing mesh-oriented solutions is presented. In Section III, the main packet types defined in RouMBLE are introduced, while the network topology construction is detailed in Section IV. In Section V, data collection

L. Davoli and G. Ferrari are with the Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy, with the National Inter-University Consortium for Telecommunications (CNIT), Research Unit of Parma, Parma, Italy, and with things2i s.r.l., Parma, Italy. E-mail: luca.davoli@unipr.it, gianluigi.ferrari@unipr.it.

M. Moreni is with TCI Telecomunicazioni Italia s.r.l., Saronno, Italy. E-mail: m.moreni@tci.it.

Manuscript received October X, 2024; revised December Y, 2024.

TABLE I: Literature works categorized along their main features.

Reference	Flooding	Routing	Open source	Multiple GWs supported in the WSN	General applicability	Routing algorithm adopted in the WSN(1)	Compute multiple path between nodes	Exploit routing table at run-time	Metric exploitation
[24]	✓	✗	✓	✗	✗	Trickle	✗	✗	✗
[25]	✓	✗	✓	✗	✗	BLEmesh	✗	✗	✓(2)
[26]	✓	✗	✓	✗	✓	ExOR	✗	✗	✓(2)
[27]	✓	✗	✓	✓	✓	SOAR	✓	✓	✓(2)
[28]	✗	✓	✓	✗	✗	✗	✗	✗	✗
[29]	✗	✓	✓	✗	✗	✗	✗	✗	✓(3)
[30]	✗	✓	✓	✓	✓	✗	✗	✓	✗
[31]	✗	✓	✓	✓	✓	RPL	✗	✓	✓(4)
[32]	✗	✓	✓	✗	✗	On-demand approach	✗	✓	✗
[33]	✗	✓	✓	✗	✗	NDN	✗	✗	✗
[34]	✗	✓	✓	✓	✓	TORA	✓	✓	✓(5)
[35]	✗	✓	✓	✓	✗	WRP-Lite	✗	✓	✓(6)
[36]	✗	✓	✓	✓(7)	✗	KRP	✓	✓	✓(5)
[37]	✗	✓	✓	✗(8)	✗	MRP	✗	✗	✓(5)
[38], [39], [40], [41], [42], [43], [44]	✗	✓	✗	✗	✗	✗	✗	✗	✗
RouMBLE	✓		✓	✓	✓	RouMBLE	✓	✓(9)	✓(10)

(1) No need of perform *on-the-flight* agreements between all nodes receiving the same packet, to select the next hop.
(2) Based on ETX.
(3) In the experimental evaluation, not in the analytical modelling of the protocol: (i) minimum connection interval, (ii) worst-case end-to-end latency.
(4) Based on residual energy and degree of the device.
(5) Based on hop counter.
(6) Based on link cost: if the link fails, the cost is set to infinity.
(7) Connected to a wired backbone network.
(8) Every client selects a single GW to connect to Internet.
(9) Routing paths discovered during BOM replies collection.
(10) Based on hop counter and number of received BOMs.

from *on-field* mesh nodes is detailed, while reference use cases and application functionalities enabled by RouMBLE are presented in Section VI, with an experimental performance evaluation being discussed in Section VII. Finally, in Section VIII we draw our conclusions.

II. Related Works

In the literature, Wireless Mesh Networks (WMNs) have been extensively investigated in terms of design aspects [14, 15], as well as routing metrics [16, 17]. In the context of BLE, BLE SIG and IETF have guided this process, creating the Bluetooth Smart Mesh Working Group [18] and proposing to adopt the IPv6 Low-Power Wireless Personal Area Networks (6LoWPAN) protocol [19] to support IPv6 over BLE networks [20]–[23]. To this end, there exist two main approaches to data transmission in BLE mesh networks, namely: (i) flooding-based and (ii) routing-based. For the sake of clarity and completeness, in Table I a summary of the literature works discussed in the following is reported, categorizing them along their main features to better underline the novelty of the proposed RouMBLE protocol.

Flooding-based solutions do not perform any kind of routing among the nodes composing the network, but, rather, broadcast packets over BLE advertising channels. An example of this approach is proposed in [24], where authors analyze the performance of a routing algorithm exploiting Trickle [45] and operating in a BLE mesh Wireless Sensor Network (WSN), focusing only on disaster seismic events monitoring and prediction, and supporting only one sink node in the network. In detail, the authors focus on how to keep energy consumption low and, at the same time, to bound latency and Packet Delivery Ratio (PDR), introducing a simple sleep/awake schedule. Similarly, in [25] the authors propose BLEmesh, a flooding mechanism limiting re-broadcasting in intermediate nodes by only admitting a sub-set of these nodes, selected on the basis of the Expected Transmission Count (ETX) [46], to perform broadcasting operations. In detail, being only applicable to BLE networks, BLEmesh forces the traffic to be sent as batch of data—guaranteeing ordered delivery and other—and exploits opportunistic routing. BLEmesh is thus forced to rely on the presence of a list of participating nodes (denoted as Forwarder List) sorted by priority, whose size is derived from the number of participating nodes and which is periodically flooded by the source node initiating a data transmission. The authors in [26] propose a flooding-oriented integrated routing and MAC protocol targeting multi-hop wireless networks, denoted as ExOR, which, unfortunately, faces the following four main challenges: (i) nodes inside the WSN have to agree on the sub-set of devices which will receive each information packet—this agreement phase introduces overhead in the communication; (ii) the protocol must have a metric reflecting the likely cost of moving a packet from any node

to the destination; (iii) there is the need to define proper policies to select useful nodes as WSN participants; and (iv) similarly to [25], the protocol operates on batches of packets, instead of managing single information contents, thus requiring to include a list of candidate forwarders in each packet (prioritized by the estimated cost to the destination node) and a copy of the sender's batch map, containing the sender's best guess of the highest priority node to have received each packet in the batch. In [27], a similar routing protocol, denoted as Simple Opportunistic Adaptive Routing (SOAR), supporting multiple simultaneous flows in WMNs, is proposed. In detail, SOAR requires every WMN node to periodically measure links quality (in terms of ETX) and maintain the entire network topology. Then, on the basis of the collected information, SOAR selects the default path and a list of (next-hop) forwarding nodes eligible for broadcasting data (including this additional overhead).

Routing-based solutions adopt a routing protocol for packet forwarding and transmit data over BLE data channels. These solutions can further be separated in static and dynamic routing. An example of static routing schemes is discussed in [28], where the authors propose a static tree topology serving as network backbone for a BLE-based WSN with a 2-byte addressing space and involving nodes with different roles: (i) root node (acting as a central device, as per BLE specifications lexicon) collecting data from the WSN; (ii) intermediate nodes, physically formed combining two BLE devices—one *peripheral* node and one *central* node—in one package; and (iii) leaf nodes, performing as peripheral entities. Unfortunately, this solution (i) being tree-oriented, lacks a mechanism to rebuild the network after a node or link failure, (ii) suffers from the single-node failure problem, (iii) involves an addressing enumeration strictly dependent on the depth which a node is located at in the WSN, and (iv) is applicable only to BLE networks. In [29], another static routing solution, denoted as Real Time BLE (RT-BLE), is proposed for industrial WMNs. RT-BLE creates multiple networks (denoted as *sub-networks*), each coordinated by a master node and requiring multiple slave nodes (acting as "bridges") between sub-networks to enable data sharing. Therefore, this solution lacks of scalability for both masters and slaves—keeping only a default route and an alternative route as a backup, and with a master able to establish a connection with at most another master. On the other hand, examples of dynamic routing schemes have been proposed in [30], where a BLE mesh solution, denoted as MultiHop Transfer Service (MHTS) and based on next-hop on-demand routing over the Generic ATTribute (GATT) layer, is discussed. In order to enable multi-hops data transfer, MHTS forces the usage of four specific read/write GATT characteristics, and requires the definition of two mechanisms: (i) connection discovery and establishment with neighboring nodes, and route discovery toward distant nodes; (ii) data storing inside intermediate nodes, and transmission handling toward the destination node. In [31], another routing-based solution, denoted as BLE Mesh Network (BMN) and similar to the Adaptation Layer between BLE and RPL (ALBER) protocol [47], exploiting a Directed Acyclic Graph (DAG) structure—inspired by the IPv6 Routing Protocol for low Power and Lossy networks (RPL) [48]—for transmitting routing messages via advertising channels, is proposed. Similarly, in [32] an on-demand routing protocol targeting the formation of scatternets—network topologies composed of interconnected piconets, so strictly applicable only to BLE networks—is proposed. In [33], authors leverage attributes, characteristics, and GATT services to apply Named Data Networking (NDN) [49, 50] to support BLE mesh networks. Finally, several routing protocols have been proposed in the literature for Mobile Ad-Hoc Network (MANET)-based scenarios [34]–[36, 51, 52]. In particular, a distributed routing protocol, denoted as Temporally-Ordered Routing Algorithm (TORA), structured as a temporally-ordered sequence of diffusing computations, each consisting of a sequence of directed link reversals, is proposed in [34]. In [35], a table-driven routing protocol using non-optimal routes, requiring the usage of both a routing table and a distance table, applicable only to IP-based networks, and denoted as Wireless Routing Protocol (WRP), is introduced. Then, a "beaconing" approach denoted as k-hop Routing Protocol (KRP), focusing on route discovery and maintenance employing the ad-hoc mode in IEEE 802.11 networks, extending the Ad-Hoc On-Demand Distance Vector (AODV) [53] protocol, and requiring a fixed backbone network of sink nodes, is proposed in [36]. Finally, in [37], it is discussed how routing functionalities (performed through a new routing paradigm, denoted as Mesh Routing Protocol, MRP) may introduce benefits in MANETs managing traffic flows only arriving from Internet. Unfortunately, MRP forces each node to reach only one sink node, preventing the use of multiple sink nodes. Moreover, it needs to rely only on Linux-based OSs, being implemented at *user space* and having to interface with the kernel only through the routing table changing calls.

Finally, the interest towards BLE mesh networks has led also to proprietary network solutions, including: CSRmesh [38], OpenMesh [39], Wirepas Mesh [40], MeshTek [41], EtherMind [42], and solutions from Estimote [43] and NXP [44]. Unfortunately, due to their closed nature, these solutions are not amendable to extensions or integration into devices and nodes not manufactured by the original manufacturers.

Unlike existing works, it is of interest to connect BLE nodes using routing algorithms even in scenarios oriented to the use of advertisement channels (as in flooding-based solutions). This is exactly the aim of the BLE mesh-oriented routing protocol proposed in the remainder of this paper, namely RouMBLE.

III. A Sink-Oriented Routing Protocol

As anticipated in Section I, RouMBLE has been inspired (as general idea) by the B.A.T.M.A.N. routing protocol, even if differences among RouMBLE and the Layer-2 (L2)/Layer-3 (L3) B.A.T.M.A.N. protocol arise. In detail, B.A.T.M.A.N. has been developed for multi-hop mesh networks composed by Linux-compliant devices and mainly features two operational components: (i) `batman-adv` [54, 55], providing routing functionalities at L2 (Ethernet layer), and (ii) `batmand` [56,

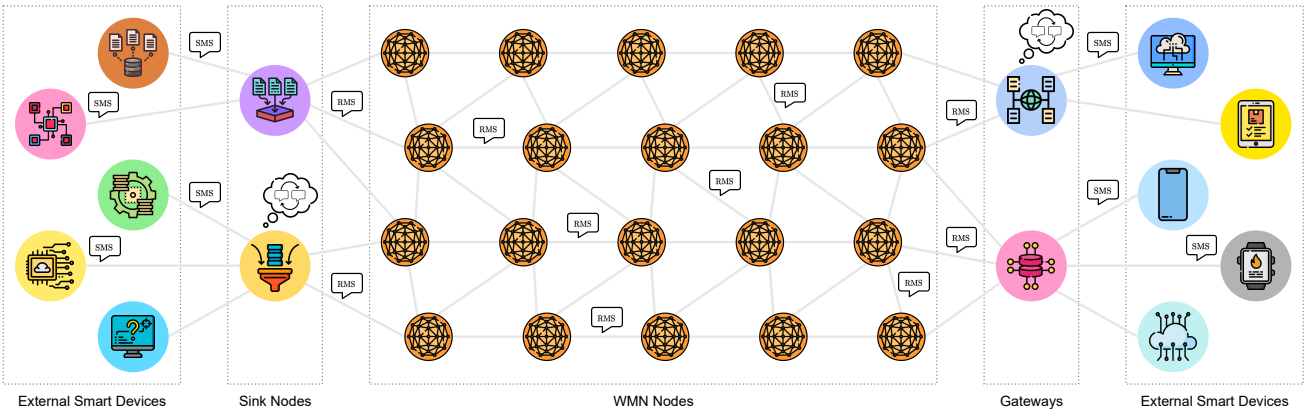

Fig. 1: Example of a generic BLE-based mesh network composed by heterogeneous devices.

57], operating at L3. In particular, `batman-adv` is implemented as a Linux kernel module and provides a virtual network interface (denoted as `bat0`) that transparently forwards data packets (not only routing information) using *raw* Ethernet frames. In fact, all nodes participating to the mesh network appear to be link-local and are unaware of the network's topology and unaffected by any network changes. Instead, `batmand` corresponds to a L3 Linux single user-space routing daemon exploiting UDP packets for managing the mesh network, hence requiring the involved nodes to be IP-compliant. This prevents B.A.T.M.A.N. to be applied to BLE-based mesh networks, which instead is the main target of the proposed RouMBLE protocol.

Nevertheless, as will be clarified looking at the definition of RouMBLE (in the remainder of this work), RouMBLE shares with B.A.T.M.A.N. the idea of featuring (in the mesh network) the presence of nodes transmitting (during specific time intervals, denoted as *originator intervals*) broadcast messages, referred to as *originator messages* (OGMs), to inform neighbours about their existences. In detail, OGMs are re-broadcasted by mesh nodes according to specific rules (thus flooding the mesh network itself), with a specific 52-byte OGM *raw* packet including IP and UDP overheads, and at least the address of the originator node, the address of the node transmitting the packet, a Time-To-Live (TTL), and a sequence number. In fact, this approach allows to share the knowledge of *end-to-end* paths between mesh nodes to all participating nodes. *However*, while B.A.T.M.A.N. fosters the possibility for each node to extract and maintain only the information about its best next-hop towards any other node, it will be shown that RouMBLE allows to maintain information on multiple paths, collecting knowledge also on "backup" routes weighed on the basis of specific storage policies and needs (e.g., given a specific service to be provided by the BLE mesh network). *Finally*, since B.A.T.M.A.N. (through its `batmand` component) maintains a sort of "tunnel connection" to every "B.A.T.M.A.N. Internet client" having to access the IP-compliant mesh network, it follows that every data has to go through this tunnel. Since this is a Linux user-space tunnel, a huge amount of copy operations between user-space and kernel areas is necessary, and this might have a detrimental impact (resulting as a bottleneck for the system) depending on the number of clients and the (CPU) computational power available at the mesh nodes.

Focusing on RouMBLE, it has been designed for BLE-based mesh networks composed by the following types of mesh nodes, depicted in Fig. 1:

- *on-field* devices, often based on commercial SoCs, in charge of sensing the environment (e.g., through external sensors connected directly to the device itself) and performing tasks (e.g., through the activation of specific registers on the host micro-controller handling the physical outputs);
- sink nodes, in charge of managing the behaviour of *on-field* devices and requesting them to perform particular actions;
- external smart devices, interested in obtaining data from the *on-field* nodes in the BLE-based mesh network: this happens through the (intermediate) sink nodes managing the *on-field* nodes and allowing the external devices to perform further actions (e.g., data analysis, visualization, etc.).

According to their nature, sink nodes can be considered as "frontier" entities allowing traffic to flow into/out of the BLE-based mesh network. For this reason, different types of data messages[1] have been defined in RouMBLE, as will be detailed in the following.

However, before focusing on the data packets to be used in the proposed routing protocol, an overview on how mesh nodes can be identified in RouMBLE is expedient. To this end, as shown in Fig. 2, in addition to the MAC address each device is assigned the following two addresses:

- 2-byte length address, adopting a binary encoding scheme (e.g., `0001 1010 1011 0011`);
- 4-byte length address, where each byte corresponds to a `HEX` character (e.g., `1AB3`).

In detail, under the assumption of identifying a mesh node with an address in the range `0000-79FF`, the encoding/decoding conversion mechanism shown in Fig. 2 is obtained considering only the LSB part of each byte and

[1]In the following, the terms "packets" and "messages" will be used with the same meaning.

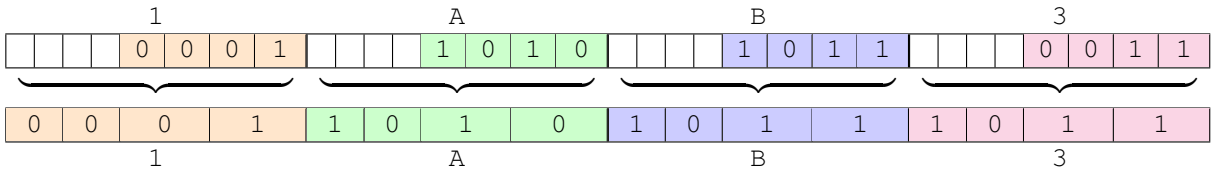

Fig. 2: Address conversion mechanism adopted in the proposed mesh-oriented routing protocol RouMBLE.

$\mathcal{P}_{\text{SMS}}$	$\mathcal{D}_{\text{TYPE}}$	$\mathcal{A}_{\text{TYPE}}$	$\mathcal{A}_{\mathcal{T}}$	$\mathbb{P}$

Fig. 3: Structure of an SMS packet sent from an external smart device toward the core part of the (e.g., BLE-based) mesh network.

combining together two LSBs into a resulting unified byte, thus reducing from 4 bytes to only 2 bytes. As a consequence, it is possible to encode/decode two HEX characters using 1 byte: the MSBs represent the first character and the LSBs represent the second character.

For the sake of completeness, in Table II the notation adopted in this paper is summarized. A detailed discussion on the main data packets defined in RouMBLE, namely *Send MeSsage* (SMS) and *Receive MeSsage* (RMS) packets, is carried out in Subsection III-A and Subsection III-B, respectively.

A. Send MeSsage (SMS) Packet

SMS packets are used by an external smart device to ask the sink node to send a request in the mesh network itself. The general structure of an SMS packet is shown in Fig. 3, where the various parts have the following meaning.

- $\mathcal{P}_{\text{SMS}}$ identifies a message as SMS packet, to be used by a smart device to interact with a sink node.
- $\mathcal{D}_{\text{TYPE}}$ identifies the type of service requested by a smart device to be executed inside the mesh network by mesh node(s)—e.g., useful in the case the target BLE mesh node(s) will be able to perform different services.
- $\mathcal{A}_{\text{TYPE}}$ specifies the destination node's address type $\mathcal{A}_{\mathcal{T}}$, and can assume the following acceptable values: *N* (unicast), *G* (group), *U* (unit), *K* (ACK).
- $\mathcal{A}_{\mathcal{T}}$ contains the address of the destination node, as well as an indication of a group of nodes, which the request/command contained in the SMS packet should be delivered to. Moreover, $\mathcal{A}_{\mathcal{T}}$ and $\mathcal{A}_{\text{TYPE}}$ represent the routing information to be used inside the BLE-based mesh network to forward data in the proper way (as will be discussed in Subsection III-B).
- $\mathbb{P}$ contains the payload information sent from the external smart device (preparing the SMS packet) toward the interested mesh target node(s) in the BLE-based mesh network itself.

When a sink node receives (via BLE advertisement channels) an SMS packet sent from an external smart device, it may perform some preliminary validation steps (based on the specific implementation of the sink node itself). Once these steps are completed, the sink node needs to translate the incoming SMS message into a corresponding data structure recognizable by mesh nodes and able to flow inside the BLE-based mesh network.

TABLE II: Main notation adopted in the paper.

$\mathbb{H}$	Header of a generic packet used in RouMBLE.
$\mathbb{U}$	Identifier of generic packet, defined as an incremental integer value.
h, h_{MHN}	Hop counter contained in the header of a packet, and maximum number of hops to be traversed before considering a packet as "outdated."
$\mathcal{P}_{\text{TYPE}}$	Field reserved for the definition of the packet type to be used in RouMBLE.
$\mathcal{P}_{\text{SMS}}$, $\mathcal{P}_{\text{RMS}}$	Identifiers of an SMS packet and an RMS packet.
$\mathcal{C}_{\text{BOM}}$, $\mathcal{C}_{\text{GET}}$, $\mathcal{C}_{\text{SEN}}$	Identifiers of a BOM packet, a GET packet, and a SEN packet.
$\mathcal{D}_{\text{TYPE}}$, *A*, *S*, *L*, *M*	Type of service requested by a smart device to be executed inside the mesh network by WMN nodes, and allowed values: *All*, *Sensor*, *Light*, *Actuator*.
$\mathcal{A}_{\text{TYPE}}$, *N*, *G*, *U*, *K*	Type of address of a WMN node, and allowed values: *unicast*, *group*, *unit*, *ACK*.
$\mathbb{P}$	Field containing the payload of a packet used in RouMBLE.
$\mathcal{S}_x$, $\mathcal{A}_{\mathcal{S}_x}$	Generic WMN node x, and its corresponding address in the WMN.
$\mathcal{S}_\varepsilon$, $\mathcal{A}_{\mathcal{S}_\varepsilon}$	Randomly-chosen WMN node, and its corresponding address in the WMN.
$\mathcal{A}_{\mathcal{T}}$	Address of a target WMN node to be reached by a specific packet.
FF	Broadcast address for a transmission toward all the devices in the WMN.
$\mathcal{G}_z$, $\mathcal{A}_{\mathcal{G}_z}$	Generic sink node z, and its corresponding address in the WMN.
$n_{\mathcal{S}_x}$	Amount of BOM packets received by the WMN node $\mathcal{S}_x$.
$d_{\mathcal{S}_x}$	Distance (in terms of number of hops) of the WMN node $\mathcal{S}_x$ toward a sink node.
$N^{(\mathcal{G}_z)}$	Total amount of BOM packets received from the sink node $\mathcal{G}_z$.
k, k_{NTM}, k_{TO}	Type of information a sink node is interested in from the WMN, and specific values defined for topology mapping and "Tracking One" functions.
$\mathbb{R}$	Set of routing tables defined in the WMN nodes.
$\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$	Routing table maintained by the WMN node $\mathcal{S}_x$ with regard to the sink node $\mathcal{G}_z$.
R_{MRFS}	Maximum number of records to be maintained by a single routing table.
w_y, Ω_{BS}	Identifier of a group of WMN nodes having to reply with their data to a GET request, and pre-defined group block size to be used to decide if a single WMN has to reply.
γ_{RSSI}, $\gamma_{\text{RSSI-TO}}$	Threshold on the RSSI to mark a WMN node as near to the sensing node (in the heatmap scenario), and more stringent threshold (to be used in the "Tracking One" scenario).
Π_{SENSE}, Π_{PROX}	Sensing table and proximity table.

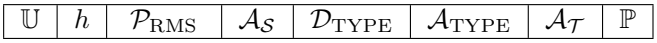

Fig. 4: Structure of an RMS packet defined in the routing protocol RouMBLE.

B. Receive MeSsage (RMS) Packet

An RMS packet is the only packet allowed, *by-design*, to flow inside the BLE network and managed by internal mesh nodes. In detail, an RMS packet is exploited to carry information requests and responses inside the BLE network, maintaining all the data needed to identify the entities involved in the interaction and the actions to be performed by *on-field* nodes of the mesh network itself.

As shown in Fig. 4, an RMS packet is composed by the following fields.

- $\mathbb{U}$ represents the identifier of the packet (as an incremental integer value) and is defined by the RMS packet originator node.
- h represents the current hop counter and defines the behavior to be adopted in managing the current RMS packet, e.g., if it needs to be re-transmitted inside the mesh network—and forwarded towards next-hop mesh node(s)—or if it has to be discarded. In detail, h is compared with a reference value (denoted as h_{MHN}) corresponding to the maximum number of hops that can be traversed inside the mesh network before considering the current RMS packet as "outdated." By default, $h_{\mathrm{MHN}} = 3$, which corresponds to a reasonable value for a not-too-large mesh network relying on the use of flooding. In general, h_{MHN} can assume values in the range $\{1, 2, \ldots, 10\}$. At the node emitting the RMS, $h = h_{\mathrm{MHN}}$. Considering the node emitting the RMS as the first node, at the t-th node ($t \in \{2, \ldots, h_{\mathrm{MHN}}\}$) the number of remaining hops $h(t)$ can be expressed as follows:
$$h(t) = h(t-1) - 1\,.$$
At this point: if $h(t) > 0$, the RMS packet is retransmitted; if $h(t) = 0$, the RMS packet is dropped.
- $\mathcal{P}_{\mathrm{TYPE}}$ identifies the message type and is set to $\mathcal{P}_{\mathrm{RMS}}$, being an RMS packet the only packet allowed to flow inside the reference mesh network.
- $\mathcal{A}_{\mathcal{S}}$ contains the address of the source node generating the RMS packet and will not be modified by intermediate nodes during its transmission inside the mesh network.
- $\mathcal{D}_{\mathrm{TYPE}}$ is derived from the corresponding field contained in the original SMS packet and, as highlighted in Subsection III-A, specifies the interested service provided by the destination node to be reached by the RMS packet—useful in the case the destination node can provide different services—and may assume a specific range of values (e.g., *S*, *M*, *L*, *A*), as listed in Table II.
- $\mathcal{A}_{\mathrm{TYPE}}$ and $\mathcal{A}_{\mathcal{T}}$ are derived from the corresponding SMS packet (as shown in Subsection III-A) and identify the address class (e.g., *N*, *G*, *U*, and *K*, as listed in Table II) and the specific address of the destination node, respectively. Then, on the basis of on the value of $\mathcal{A}_{\mathrm{TYPE}}$, $\mathcal{A}_{\mathcal{T}}$ may correspond to: (i) the address of a single target node (for a unicast transmission, with $\mathcal{A}_{\mathrm{TYPE}}$ = *N*), (ii) a group address (for a group communication, with $\mathcal{A}_{\mathrm{TYPE}}$ = *G*), or (iii) the broadcast address `FF` (for a broadcast transmission toward all the devices in the mesh network).
- $\mathbb{P}$ is derived *as-is* from the original SMS packet and contains the payload to be sent to the destination node.

In addition to the validation checks mentioned in Subsection III-A, the application component (e.g., a software daemon) handling the RMS packets internally in a node checks the node address $\mathcal{A}_{\mathcal{T}}$, in order to understand if the packet is for itself or if it needs to be re-transmitted inside the network (e.g., across BLE advertising channels). Should this be a broadcast message (e.g., with $\mathcal{A}_{\mathcal{T}}$ = `FF`), then no further checks, based on the value of the field $\mathcal{A}_{\mathrm{TYPE}}$, will be performed. Moreover, as part of this additional admission check stage, the daemon would be able to filter and remove repeated messages (to avoid to flood the network): this is the case, for example, that the sender of the RMS packet is itself. Finally, if all checks are satisfied and $1 \leq h \leq h_{\mathrm{MHN}}$, the RMS packet may be re-transmitted with different priorities: if $\mathcal{A}_{\mathrm{TYPE}}$ = *U* or $\mathcal{A}_{\mathrm{TYPE}}$ = *G* the priority is high; if $\mathcal{A}_{\mathrm{TYPE}}$ = *K* the priority is mid/low.

IV. Network Topology Construction

As highlighted in Section III, RouMBLE focuses on the introduction of routing functionalities in mesh networks with flooding-like information exchange (namely, BLE mesh networks exchanging data through BLE advertisement channels). To this end, a mechanism enabling the definition of a topology—as well as multiple ones—oriented to sink nodes and based on routing tables maintained by each mesh node participating to the network itself, should be exploited. In detail, as will be highlighted in the following, this allows to provide the overall mesh network with multiple topological overlays (each one referring to a specific sink node), thus improving the reliability of the entire network.

A. BLE Originator Message (BOM) Packet

In order to start a topology construction (with its consequent routing tables), in RouMBLE a particular RMS packet, denoted as *BLE Originator Message* (BOM), is defined. The BOM can be generated only by a sink node (and not by intermediate mesh nodes, by construction) and propagates inside the mesh network through flooding. In the following, for the sake of simplicity, the case of a single sink node instantiating BOMs will be considered. However, the proposed approach already encompasses the presence of multiple sink nodes, also in the presence of constrained nodes with limited memory (as the routing tables involved in RouMBLE would have a limited flash memory footprint).

As shown in Fig. 5, with period T_{BOM} (dimension: [s]) the sink node instantiates a new BOM packet (shown in

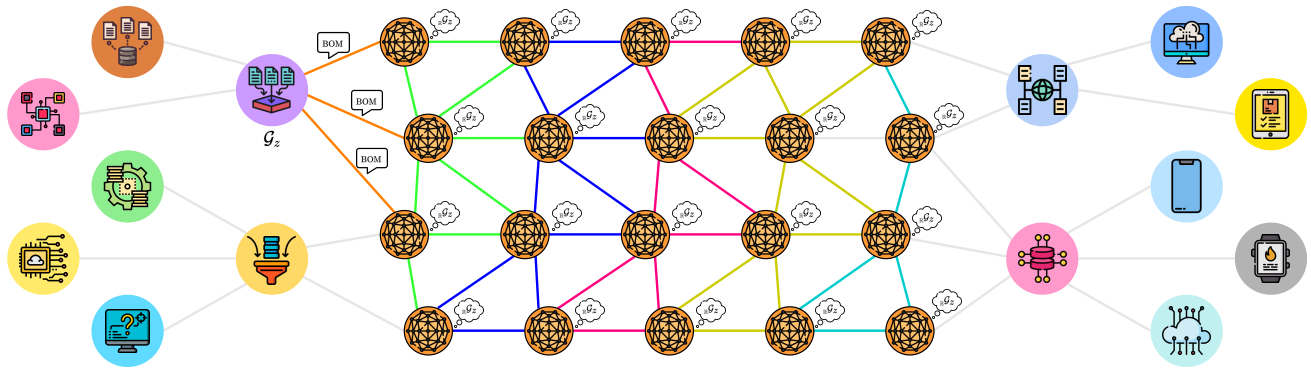

Fig. 5: BOM packets flooding into the BLE-based mesh network for topology construction.

$\mathbb{H}$	$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$	L	N	FF	$\mathcal{C}_{\text{BOM}}$	$\mathcal{A}_{\mathcal{G}_z}$	d

Fig. 6: Structure of a BOM packet used to build a topology in the BLE-based mesh network.

Fig. 6) carrying the following relevant information for the construction of routing tables:

- $\mathbb{H}$ corresponds to the packet header and, as shown in Fig. 4, contains both the packet identifier $\mathbb{U}$ and the information on the already-traversed hops h—by construction, $h = h_{\text{MHN}}$, since the BOM is generated *ex-novo*.
- $\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$ contains the address of the sink node originating the BOM packet—by construction, $\mathcal{A}_{\mathcal{S}_{\text{CURR}}} = \mathcal{A}_{\mathcal{G}_z}$, since the BOM is generated *ex-novo*.
- $\mathcal{D}_{\text{TYPE}} = L$ identifies the class of those nodes to be involved in the topology construction (for example, only nodes providing L = light services may be involved).
- $\mathcal{A}_{\text{TYPE}} = N$ contains the address type (N = unicast).
- $\mathcal{A}_{\mathcal{T}} = \texttt{FF}$ identifies the broadcast address and, according to the details given in Subsection III-B, each mesh node in the network should handle BOM packets, regardless of the specific value of the field $\mathcal{A}_{\text{TYPE}}$.
- $\mathcal{C}_{\text{BOM}}$ marks the current message as a BOM packet.
- $\mathcal{A}_{\mathcal{G}_z}$ contains the address of the sink node $\mathcal{G}_z$ originating the topology construction (through the BOM packet) and is expressed with the 2-byte binary encoding scheme shown in Fig. 2.
- d maintains the distance (e.g., the depth in terms of number of hops) between the sink node $\mathcal{G}_z$, originating the topology construction operation, and the node receiving the BOM. d is incremented at each traversed node. However, its meaning should not be confused with that of the field h, as h is increased at each RMS repetition and forces to discard the message when it reaches a value equal to h_{MHN}.

The BOM packet originates from the sink node and propagates inside the network till it reaches (thanks to the field $\mathcal{A}_{\mathcal{T}}$ equal to FF) all nodes.

As shown in Algorithm 1, when a generic node $\mathcal{S}_x$ (identified by its address[2] $\mathcal{A}_{\mathcal{S}_x}$) receives a packet, it performs the following checks:

1) it verifies if the received message is a BOM packet: this is true *iif* the payload $\mathbb{P}$ contains the BOM identifier $\mathcal{C}_{\text{BOM}}$;
2) it verifies if the current node $\mathcal{S}_x$ still corresponds to the sink node $\mathcal{G}_z$ originating the BOM packet: this is true *iif* $\mathcal{A}_{\mathcal{S}_x} = \mathcal{A}_{\mathcal{G}_z}$ and, should this be the case, the BOM packet is discarded;
3) it extracts the distance between the originator sink node $\mathcal{G}_z$ and the current node $\mathcal{S}_x$ from the field d.

At this point, the knowledge obtained in the third verification step requires to perform a lookup operation in the internal routing tables of the node $\mathcal{S}_x$, denoted as $\mathbb{R}_{\mathcal{S}_x}^{[\cdot]}$ and containing information useful for routing purposes, such as: the identity of the next-hop node from which a BOM has been received; the distance (as number of hops) from the specific sink node $\mathcal{G}_z$; and a ranking among different table records.

Therefore, with regard to the third verification step, if the originator sink node $\mathcal{G}_z$ is considered "too far" from the current node $\mathcal{S}_x$, then the BOM packet is discarded. Otherwise, if the number of records for the specific sink node $\mathcal{G}_z$ in the corresponding routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ has not already reached the maximum admissible amount (denoted as R_{MRFS}), then the current node $\mathcal{S}_x$ will (i) update its routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ about the information related to the specific originator sink node $\mathcal{G}_z$ (as discussed in Subsection IV-B) and (ii) generate a new BOM packet (as discussed in Subsection IV-C). Hence, a generic node $\mathcal{S}_x$ may avoid to modify (in fact corrupting) and retransmit the "old" BOM packet (received by its "predecessor" parent).[3]

In the presence of multiple sink nodes, instead of defining a separate routing table $\mathbb{R}_{\mathcal{S}_x}^{[\cdot]}$ for each sink node that will become known to the node $\mathcal{A}_{\mathcal{S}_x}$ itself, it might be possible to use a unique routing table $\mathbb{R}_{\mathcal{S}_x}$ containing all the records for all the sink nodes. This approach would allow further optimizations.

[2]With reference to Fig. 2, each node is identified inside the mesh network by its 2-byte compressed address, while internally storing the 4-byte extended version of the address

[3]As in fact RouMBLE allows to organize the candidate BLE-based mesh network as a multi-sink tree-oriented topology, it may have sense to talk in terms of parent and child nodes, as well as predecessor and successor nodes.

Algorithm 1 Network topology construction procedure.

if $\mathcal{P}_{\text{TYPE}} = \mathcal{P}_{\text{RMS}}$ **then**
  if $\mathbb{P}$ contains $\mathcal{C}_{\text{BOM}}$ **then**
    if $\mathcal{A}_{\mathcal{S}_x} = \mathcal{A}_{\mathcal{G}_z}$ **then**
      discard BOM packet
    else
      if $\left[\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)} \right]_{\text{NUMRECORDS}} > R_{\text{MRFS}}$ **then**
        discard BOM packet
      else
        update routing table $\mathbb{R}$
        generate new BOM packet
        discard old BOM packet
  else
    continue
else
  continue

TABLE III: Example of a routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ stored inside a generic mesh node $\mathcal{S}_x$.

Ranking ($\mathcal{R}$)	**Sender** ($\mathcal{A}_{\mathcal{S}}$)	**Hops [num]**	**Received BOMs**	
			[num]	**[%]**
1	$\mathcal{A}_{\mathcal{S}_a}$	$d_{\mathcal{S}_a}$	$n_{\mathcal{S}_a}$	$n_{\mathcal{S}_a}/N^{(\mathcal{G}_z)}$
2	$\mathcal{A}_{\mathcal{S}_b}$	$d_{\mathcal{S}_b}$	$n_{\mathcal{S}_b}$	$n_{\mathcal{S}_b}/N^{(\mathcal{G}_z)}$
3	$\mathcal{A}_{\mathcal{S}_c}$	$d_{\mathcal{S}_c}$	$n_{\mathcal{S}_c}$	$n_{\mathcal{S}_c}/N^{(\mathcal{G}_z)}$

B. Routing Table Update Operations

Each time a generic mesh node $\mathcal{S}_x$ receives a valid BOM packet from the sink node $\mathcal{A}_{\mathcal{G}_z}$ and does not have an already-existing routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$, it will create a new routing table with the structure shown in Table III. If, instead, the routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ already exists, but *there is not* an entry for the current predecessor node $\mathcal{S}_{\text{PRED}}$, then a new entry is added into $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ with the following information: $\mathcal{A}_{\mathcal{S}_{\text{PRED}}}$ as sender address; $d_{\mathcal{S}_{\text{PRED}}}$ as the distance (i.e., depth) from the sink node $\mathcal{G}_z$; the value 1 as number of received BOMs (denoted, in general, as $n_{\mathcal{S}_{\text{PRED}}}$). Then, the percentage of received BOMs (over the total number of BOMs received for the sink node $\mathcal{G}_z$, denoted as $N^{(\mathcal{G}_z)}$), is calculated. Otherwise, if $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ is already existing and *there is* an already-existing entry for $\mathcal{S}_{\text{PRED}}$, then a comparison between $d_{\mathcal{S}_{\text{PRED}}}$ (contained in $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$) and d (contained in the received BOM) is carried out:

- if $d_{\mathcal{S}_{\text{PRED}}} < d$, then no modifications are needed, as $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$ contains ranking information more effective (smaller distance from the sink) than that carried by the BOM packet;
- if $d_{\mathcal{S}_{\text{PRED}}} \geq d$, then the receiving node $\mathcal{S}_x$ needs to (i) update $d_{\mathcal{S}_{\text{PRED}}}$ to d, (ii) increment by 1 the number of received BOMs $n_{\mathcal{S}_{\text{PRED}}}$, and (iii) re-calculate the percentage of received BOMs as $n_{\mathcal{S}_{\text{PRED}}}/N^{(\mathcal{G}_z)}$.

In this way, only significant BOM packets are considered to keep all the routing tables $\mathbb{R}$ of a node updated during its lifetime.

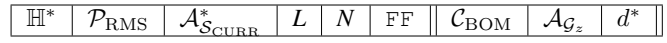

Fig. 7: Structure of a BOM packet with updated fields (defined based on an old BOM received by a predecessor node in the BLE-based mesh network).

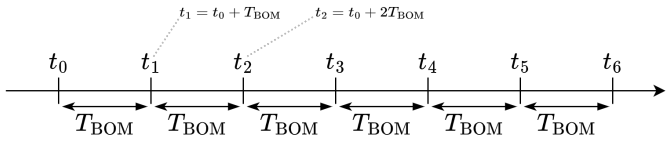

Fig. 8: Time instants considered in the routing tables construction example.

C. BOM Packets Forwarding With Updated Fields

In order to (i) keep as simple as possible the structure of the data packets defined in RouMBLE and (ii) avoid modifications in the RMS packets each time an information should be forwarded in the mesh network (e.g., BOM packets for topology construction), a new BOM packet, containing the identity information of the current node $\mathcal{S}_x$ (which, in turn, will appear as $\mathcal{S}_{\text{PRED}}$ for its successor nodes), has to be generated.

Assuming that $\mathcal{S}_x$ is the current node which has to continue the topology construction, a new BOM prepared by $\mathcal{S}_x$ to inform of its existence the overall mesh network has the structure shown in Fig. 7. The fields of this BOM can be described as follows.

- $\mathbb{H}^*$ is the updated header of the RMS packet (as detailed in Fig. 4). In detail, the hop counter h will be reduced by 1, with respect to that contained in the previously received BOM packet, while $\mathbb{U}$ will contain a new message identifier for the new BOM packet.
- $\mathcal{A}^*_{\mathcal{S}_{\text{CURR}}}$ contains the address of the current node emitting the new BOM packet, so that $\mathcal{A}^*_{\mathcal{S}_{\text{CURR}}} = \mathcal{A}_{\mathcal{S}_x}$.
- d^* is set to the depth contained in the old BOM packet increased by 1, in order to inform its successor nodes about the updated distance from the originator sink node $\mathcal{G}_z$.

The remaining fields in the BOM packet have the same meanings of those detailed in Subsection IV-A, as the address of the sink node $\mathcal{G}_z$ (i.e., $\mathcal{A}_{\mathcal{G}_z}$) is not modified, being the originator sink node which initiated the topology construction.

D. Routing Tables Construction Example

According to the operational steps shown in Algorithm 1 and described in Subsection IV-B and Subsection IV-C, in the following a *step-by-step* topology construction procedure is described. In detail, the time instants that will be considered refer to the timeline shown in Fig. 8, while the routing tables refer to the network shown in Fig. 9, in which lines connecting the different nodes correspond to physical links enabled by local node coverage. As an example, the sink node $\mathcal{G}_k$ is directly connected to nodes $\mathcal{S}_1$, $\mathcal{S}_6$ and $\mathcal{S}_7$, while $\mathcal{S}_2$ is directly connected to both $\mathcal{S}_1$ and $\mathcal{S}_6$.

We now characterize the time evolution. At instant t_0, the sink node $\mathcal{G}_k$ emits the BOM packet shown in Fig. 10, in which

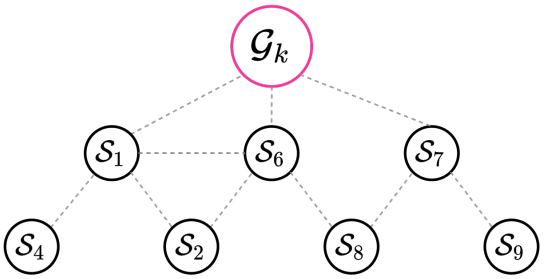

Fig. 9: Physical (connectivity-based) topology.

3	$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{G}_k}$	L	N	FF	$\mathcal{C}_{\text{BOM}}$	$\mathcal{A}_{\mathcal{G}_k}$	1

Fig. 10: BOM packet built by $\mathcal{G}_k$ at instant t_0.

2	$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_1}$	L	N	FF	$\mathcal{C}_{\text{BOM}}$	$\mathcal{A}_{\mathcal{G}_k}$	2

Fig. 11: BOM packet built by $\mathcal{S}_1$ at instant t_1.

2	$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_6}$	L	N	FF	$\mathcal{C}_{\text{BOM}}$	$\mathcal{A}_{\mathcal{G}_k}$	2

Fig. 12: BOM packet built by $\mathcal{S}_6$ at instant t_1.

TABLE IV: Routing table $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$ at T_{BOM0}, instant t_0.

Ranking ($\mathcal{R}$)	**Sender** ($\mathcal{A}_{\mathcal{S}}$)	**Hops [num]**	**Received BOMs**	
			[num]	**[%]**
1	$\mathcal{A}_{\mathcal{S}_1}$	2	1	$1/2 = 50\%$
2	$\mathcal{A}_{\mathcal{S}_6}$	2	1	$1/2 = 50\%$

2	$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_6}$	L	N	FF	$\mathcal{C}_{\text{BOM}}$	$\mathcal{A}_{\mathcal{G}_k}$	3

Fig. 13: BOM packet built by $\mathcal{S}_6$ at instant t_1.

the hop counter $h = h_{\text{MHN}} = 3$, while the depth d (inside the payload $\mathbb{P}$) is equal to 1 (as the direct children will be at distance 1 from the originator sink $\mathcal{G}_k$). This BOM packet is then received by nodes $\mathcal{S}_1$, $\mathcal{S}_6$ and $\mathcal{S}_7$, which, in turn, check it and add an entry in their routing tables $\mathbb{R}_{\mathcal{S}_1}^{(\mathcal{G}_k)}$, $\mathbb{R}_{\mathcal{S}_6}^{(\mathcal{G}_k)}$ and $\mathbb{R}_{\mathcal{S}_7}^{(\mathcal{G}_k)}$, respectively (all the entries refer to the originator sink node $\mathcal{G}_k$).

At instant $t_1 = t_0 + T_{\text{BOM}}$, $\mathcal{S}_1$, $\mathcal{S}_6$ and $\mathcal{S}_7$ are at distance 2 from the sink node $\mathcal{G}_k$. At this point: (i) $\mathcal{S}_1$ emits its new BOM packet (shown in Fig. 11) that will be received by $\mathcal{G}_k$, $\mathcal{S}_6$, $\mathcal{S}_2$ and $\mathcal{S}_4$; (ii) $\mathcal{S}_6$ sends its new BOM packet (shown in Fig. 12) that will be received by $\mathcal{G}_k$, $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_8$; (iii) $\mathcal{S}_7$ sends its new BOM packet (not shown here for simplicity) that will be received by $\mathcal{G}_k$, $\mathcal{S}_8$ and $\mathcal{S}_9$. Node $\mathcal{S}_2$ then receives two BOM packets ($N_{\mathcal{S}_2}^{(\mathcal{G}_k)} = 2$) and needs to update its routing table related to $\mathcal{G}_k$ (denoted as $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$) as shown in Table IV.

At instant $t_2 = t_0 + 2T_{\text{BOM}}$, $\mathcal{S}_2$ receives another BOM packet (shown in Fig. 13) from $\mathcal{S}_6$, which, instead, is not the same BOM packet received at instant t_1, since, in this case, $\mathcal{S}_6$ is at distance 3 from the sink node $\mathcal{G}_k$ ($\mathcal{G}_k \leftarrow \mathcal{S}_1 \leftarrow \mathcal{S}_6$). Then, $\mathcal{S}_2$ checks if it needs to update its routing table $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$: as the depth d of the BOM from $\mathcal{S}_6$ is higher than that contained in $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$, no further updates are needed.

TABLE V: Evolution of the routing table $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$ over different time slots.

	Ranking ($\mathcal{R}$)	**Sender** ($\mathcal{A}_{\mathcal{S}}$)	**Hops [num]**	**Received BOMs**	
				[num]	**[%]**
t_1	1	$\mathcal{A}_{\mathcal{S}_1}$	2	2	$2/4 = 50\%$
	2	$\mathcal{A}_{\mathcal{S}_6}$	2	2	$2/4 = 50\%$
	...				
t_2	1	$\mathcal{A}_{\mathcal{S}_1}$	2	3	$3/6 = 50\%$
	2	$\mathcal{A}_{\mathcal{S}_6}$	2	3	$3/6 = 50\%$
	...				
t_3	1	$\mathcal{A}_{\mathcal{S}_1}$	2	4	$4/8 = 50\%$
	2	$\mathcal{A}_{\mathcal{S}_6}$	2	4	$4/8 = 50\%$
	...				
t_4	2	$\mathcal{A}_{\mathcal{S}_1}$	2	4	$4/9 = 44\%$
	1	$\mathcal{A}_{\mathcal{S}_6}$	2	5	$5/9 = 56\%$
	...				
t_5	2	$\mathcal{A}_{\mathcal{S}_1}$	2	4	$4/10 = 40\%$
	1	$\mathcal{A}_{\mathcal{S}_6}$	2	6	$6/10 = 60\%$
	...				
t_6	2	$\mathcal{A}_{\mathcal{S}_1}$	2	4	$4/11 = 36\%$
	1	$\mathcal{A}_{\mathcal{S}_6}$	2	7	$7/11 = 64\%$

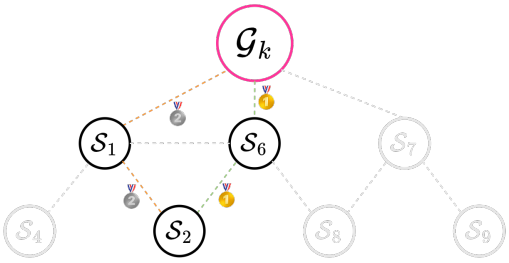

Fig. 14: Routing paths from $\mathcal{S}_2$ toward $\mathcal{G}_k$ at instant t_6 (based on Table V).

Then, focusing (as an example) on the node $\mathcal{S}_2$, its routing table will change as shown in Table V, assuming that, at instant $t_4 = t_0 + 4T_{\text{BOM}}$, the node $\mathcal{S}_1$ starts failing, sending no BOM to its neighbors, so that, at that point, a ranking update in $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$ may be needed. The corresponding routing paths, with reference to instant t_6, are shown in Fig. 14.

In this way, it is possible to classify different routes toward a specific sink node, as well as to maintain different routing paths toward all the sink nodes that started (at least once) a topology construction procedure. For the sake of completeness, with reference to Fig. 9, an example of all the possible routing paths among the nodes (with no BOM flooding interruption due to the hop counter h becoming equal to 0) is shown in Appendix A. Moreover, the decrease of the hop counter h in the BOM packet's header $\mathbb{H}$ insures that, after h_{MHN} attempts, these BOM packets stop. Finally, the strategy of using broadcast packets is not exploited in the overall mesh network, but only between neighboring nodes.

V. On-Field Data Collection

A. Data Collection - Request (Downlink)

Data collection is initiated by a sink node $\mathcal{G}_z$ interested in collecting data from the nodes composing the mesh network, thus starting an asynchronous "harvesting"—this could be a proactive action generated by the sink node $\mathcal{G}_z$ itself or a

$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$	L	N	FF	$\mathcal{C}_{\text{GET}}$	k	$\mathcal{A}_{\mathcal{T}}$	$\mathcal{A}_{\mathcal{G}_z}$	c	w_y

Fig. 15: Structure of a GET packet used for data collection from the BLE mesh nodes.

$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$	L	N	$\mathcal{A}_{\text{NH}}$	$\mathcal{C}_{\text{SEN}}$	k	$\mathcal{A}_{\text{ORIG}}$	$\mathcal{A}_{\mathcal{G}_z}$	T	v

Fig. 16: Structure of a SEN packet used by nodes to send information inside the BLE-based mesh network itself.

request by an external smart device. Assuming that a tree topology rooted at the sink node $\mathcal{G}_z$ itself has been built as detailed in Section IV (with routing tables $\mathbb{R}^{(\mathcal{G}_z)}$ at the mesh nodes), then $\mathcal{G}_z$ emits a specific data collection request to all nodes in the network. In detail, this is a particular RMS packet, denoted as GET packet and shown in Fig. 15, containing the following information (in various fields) useful to characterize the request.

- $\mathcal{C}_{\text{GET}}$ marks the current message as a GET packet.
- k identifies the type of information that the originator sink node $\mathcal{G}_z$ is interested on. It is expressed in binary encoding, thus allowing $2^8 = 256$ different data values.
- $\mathcal{A}_{\mathcal{T}}$ contains the address of the node(s) that should react to the current GET request by transmitting to the originator sink node $\mathcal{G}_z$ their k class-related information. As for similar fields, $\mathcal{A}_{\mathcal{T}}$ is represented in 2-byte binary encoding, as shown in Fig. 2.
- $\mathcal{A}_{\mathcal{G}_z}$ contains the address of the sink node $\mathcal{G}_z$ originating the GET data collection request and is represented in 2-byte binary encoding, as shown in Fig. 2.
- c contains an incremental index useful for packet marking and endless repetition avoidance.
- w_y represents an index identifying the group of nodes that have to reply with their k class-related data (if present), as further detailed in the following.

According to the GET packet's format shown in Fig. 15, RouMBLE provides two different approaches (in terms of data collection), leaving the specific sink node emitting the GET message to (i) send a request to all nodes (in broadcast, to collect data from all of them), (ii) target only a specific node, or (iii) target a specific group of nodes (through a group index w_y). In detail, RouMBLE performs as follows.

Should the originator sink node $\mathcal{G}_z$ be interested in targeting *all nodes* composing the mesh network, then it should emit a generalized GET request with $\mathcal{A}_{\mathcal{T}}$ = FF—as discussed in Fig. 2, this can be considered as a particular broadcast address, as valid unicast addresses are in the range 0000-79FF.

Should the sink node $\mathcal{G}_z$ be interested in targeting *only a specific node*, then it should send a GET request enumerating $\mathcal{A}_{\mathcal{T}}$ with the 2-byte address representation of the interested node (e.g., with reference to Fig. 2, the address 1AB3).

Should the sink node $\mathcal{G}_z$ be interested in data pertaining only a *specific sub-set of nodes*, then it should emit a generalized GET request with $\mathcal{A}_{\mathcal{T}}$ = FF and specify a proper integer group index w_y interpreted as follows:

$$\left\{\mathcal{A}_{\mathcal{S}_x}\right\}_{\text{DEC}} \in \left\{w_y\Omega_{\text{BS}}, w_y\Omega_{\text{BS}}+1, \ldots, (w_y+1)\Omega_{\text{BS}}-1\right\} \quad (1)$$

where Ω_{BS} it the size of the sub-set and $\{\cdot\}_{\text{DEC}}$ corresponds to a HEX-to-DEC base conversion. In detail, a node $\mathcal{S}_x$ belongs to a certain group w_y if its address $\mathcal{A}_{\mathcal{S}_x}$ (converted in the decimal base) is included in the values' set defined in Eq. (1).

As a numerical example, considering $\Omega_{\text{BS}} = 40$ and $w_y = 7$, a group-targeted GET request will be handled as follows:

- a node $\mathcal{S}_x$ with address $\mathcal{A}_{\mathcal{S}_x}$ = 0127 will reply to a sink node $\mathcal{G}_z$ originating the GET packet, since $\{0127\}_{\text{DEC}} = 295 \in \{280, \ldots, 319\}$;
- a node $\mathcal{S}_x$ with address $\mathcal{A}_{\mathcal{S}_x}$ = 0144 will not reply to a sink node $\mathcal{G}_z$ originating the GET packet, since $\{0144\}_{\text{DEC}} = 324 \notin \{280, \ldots, 319\}$.

Hence, when a GET packet is received, each node checks it and, in the presence of a positive outcome, the node starts preparing a response message (discussed in Subsection V-B) for each information to be returned to the sink node $\mathcal{G}_z$ originating the GET request. Finally, in the case of a broadcast GET packet (e.g., with $\mathcal{A}_{\mathcal{T}}$ = FF), each node receiving the GET request has to support the propagation of this GET message in the mesh network: it thus generates a new GET packet with an updated $\mathcal{A}^*_{\mathcal{S}_{\text{CURR}}}$ equal to its own address (as detailed for BOM packets in Subsection IV-C).

B. Data Collection - Response (Uplink)

When a mesh node needs to send data to a sink node (e.g., $\mathcal{G}_z$)—in both the cases that the sink node is "polling" the network or the node itself is proactively sending data because of its operational characteristics—it will exploit the existence of the routing tables created as discussed in Section IV. In particular, each node should use a specific message, denoted as SEN packet and shown in Fig. 16, able to flow in the mesh network and targeting the corresponding sink node $\mathcal{G}_z$. The fields of the SEN packet can be denoted as follows.

- $\mathcal{A}_{\text{NH}}$ contains the address of the next-hop node toward the sink node $\mathcal{G}_z$ (retrieved by the specific routing table $\mathbb{R}^{(\mathcal{G}_z)}$ at the node).
- $\mathcal{C}_{\text{SEN}}$ marks the current message as a SEN packet.
- k identifies the type of payload (requested by the $\mathcal{G}_z$through a GET request, or proactively decided by the node itself) carried by the SEN message. As detailed in Subsection V-A, k is expressed in binary encoding, thus allowing $2^8 = 256$ different data values.
- $\mathcal{A}_{\text{ORIG}}$ contains the address of the mesh node that originated the response transmission (through the SEN packet). As similar address fields, $\mathcal{A}_{\text{ORIG}}$ is represented in 2-byte binary encoding, as shown in Fig. 2.
- $\mathcal{A}_{\mathcal{G}_z}$ contains the address of the sink node $\mathcal{G}_z$ which the information payload v should be delivered to. As $\mathcal{A}_{\text{ORIG}}$, $\mathcal{A}_{\mathcal{G}_z}$ is represented in 2-byte binary encoding, as shown in Fig. 2.
- T contains the timestamp reference of the collection instant by the mesh node $\mathcal{A}_{\text{ORIG}}$ and is expressed with binary encoding.
- v corresponds to the information payload to be delivered to the sink node $\mathcal{G}_z$.

Algorithm 2 SEN packets handling by a BLE mesh node $\mathcal{S}_x$.

```
if P_TYPE = P_RMS then
    if P contains C_SEN then
        if A_NH = A_Sx then
            if A_Gz = A_Sx then                ▷ [sink node G_z]
                handle received data
            else                               ▷ [Mesh node S_x]
                retrieve new next-hop node from R_Sx^(Gz)
                generate new SEN packet
                discard old SEN packet
        else
            discard SEN packet
    else
        continue
else
    continue
```

Moreover, as highlighted in Fig. 16, since a SEN packet should strictly address a specific sink node (i.e., the one originating the previous GET request, e.g., $\mathcal{G}_z$, as well as a specific destination sink node), the fields $\mathcal{D}_{\mathrm{TYPE}}$ and $\mathcal{A}_{\mathrm{TYPE}}$ should comply with a unicast transmission (e.g., with $\mathcal{D}_{\mathrm{TYPE}} = L$ and $\mathcal{A}_{\mathrm{TYPE}} = N$).

Since RouMBLE exploits data transmission over BLE advertisement channels, each SEN packet is received by all neighbors (i.e., within the transmission range) of each node. To this end, each neighbor node receiving the SEN packet is aware (by construction) of being either a sink node or a normal node. Because of this, each node will perform different checks, as shown in Algorithm 2, to verify if the identity of the next-hop node $\mathcal{A}_{\mathrm{NH}}$ is that of the sink node $\mathcal{A}_{\mathcal{G}_z}$. Should this be the case, this would mean that the SEN packet has reached its final destination $\mathcal{G}_z$. At the opposite (i.e., if $\mathcal{A}_{\mathrm{NH}} \neq \mathcal{G}_z$), then the node with address $\mathcal{A}_{\mathrm{NH}}$ (say, for example, $\mathcal{S}_x$) should perform the following steps:

1) it retrieves the next-hop node for the referred sink node $\mathcal{G}_z$ from its internal routing table $\mathbb{R}_{\mathcal{S}_x}^{(\mathcal{G}_z)}$;
2) it generates a new SEN packet, updating the address of the current sending node $\mathcal{A}^*_{\mathcal{S}_{\mathrm{CURR}}}$ and that of the next-hop node $\mathcal{A}^*_{\mathrm{NH}}$ toward the sink node $\mathcal{G}_z$, and transmits it;
3) it discards the old SEN packet, not re-transmitting it to avoid congesting the mesh network.

C. Data Collection: an Illustrative Example

In order to better clarify the data collection procedure detailed in Subsection V-A and Subsection V-B, in the following an illustrative example, based on the network shown in Fig. 9, is presented.

Given that every node in the network in Fig. 9 has its own routing table $\mathbb{R}^{(\mathcal{G}_z)}$ built as shown in Subsection IV-D, assume that, at a certain time instant, a GET request is sent by the sink node $\mathcal{G}_z$ and node $\mathcal{S}_2$ is the destination node which needs to reply to $\mathcal{G}_z$. Then:

1) $\mathcal{S}_2$ executes a lookup operation in its routing table $\mathbb{R}_{\mathcal{S}_2}^{(\mathcal{G}_k)}$, looking for the best candidate next-hop node toward the sink node $\mathcal{G}_z$, and selects $\mathcal{S}_6$ (with its address $\mathcal{A}_{\mathcal{S}_6}$) as next-hop node;
2) $\mathcal{S}_2$ creates a SEN packet (with the structure shown in Fig. 17), setting the SEN originator address $\mathcal{A}_{\mathrm{ORIG}}$ and the RMS originator address $\mathcal{A}_{\mathcal{S}_{\mathrm{CURR}}}$ with its own address $\mathcal{A}_{\mathcal{S}_2}$, and the next-hop node address with the address of $\mathcal{S}_6$ (namely, $\mathcal{A}_{\mathcal{S}_6}$).

$\mathcal{P}_{\mathrm{RMS}}$	$\mathcal{A}_{\mathcal{S}_2}$	L	N	$\mathcal{A}_{\mathcal{S}_6}$	$\mathcal{C}_{\mathrm{SEN}}$	k	$\mathcal{A}_{\mathcal{S}_2}$	$\mathcal{A}_{\mathcal{G}_k}$	T	v

Fig. 17: SEN packet sent from node $\mathcal{S}_2$.

$\mathcal{P}_{\mathrm{RMS}}$	$\mathcal{A}_{\mathcal{S}_6}$	L	N	$\mathcal{A}_{\mathcal{G}_k}$	$\mathcal{C}_{\mathrm{SEN}}$	k	$\mathcal{A}_{\mathcal{S}_2}$	$\mathcal{A}_{\mathcal{G}_k}$	T	v

Fig. 18: SEN packet sent from node $\mathcal{S}_6$.

When $\mathcal{S}_6$ receives the SEN packet sent from $\mathcal{S}_2$, it performs the checks detailed in Algorithm 2 and, once positively concluded, it creates a new SEN packet as shown in Fig. 18. In this new packet, the SEN's payload is left unmodified, while inserting as RMS originator address $\mathcal{A}_{\mathcal{S}_{\mathrm{CURR}}}$ its own address $\mathcal{A}_{\mathcal{S}_6}$and as next-hop node address the address of the sink node $\mathcal{G}_z$ (namely, $\mathcal{A}_{\mathcal{G}_z}$), since $\mathcal{S}_6$ is directly connected with $\mathcal{G}_z$.

VI. Use Cases

In order to further highlight the possibilities given by the proposed routing protocol, we now outline potentially applicable use cases.

A. Topology Mapping at Sink Node Side

In RouMBLE, the use of SEN packets would allow to map the mesh network topology (built through BOM packets) directly at the sink node(s). To do this, it would be sufficient to reserve a specific value for the information type identifier k (denoted as k_{NTM}): when a GET request is received from the sink node $\mathcal{G}_z$, each node replies preparing a SEN packet to $\mathcal{G}_z$ with the structure shown in Fig. 19, where the fields can be described as follows.

- $\mathcal{A}_{\mathrm{NH}}$ contains the address of the next-hop node toward the sink node $\mathcal{G}_z$ (retrieved by its internal routing table $\mathbb{R}^{(\mathcal{G}_z)}$).
- k_{NTM} represents the information type identifier reserved for the network topology mapping.
- $\mathcal{A}_{\mathrm{ORIG}}$ contains the address of the node that generates the response SEN packet. As for similar address fields, $\mathcal{A}_{\mathrm{ORIG}}$ is expressed in 2-byte binary encoding, as shown in Fig. 2.
- ℓ is retrieved from the routing table $\mathbb{R}^{(\mathcal{G}_z)}$ of the node and corresponds to the depth (in terms of number of hops) between the node itself and the sink node $\mathcal{G}_z$ which the SEN packet should be sent to.
- x contains the address of the next-hop node toward the sink node $\mathcal{G}_z$ (retrieved by its internal routing table $\mathbb{R}^{(\mathcal{G}_z)}$).

Through the SEN packet shown in Fig. 19, it is possible to let the sink node $\mathcal{G}_z$ be aware of the next-hop entity of each node participating to the mesh network itself, with a "backup" information represented by the depth ℓ contained in

$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$	$\mathcal{D}_{\text{TYPE}}$	N	$\mathcal{A}_{\text{NH}}$	$\mathcal{C}_{\text{SEN}}$	k_{NTM}	$\mathcal{A}_{\text{ORIG}}$	$\mathcal{A}_{\mathcal{G}_z}$	ℓ	x

Fig. 19: SEN packet built by mesh nodes for network topology mapping purposes.

TABLE VI: Sensing table Π_{SENSE} based on RSSI information and stored inside a BLE node.

MAC Address	RSSI [dBm]	Packets Counter
8A:FD:7A:87:76:11	−87	6
4A:5B:6D:3C:2D:1E	−81	45
...	...	...

each SEN packet. Thus, once the SEN packet is received by the sink node, the sink node will be able to infer the current tree topology (rooted at itself) for further purposes (e.g., graphical representation)

Finally, it should be noted that, even though the original SEN packet sent by each node contains the address of its next-hop node twice (in the fields x and $\mathcal{A}_{\text{NH}}$), the value of the field x will not be modified during the intermediate operations that will be performed on the intermediate SEN packets, while $\mathcal{A}_{\text{NH}}$ will change at each SEN message's redefinition.

B. Sensing of BLE Devices in the Neighborhood

In addition to routing functionalities, RouMBLE may be applied, as anticipated in Section I, to sense BLE devices through BLE signal parameters (e.g., Received Signal Strength Indicator, RSSI).

1) Sensing Table Definition: Assume that, at fixed times, each BLE device emits an ADVertisement (ADV) packet which contains the originator nodes's MAC address, RSSI value, and additional information. Each time a BLE ADV packet is detected by a BLE node, an admission decision is made, based on the RSSI of the packet itself and on its information content. If this RSSI-based check is successfully completed, then the contained information is forwarded to application functionalities (internal to the node itself) in charge of processing (and possibly storing) them for further analysis. As an example, information on discovered BLE devices can be stored in a structure such as the one shown in Table VI, containing (i) the MAC address of the detected BLE device (obtained at lower stack layers), (ii) the number of BLE ADV packets detected, and (iii) the best RSSI value (dimension: [dBm]) among all these ADV packets.

2) Heatmap Creation: Upon construction of a BLE sensing table, as discussed in Subsection VI-B1, one can derive a "heatmap" of the environment where BLE devices are still active (and could move in, if needed). In detail, such a heatmap derives from the sensing table Π_{SENSE} shown in Table VI, with a supplementary tracking information. More precisely, this information is a binary information on the proximity of the "sensed" BLE device, as shown in Table VII (Near: Yes or No). In detail, each time an ADV packet is detected by a listening BLE device, the following steps take place:

- if an entry for the detected MAC address is not present in Π_{PROX}, then a new record is added;
- if the MAC address already exists in Π_{PROX}, then (i) the packets counter is increased by 1, (ii) a check on the RSSI strength (with respect to that contained in the heatmap table Π_{PROX}) is performed multiple times, and (iii) if the majority of the sensed RSSIs is above a certain threshold γ_{RSSI}, then the detected MAC address will be marked as near (Near = Y); otherwise, it will be marked as far (Near = N) inside the heatmap table Π_{PROX}.

TABLE VII: Enhanced proximity table Π_{PROX} used for heatmap generation and based on RSSI information.

MAC Address	RSSI [dBm]	Packets Counter	Near
8A:FD:7A:87:76:11	−87	6	N
4A:5B:6D:3C:2D:1E	−81	45	Y
...	...	...	...

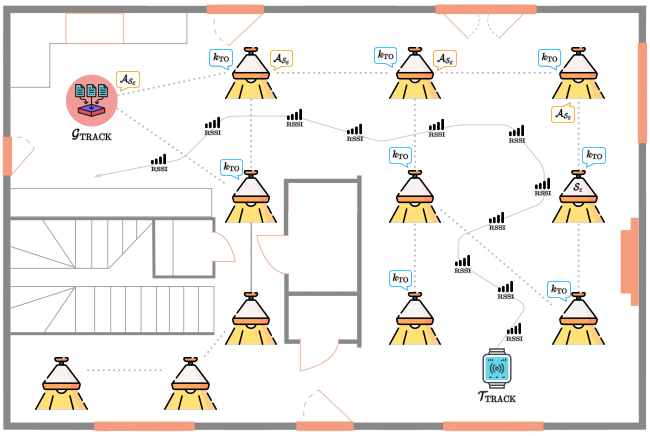

Fig. 20: Example of the tracking of a device $\mathcal{T}_{\text{TRACK}}$ inside an environment.

$\mathcal{P}_{\text{RMS}}$	$\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$	$\mathcal{D}_{\text{TYPE}}$	N	$\mathcal{A}_{\text{NH}}$	$\mathcal{C}_{\text{SEN}}$	k_{TO}	$\mathcal{A}_{\text{ORIG}}$	$\mathcal{A}_{\mathcal{G}_{\text{TRACK}}}$	T	v

Fig. 21: SEN packet defined for the tracking of a device $\mathcal{T}_{\text{TRACK}}$.

Moreover, in order to keep the sensed information updated, the information inside the heatmap Π_{PROX} may be periodically refreshed (e.g., filtering the information with sliding windows).

3) Device Tracking: Finally, another use case that could be handled through the adoption of RouMBLE is tracking of a mobile device $\mathcal{T}_{\text{TRACK}}$ (identified by its MAC address $\mathcal{M}_{\mathcal{T}_{\text{TRACK}}}$).

In detail, once an ADV packet positively satisfies both the *first* admission check for Π_{SENSE} (discussed in Subsection VI-B1) and the *second* proximity check for Π_{PROX} (discussed in Subsection VI-B2), then its RSSI is compared with a *third* (more stringent) RSSI threshold (denoted as $\gamma_{\text{RSSI}-\text{TO}}$). Should the sensed RSSI be above $\gamma_{\text{RSSI}-\text{TO}}$, then, as shown in Fig. 20, a new SEN packet should be prepared to keep the specific sink node $\mathcal{G}_{\text{TRACK}}$ updated on $\mathcal{T}_{\text{TRACK}}$. Its structure is shown in Fig. 21 and its fields can be described as follows.

- $\mathcal{A}_{\mathcal{S}_{\text{CURR}}}$ contains the address of the BLE node detecting the specific MAC address $\mathcal{M}_{\mathcal{T}_{\text{TRACK}}}$ and originating the current SEN packet.
- $\mathcal{A}_{\text{NH}}$ contains the address of the next-hop node toward the sink node $\mathcal{G}_{\text{TRACK}}$ interested in tracking $\mathcal{M}_{\mathcal{T}_{\text{TRACK}}}$ inside the mesh network itself.

- k_{TO} represents the service type identifier defined and reserved for the tracking functionality.
- $\mathcal{A}_{\mathrm{ORIG}}$ contains the address of the originator node that detected the specific MAC $\mathcal{M}_{\mathcal{T}_{\mathrm{TRACK}}}$ and, as for similar address fields, is expressed in 2-byte binary encoding (as shown in Fig. 2).
- $\mathcal{A}_{\mathcal{G}_{\mathrm{TRACK}}}$ contains the address of the sink node interested in receiving the tracking information and is expressed in 2-byte binary encoding (as shown in Fig. 2).
- T contains a time reference (e.g., a timestamp) related to the RSSI detection's time instant.
- v may contain additional data to be inserted inside the SEN packet toward the sink node $\mathcal{G}_{\mathrm{TRACK}}$.

When the sink node $\mathcal{G}_{\mathrm{TRACK}}$ receives SEN packets carrying the service type k_{TO}, it then keeps track of the nodes "sensing" the specific MAC address $\mathcal{M}_{\mathcal{T}_{\mathrm{TRACK}}}$. In this way, each time a SEN packet is received, $\mathcal{G}_{\mathrm{TRACK}}$ may process its information and react, as a consequence, with specific behaviors: as an example, the sink node might select a BLE node $\mathcal{S}_{\varepsilon}$ (identified by its address $\mathcal{A}_{\mathcal{S}_{\varepsilon}}$) among those that previously detected $\mathcal{G}_{\mathrm{TRACK}}$ and send it a new request to execute a specific task (similarly to a GET request).

Finally, an additional use case that is enabled by RouMBLE corresponds to an aggregate export (toward a requesting sink node) of the sensing table Π_{SENSE} maintained by each BLE node with an active sensing functionality. Moreover, defining and reserving a proper set of service types, it would be possible for a sink node to retrieve (with either unicast or broadcast communications) statistical information based on the content of Π_{SENSE}, such as, for example: (i) the sum of packets counters for each MAC address stored in Π_{SENSE} for each specific BLE node; (ii) the amount of MAC addresses present in Π_{SENSE} for each specific BLE node; and (iii) a mixed combination of these indicators.

Eventually, the format defined for the SEN packet can be used to retrieve the sensing table content from each node which is sensing the BLE devices in its neighborhood. In detail, each node will have to send to the specific sink node (interested on this kind of information) a SEN packet for each record in its sensing tables.

C. Node Failures and Network Attack Handling

Finally, we comment on how RouMBLE can handle two possible problematic situations, namely: (i) the failure of a mesh node and (ii) an attack to the mesh network itself.

1) Node Failure: Detecting and handling a node failure is enabled by the inner features of RouMBLE, in particular its compatibility with multiple sink nodes as well as the identification of multiple routes toward a specific sink node. Thanks to the underlying wireless nature of the BLE protocol and the flooding paradigm exploited by RouMBLE, each active node "hears" the network traffic flowing through the mesh network itself. This allows each node to rank the nodes according to their transmission activity ("talkative" or "silent"). Each time an active node (denoted as $\mathcal{A}_{\mathrm{ALIVE}}$) suspects a "too silent" node (denoted as $\mathcal{A}_{\mathrm{DEAD}}$) to be *dead*, $\mathcal{A}_{\mathrm{ALIVE}}$ could send a "warning" SEN packet to all its known—due to their presence in its routing tables—sink nodes, in order to warn them about this possibility. Consequently, each of these sink nodes will be able to estimate (on the basis of the amount of warning notifications received from the mesh nodes) the level of criticality brought by $\mathcal{A}_{\mathrm{DEAD}}$'s death. If the level of criticality is recognized as being *low*, then no action is taken by the sink node; otherwise, if the level of criticality is deemed high by a sink node, this sink node broadcasts a GET packet toward the mesh nodes denoted by a specific service type identifier (e.g., denoted as k_{DEAD}) and containing $\mathcal{A}_{\mathrm{DEAD}}$ as payload, in order to ask each mesh node to mark all their routing table entries containing $\mathcal{A}_{\mathrm{DEAD}}$ as next-hop as "not preferrable." In a limiting case, a sink node may request each mesh node to completely remove (and not only to mark it as "not preferrable") $\mathcal{A}_{\mathrm{DEAD}}$ from their routing tables in the event that $\mathcal{A}_{\mathrm{DEAD}}$ does not reply to a direct GET request sent from the sink node itself.

2) Network Attack: We focus on two relevant attacks, namely: (i) a malicious node trying to join the BLE mesh network; (ii) a jamming node trying to pollute the network with malicious traffic.

In order to counteract a new malicious node trying to join the mesh network, RouMBLE can be integrated with the following two additional functionalities involving both sink nodes and legitimate mesh nodes: (i) an admission check and (ii) the adoption of security group keys. Both these features can be managed by the sink nodes governing the mesh network, with the admission check based on pre-filled lookup tables, containing a direct mapping between MAC and node (BLE) addresses of each legitimate node, and group keys defined by sink nodes and distributed (for their consequent adoption) to all the legitimate nodes. The use of group keys requires to handle keys' refreshing (in the presence of ousted nodes), but this might rely on well-known approaches proposed in the literature. We remark that a pre-admission stage is already considered in the current version of RouMBLE to avoid addresses' mismatch and duplication.

If a malicious node (generically denoted as $\mathcal{A}_{\mathrm{MALICIOUS}}$) manages to join the network and want to start poisoning the mesh network with malicious traffic, RouMBLE can counteract it exploiting knowledge of the behaviour of the nodes—namely, *static* or *mobile*—and the ability to parse the payloads of the BLE packets defined by RouMBLE itself. With regard to the nodes' behaviour, given that a legitimate node (generically denoted as $\mathcal{A}_{\mathrm{LEGIT}}$) can estimate (as detailed in Subsection VI-B) how neighboring nodes are performing (at the RSSI level): *if* the scenario is *static* and $\mathcal{A}_{\mathrm{LEGIT}}$ senses that a (malicious) node (denoted as $\mathcal{A}_{\mathrm{MALICIOUS}}$) returns an "unusual" RSSI, then $\mathcal{A}_{\mathrm{LEGIT}}$ can mark $\mathcal{A}_{\mathrm{MALICIOUS}}$ as a *banned* mesh node inside its internal admission check stage, thus discarding (e.g., on the basis of the MAC address) each packet sent by $\mathcal{A}_{\mathrm{MALICIOUS}}$. As a consequence, should the sink nodes governing the mesh network be notified about this malicious entity, a new service type identifier (e.g., denoted as $k_{\mathrm{MALICIOUS}}$) can be defined and used to notify the nodes. Finally, exploiting the parsing ability of RouMBLE, if $\mathcal{A}_{\mathrm{LEGIT}}$ repeatedly detects exceptions when parsing the payloads received by a specific mesh node (namely,

$\mathcal{A}_{\text{MALICIOUS}}$), it can identify this behavior as representative of a malicious node attempting to pollute the mesh network with "unknown" information. In general, it would be useful to define a guarantee policy to minimize the probability of incorrectly classifying a node as malicious. In this case, similarly to the previous situation, $\mathcal{A}_{\text{LEGIT}}$ can mark $\mathcal{A}_{\text{MALICIOUS}}$ as a *banned* node and inform all its known sink nodes about this situation (exploiting the newly-defined $k_{\text{MALICIOUS}}$ service type identifier).

Finally, regardless of the specific threat, RouMBLE is attractive (in comparison with other routing protocols) since it allows to also localise the malicious node exploiting, for example, the operational mode detailed in Subsection VI-B3 and tracking $\mathcal{A}_{\text{MALICIOUS}}$.

VII. Experimental Performance Evaluation

In order to investigate the behaviour of RouMBLE in a real context, an experimental performance evaluation has been performed considering a BLE mesh network (shown in Fig. 22) composed by 1 sink node and 50 *on-field* nodes. In detail, both sink and *on-field* nodes are custom-made boards, each hosting a BMD-300 BLE module [58] (based on the Nordic Semiconductor's nRF52832 general-purpose SoC [59]) and a white LED being turned *on* and *off* based on the instructions defined in the nodes' FirmWare (FW). This testbed was organised at the headquarter of TCI Telecomunicazioni Italia s.r.l., Saronno, Italy.

The 50 mesh nodes were deployed in a 2-floor office building, with the sink node located in a position not allowing direct interaction with all the *on-field* nodes and, thus, requiring the use of mesh networking with RouMBLE. Moreover, as the operational mode chosen for the experimental evaluation is identical to that of the use case discussed in Subsection VI-A, in order to verify the correct implementation of RouMBLE, the on-board FW of each node has been coded so that its on-board LED turns *on* when a BOM packet is received and turns *off* upon reception of a GET request.

Various experimental data collection campaigns have been performed, trying to identify a reasonable number of BOMs to be sent from the sink node toward the nodes, in order to allow completion of their routing tables and maximise the amount of responses (i.e., SEN messages) received by the sink itself when it requests an action through the emission of a GET packet. As shown in Fig. 23 (where the error bars' limits identify worst and best occurrences among the overall evaluations), the experimental results suggest that the configuration with 20 BOM packets is the best candidate in such a scenario. As a general consideration, the obtained results are specific for the considered configuration, while even a smaller amount of BOMs might be considered as acceptable in a different context (e.g., this depends on the acceptability threshold that each scenario may present).

VIII. Conclusions

In this paper, we have proposed RouMBLE, a sink-oriented routing protocol for BLE-based mesh networks intended to address different classes of devices and build a sink-oriented

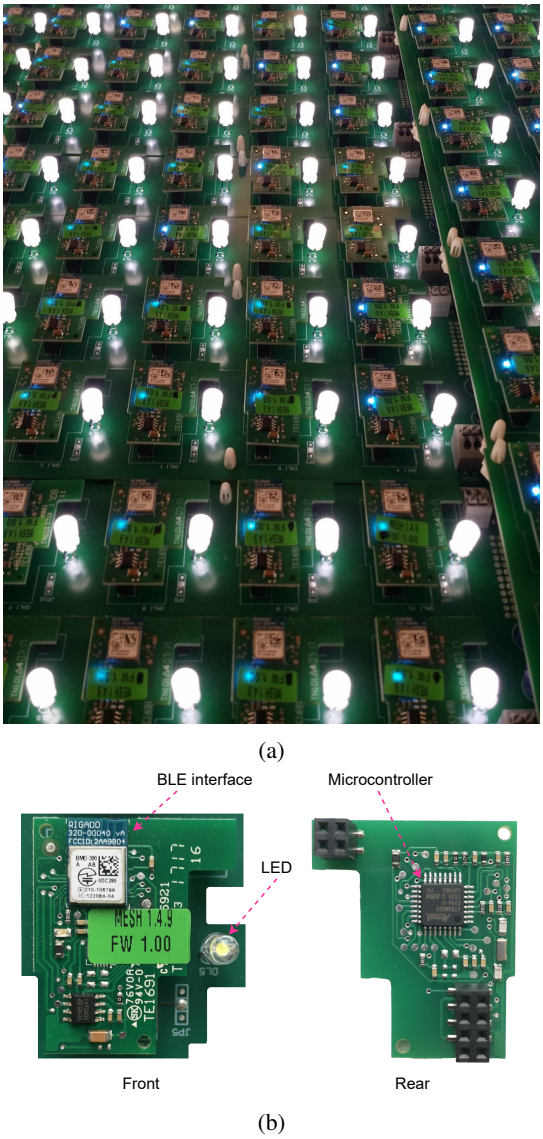

Fig. 22: (a) Experimental testbed composed by custom-made boards hosting a BMD-300 BLE module (based on the Nordic Semiconductor's nRF52832 SoC). (b) Front and rear view of an experimental board involved in the RouMBLE-based mesh network, and composed by the BMD-300 BLE module, an Atmel microcontroller hosting the FW functionalities, and a white LED used for visual feedback.

topology, thus allowing to perform various tasks. In detail, BLE mesh networks with RouMBLE can be seen as composed by cooperating heterogeneous devices which exploit a topological organisation based on the use of a "controlled" flooding mechanism at low layers. The proposed approach embodies the advantages of the flooding mechanism (e.g., simplicity and absence of connection establishment requirements) with those of routing-based schemes and, in the specific BLE implementation case, advertisement channel-based communications.

Further research activities of interest might include: (i) the integration of IPv6 support (through proper address compression, to be adopted by IoT constrained devices); (ii) comparisons, in the BLE case, with approaches based on the use of

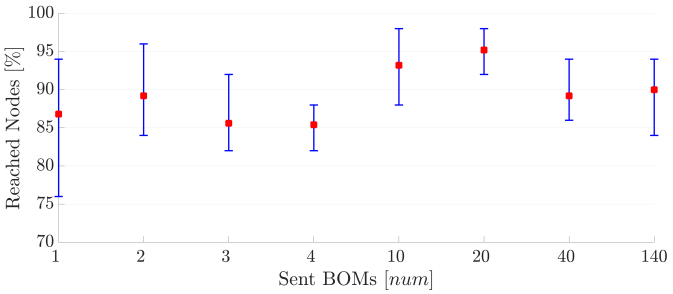

Fig. 23: Experimental performance results obtained in a BLE-based mesh network composed by 1 sink node and 50 *on-field* nodes.

data channels; (iii) the implementation of RouMBLE through microcontroller-oriented (e.g., MicroPython) and high-level (e.g., Python or Java) programming languages, to let the proposed routing protocol being deployed also on devices supporting these libraries (with a consequent performance comparison); and (iv) an experimental performance evaluation in heterogeneous scenarios (e.g., crowded events, such as concerts and music festivals) in order to analysis the Quality of Experience (QoE) perceived by end-users wearing, as an example, wearable nodes receiving commands from the control room. Finally, although RouMBLE has been presented for BLE networks, its principles could be applied to networks based on different wireless technologies.

ACKNOWLEDGMENTS

This work was supported by TCI Telecomunicazioni Italia s.r.l.. The work of L. Davoli and G. Ferrari has been partially funded by the European Union's Horizon 2020 research and innovation program ECSEL Joint Undertaking (JU) under grant agreement No. 876038, InSecTT project - "Intelligent Secure Trustable Things," and by the "Technologies, Algorithms, and Protocols for Use Cases of Industrial Networks" (TAP-IN) cascade call project, under the National Recovery and Resilience Plan (NRRP) of NextGenerationEU, Mission 4 Component 2 Investment 1.3, "RESearch and innovation on future Telecommunications systems and networks, to make Italy more smart" (RESTART) project (code PE00000001) Call n. 341 of 15.03.2022 of the Italian Ministry of University and Research (MUR), Spoke 5 "Industrial and Digital Transition Networks" (IN). The JU received support from the European Union's Horizon 2020 research and innovation programme and the nations involved in the mentioned projects. The work reflects only the authors' views; the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Internet Engineering Task Force (IETF). Accessed on Sept. 25, 2024. [Online]. Available: https://www.ietf.org/

[2] "Bluetooth Special Interest Group (SIG)," accessed on Sept. 25, 2024. [Online]. Available: https://www.bluetooth.com/

[3] J. Nieminen *et al.*, "IPv6 over BLUETOOTH(R) Low Energy," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 7668, October 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7668

[4] P. Di Marco *et al.*, "Coverage Analysis of Bluetooth Low Energy and IEEE 802.11ah for Office Scenario," in *IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Hong Kong, China, August 2015, pp. 2283–2287, doi:10.1109/PIMRC.2015.7343678.

[5] Z. Pei *et al.*, "Application-Oriented Wireless Sensor Network Communication Protocols and Hardware Platforms: A Survey," in *IEEE International Conference on Industrial Technology (ICIT)*, Chengdu, April 2008, pp. 1–6, doi:10.1109/ICIT.2008.4608532.

[6] A. Hernández-Solana *et al.*, "Bluetooth Mesh Analysis, Issues, and Challenges," *IEEE Access*, vol. 8, pp. 53 784–53 800, 2020, doi:10.1109/ACCESS.2020.2980795.

[7] S. M. Darroudi and C. Gomez, "Bluetooth Low Energy Mesh Networks: A Survey," *Sensors*, vol. 17, no. 7, 2017, doi:10.3390/s17071467.

[8] M. R. Ghori, T.-C. Wan, and G. C. Sodhy, "Bluetooth Low Energy Mesh Networks: Survey of Communication and Security Protocols," *Sensors*, vol. 20, no. 12, 2020, doi:10.3390/s20123590.

[9] S. Sirur *et al.*, "A Mesh Network for Mobile Devices using Bluetooth Low Energy," in *IEEE SENSORS*, Busan, Korea (South), November 2015, pp. 1–4, doi:10.1109/ICSENS.2015.7370451.

[10] S. S. Basu, M. Baert, and J. Hoebeke, "QoS Enabled Heterogeneous BLE Mesh Networks," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, 2021, doi:10.3390/jsan10020024.

[11] M. Spörk *et al.*, "BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices," in *ACM Conference on Embedded Network Sensor Systems (SenSys)*, Delft, Netherlands, 2017, pp. 1–14, doi:10.1145/3131672.3131687.

[12] Blue Light Link (BLL), "BLL System," accessed on Sept. 25, 2024. [Online]. Available: https://www.bluelightlink.com/bll-system/

[13] Freifunk, "Better Approach to Mobile Ad-hoc Networking (B.A.T.M.A.N.) Routing Protocol," accessed on Sept. 25, 2024. [Online]. Available: https://www.open-mesh.org/projects/open-mesh/wiki

[14] I. Akyildiz and X. Wang, "A Survey on Wireless Mesh Networks," *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23–S30, 2005, doi:10.1109/MCOM.2005.1509968.

[15] S. Waharte *et al.*, "Routing Protocols in Wireless Mesh Networks: Challenges and Design Considerations," *Multimedia Tools and Applications*, vol. 29, no. 3, pp. 285–303, June 2006, doi:10.1007/s11042-006-0012-8.

[16] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *Annual International Conference on Mobile Computing and Networking (MobiCom)*, Philadelphia, PA, USA, 2004, pp. 114–128, doi:10.1145/1023720.1023732.

[17] M. E. M. Campista *et al.*, "Routing Metrics and Protocols for Wireless Mesh Networks," *IEEE Network*, vol. 22, no. 1, pp. 6–12, 2008, doi:10.1109/MNET.2008.4435897.

[18] Bluetooth, "Mesh Networking," accessed on Sept. 25, 2024. [Online]. Available: https://www.bluetooth.com/learn-about-bluetooth/recent-enhancements/mesh/

[19] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 4919, August 2007. [Online]. Available: https://tools.ietf.org/rfc/rfc4919

[20] B. Luo *et al.*, "Neighbor Discovery for IPv6 over BLE Mesh Networks," *Applied Sciences*, vol. 10, no. 5, 2020, doi:10.3390/app10051844.

[21] S. M. Darroudi and C. Gomez, "Experimental Evaluation of 6BLEMesh: IPv6-Based BLE Mesh Networks," *Sensors*, vol. 20, no. 16, Aug 2020, doi:10.3390/s20164623.

[22] C. Gomez *et al.*, "From 6LoWPAN to 6Lo: Expanding the Universe of IPv6-Supported Technologies for the Internet of Things," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 148–155, 2017, doi:10.1109/MCOM.2017.1600534.

[23] T. Savolainen, J. Soininen, and B. Silverajan, "IPv6 Addressing Strategies for IoT," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3511–3519, 2013, doi:10.1109/JSEN.2013.2259691.

[24] A. Gogic *et al.*, "Performance Analysis of Bluetooth Low Energy Mesh Routing Algorithm in Case of Disaster Prediction," *International Journal of Computer and Information Engineering*, vol. 10, no. 6, pp. 1075–1081, 2016, doi:10.5281/zenodo.1124690.

[25] H. Kim, J. Lee, and J. W. Jang, "BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices," in *International Conference on Future Internet of Things and Cloud (FiCloud)*, Rome, Italy, August 2015, pp. 558–563, doi:10.1109/FiCloud.2015.21.

[26] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-Hop Routing for Wireless Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 133–144, 2005, doi:10.1145/1090191.1080108.

[27] E. Rozner *et al.*, "SOAR: Simple Opportunistic Adaptive Routing Protocol for Wireless Mesh Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1622–1635, 2009, doi:10.1109/TMC.2009.82.

[28] B. K. Maharjan, U. Witkowski, and R. Zandian, "Tree Network based on Bluetooth 4.0 for Wireless Sensor Network Applications," in *European Embedded Design in Education and Research Conference (EDERC)*, Milan, Italy, September 2014, pp. 172–176, doi:10.1109/EDERC.2014.6924382.

[29] G. Patti, L. Leonardi, and L. Lo Bello, "A Bluetooth Low Energy real-time protocol for Industrial Wireless mesh Networks," in *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Florence, Italy, October 2016, pp. 4627–4632, doi:10.1109/IECON.2016.7793093.

[30] K. Mikhaylov and J. Tervonen, "Multihop Data Transfer Service for Bluetooth Low Energy," in *International Conference on ITS Telecommunications (ITST)*, Tampere, Finland, November 2013, pp. 319–324, doi:10.1109/ITST.2013.6685566.

[31] Y. K. Reddy *et al.*, "Demo: A Connection Oriented Mesh Network for Mobile Devices Using Bluetooth Low Energy," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Seoul, South Korea, 2015, pp. 453–454, doi:10.1145/2809695.2817850.

[32] Z. Guo *et al.*, "An On-Demand Scatternet Formation and Multi-Hop Routing Protocol for BLE-based Wireless Sensor Networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, USA, March 2015, pp. 1590–1595, doi:10.1109/WCNC.2015.7127705.

[33] A. Balogh *et al.*, "Service Mediation in Multihop Bluetooth Low Energy Networks based on NDN Approach," in *International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, September 2015, pp. 285–289, doi:10.1109/SOFTCOM.2015.7314123.

[34] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *IEEE INFOCOM*, vol. 3, Kobe, Japan, 1997, pp. 1405–1413, doi:10.1109/INFCOM.1997.631180.

[35] J. Raju and J. Garcia-Luna-Aceves, "A comparison of on-demand and table driven routing for ad-hoc wireless networks," in *IEEE International Conference on Communications (ICC)*, vol. 3, New Orleans, LA, USA, 2000, pp. 1702–1706, doi:10.1109/ICC.2000.853784.

[36] W. List and N. Vaidya, "A routing protocol for k-hop networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 4, Atlanta, GA, USA, 2004, pp. 2545–2550, doi:10.1109/WCNC.2004.1311489.

[37] J. Jun and M. L. Sichitiu, "MRP: Wireless Mesh Networks Routing Protocol," *Computer Communications*, vol. 31, no. 7, pp. 1413–1435, May 2008, doi:10.1016/j.comcom.2008.01.038.

[38] Qualcomm, "CSRmesh Development Kit," accessed on Sept. 25, 2024. [Online]. Available: https://www.qualcomm.com/products/technology/bluetooth/csrmesh-development-kit

[39] Nordic Semiconductor, "nRF OpenMesh (formerly nRF51-ble-broadcast-mesh)," accessed on Sept. 25, 2024. [Online]. Available: https://github.com/NordicPlayground/nRF51-ble-bcast-mesh

[40] Wirepas, "Wirepas Mesh," accessed on Sept. 25, 2024. [Online]. Available: https://haltian.com/resource/what-is-wirepas-mesh/

[41] MeshTek, "Advanced Bluetooth Mesh Network," accessed on Sept. 25, 2024. [Online]. Available: https://www.meshtek.com/meshtek-platform/

[42] Mindtree, "EtherMind Bluetooth Mesh," accessed on Sept. 25, 2024. [Online]. Available: https://www.ltimindtree.com/services/customer-success/product-engineering-services/wireless-ip-engineering-services/ethermind-bluetooth-mesh/

[43] "Estimote," accessed on Sept. 25, 2024. [Online]. Available: https://estimote.com/

[44] NXP Semiconductors, "Bluetooth Smart/Bluetooth Low Energy," accessed on Sept. 25, 2024. [Online]. Available: https://www.nxp.com/products/wireless-connectivity/bluetooth-low-energy:BLUETOOTH-LOW-ENERGY-BLE

[45] P. Levis *et al.*, "The Trickle Algorithm," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6206, March 2011. [Online]. Available: https://tools.ietf.org/rfc/rfc6206

[46] D. S. J. De Couto *et al.*, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, July 2005, doi:10.1007/s11276-005-1766-z.

[47] T. Lee *et al.*, "A Synergistic Architecture for RPL over BLE," in *Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, London, UK, June 2016, pp. 1–9, doi:10.1109/SAHCN.2016.7732968.

[48] T. Winter *et al.*, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Requests for Comments, Internet Engineering Task Force (IETF), RFC 6550, March 2012. [Online]. Available: https://tools.ietf.org/rfc/rfc6550

[49] L. Zhang *et al.*, "Named Data Networking," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, July 2014, doi:10.1145/2656877.2656887.

[50] V. Jacobson *et al.*, "Networking Named Content," in *International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Rome, Italy, 2009, pp. 1–12, doi:10.1145/1658939.1658941.

[51] E. Royer and Chai-Keong Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, 1999, doi:10.1109/98.760423.

[52] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*. Boston, MA: Springer US, 1996, pp. 153–181, doi:10.1007/978-0-585-29603-6_5.

[53] C. Perkins and E. Royer, "Ad-Hoc On-Demand Distance Vector Routing," in *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, New Orleans, LA, USA, 1999, pp. 90–100, doi:10.1109/MCSA.1999.749281.

[54] Freifunk, "B.A.T.M.A.N. Advanced," accessed on Sept. 25, 2024. [Online]. Available: https://www.open-mesh.org/projects/batman-adv/wiki/Wiki

[55] L. Liu *et al.*, "Performance Evaluation of BATMAN-Adv Wireless Mesh Network Routing Algorithms," in *IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, Shanghai, China, 2018, pp. 122–127, doi:10.1109/CSCloud/EdgeCom.2018.00030.

[56] Freifunk, "B.A.T.M.A.N. daemon," accessed on Sept. 25, 2024. [Online]. Available: https://www.open-mesh.org/projects/batmand/wiki/Doc-overview

[57] K. Kiran *et al.*, "Experimental Evaluation of BATMAN and BATMAN-Adv Routing Protocols in a Mobile Testbed," in *IEEE Region 10 Conference (TENCON)*, Jeju, Korea (South), 2018, pp. 1538–1543, doi:10.1109/TENCON.2018.8650222.

[58] u-blox, "BMD-300 – Stand-alone Bluetooth Low Energy Module," accessed on Sept. 25, 2024. [Online]. Available: https://content.u-blox.com/sites/default/files/BMD-300_DataSheet_UBX-19033350.pdf

[59] Nordic Semiconductor, "nRF52832 System-on-Chip," accessed on Sept. 25, 2024. [Online]. Available: https://www.nordicsemi.com/products/nrf52832

Luca Davoli (S'14–M'17) is a (non-tenured) Assistant Professor at the Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy. He obtained his Dr. Ing. degree in computer engineering and his Ph.D. in Information Technologies at the Department of Information Engineering of the same university, in 2013 and 2017, respectively. He is a Research Scientist at things2i ltd., a spin-off of the University of Parma dedicated to IoT and smart systems. His research interests focus on IoT, Pervasive Computing, Big Stream and Software-Defined Networking.

Massimo Moreni is a developer and project manager at the wireless and IoT department of TCI Telecomunicazioni Italia s.r.l., Saronno, Italy. He has more than twenty years' experience in 2.4 GHz wireless systems, since 2000 as Field Application Engineer for ZigBee components and since 2010 in Bluetooth Mesh Smart Lighting applications. In TCI Telecomunicazioni Italia he has developed a proprietary wireless mesh communication protocol based on BLE technology (BLL - Blue Light Link). His professional interests are IoT, wireless mesh networks, open source home/building automation, and energy harvesting wireless sensors.

Gianluigi Ferrari (S'96–M'98–SM'12) received the Laurea (*summa cum laude*) and Ph.D. degrees in electrical engineering from the University of Parma, Parma, Italy, in 1998 and 2002, respectively. Since 2002, he has been with the University of Parma, where he is currently a Full Professor of telecommunications and also the coordinator of the Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy. He is co-founder and President of things2i ltd., a spin-off of the University of Parma dedicated to IoT and smart systems. His current research interests include signal processing, advanced communication and networking, and IoT and smart systems.

APPENDIX

APPENDIX A: ROUTING PATHS

With reference to Fig. 9, a list of all the possible routing paths among the nodes (without considering a BOM flooding interruption due to the hop counter h) is shown in Table VIII. For the sake of completeness, routing paths considered as "not allowed" because of the presence of endless loops are highlighted in gray color.

TABLE VIII: Routing paths among the nodes composing the topology shown in Fig. 9.

Distance d from $\mathcal{G}_k$	**Routing paths**
$d=1$	$\mathcal{G}_k$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$
$d=2$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_4$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_9$
$d=3$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_4$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_4$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_9$←$\mathcal{S}_7$
$d=4$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_4$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_4$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_8$
$d=5$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_4$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_4$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$
$d=6$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_8$; $\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_1$←$\mathcal{S}_4$←$\mathcal{S}_1$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{G}_k$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_6$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_2$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_4$
$d=7$	$\mathcal{G}_k$←$\mathcal{S}_1$←$\mathcal{S}_2$←$\mathcal{S}_6$←$\mathcal{S}_8$←$\mathcal{S}_7$←$\mathcal{S}_9$←$\mathcal{S}_7$; $\mathcal{G}_k$←$\mathcal{S}_7$←$\mathcal{S}_8$←$\mathcal{S}_6$←$\mathcal{S}_2$←$\mathcal{S}_1$←$\mathcal{S}_4$←$\mathcal{S}_1$;