IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 19, 2018, accepted April 21, 2018, date of publication May 8, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2834479

Multi-Hop Real-Time Communications Over Bluetooth Low Energy Industrial Wireless Mesh Networks

LUCA LEONARDI, GAETANO PATTI, AND LUCIA LO BELLO, (Senior Member, IEEE)
Department of Electrical, Electronic and Computer Engineering, University of Catania, 95125 Catania, Italy
Corresponding author: Lucia Lo Bello (lobello@unict.it)

ABSTRACT Industrial wireless sensor networks (IWSNs) are used to acquire sensor data that need real-time processing, therefore they require predictable behavior and real-time guarantees. To be cost effective, IWSNs are also expected to be low cost and low power. In this context, Bluetooth low energy (BLE) is a promising technology, as it allows implementing low-cost industrial networks. As BLE is a short-range technology, a multihop mesh network is needed to cover a large area. Nevertheless, the recently published Bluetooth mesh networking specifications do not provide support for real-time communications over multihop mesh networks. To overcome this limitation, this paper proposes the multihop real-time BLE (MRT-BLE) protocol, a real-time protocol developed on top of BLE, that allows for bounded packet delays over mesh networks. MRT-BLE also provides priority support. This paper describes in detail the MRT-BLE protocol and how to implement it on commercial-off-the-shelf devices. Two kinds of performance evaluation for the MRT-BLE protocol are provided. The first one is a worst case end-to-end delay analysis, while the second one is based on the experimental results obtained through measurements on a real testbed.

INDEX TERMS Bluetooth low energy, industrial wireless sensor networks, real-time, wireless mesh networks.

I. INTRODUCTION AND MOTIVATION

Bluetooth Low Energy [1] is a wireless technology intended for low-power, low-cost, low-complexity short-range communications, that represents an interesting solution for implementing Industrial Wireless Sensor Networks (IWSN). In fact, BLE provides lower energy consumption compared to other wireless technologies adopted for industrial applications [2], [3]. Moreover, thanks to the low cost of devices and to their diffusion, BLE is a good candidate for several industrial applications [4]. However, the BLE specifications [1], [5], [6] do not include support for real-time traffic, thus the BLE standard is not suitable for IWSN, which require bounded packet delays. A recent work [7] has provided a configuration method for the BLE standard to guarantee bounded packet latencies over star networks. As BLE is a short-range technology, a multi-hop mesh network is needed to cover a large area. However, mesh topologies are not foreseen in the new Bluetooth version 5 specification [6]. The Bluetooth Smart Mesh Working Group has recently introduced the Bluetooth mesh networking specifications [8], which define the requirements to enable mesh networking solutions for BLE. Unfortunately, such specifications do not provide support for real-time communications over multi-hop mesh networks.

To overcome the above discussed limitations, thus enabling BLE to be introduced in industrial environments, this paper proposes the Multi-hop Real-time BLE (MRT-BLE), that is a real-time protocol, developed on top of BLE, able to realize low-cost IWSN with mesh topologies.

The MRT-BLE protocol is connection-oriented, unlike the Bluetooth mesh networking specifications [8], which are defined as connectionless communications. There are two reasons for this choice. Firstly, the connection-oriented approach provides a higher throughput than the connectionless one, as packets can be transmitted more frequently. Secondly, in the connection-oriented approach, 37 channels can be exploited for channel hopping, instead of the 3 channels that can be used for connection-less advertising. This entails a lower channel collision probability and therefore a higher reliability. The MRT-BLE protocol adopts a Time Division Multiple Access (TDMA) approach that exploits a transmission allocation scheme able to support the timely

exchange of multi-hop real-time data packets. This paper builds upon the approach proposed in [7], that outlines a protocol working on top of BLE. Such a protocol allows to maintain bounded latencies over mesh networks. However, in the work in [7] the approach is merely sketched. The ability of taking advantage of the Client Characteristic Configuration Descriptor (CCCD) in order to enable/disable connection is mentioned, but nothing is said about the timing of this mechanism. Moreover, the work in [7] does not address the implementation of the MRT-BLE on Commercial-Off-The-Shelves (COTS) devices. Instead, this paper aims to describe in detail how the MRT-BLE works, address the timing of the connection enabling/disabling, introduce the support for packet priority, and offer two kinds of evaluation. The first one is based on measurements obtained in a realistic scenario, while the second one is a worst-case end-to-end delay analysis. The contribution of the paper consists in the MRT-BLE mechanism itself and the relevant timing analysis that represents a useful tool for the network designers, since it allows them to assess whether a feasible schedule for a given flow set can be found or not. In the negative case, the timing analysis helps designers in the re-engineering process, allowing them to see the effects of changing the flow periods or paths along the network. To the best of our knowledge, this is the first work that proposes and assesses a mechanism to provide real-time multi-hop communications over BLE through assessment in a real scenario. The paper is organized as follows. Section II overviews related work, while Section III summarizes BLE and the most relevant features exploited by the proposed protocol. Section IV presents the MRT-BLE protocol design, while Section V provides a timing analysis of the protocol. Section VI describes an implementation of the MRT-BLE protocol on real devices and presents some experimental results. Finally, Section VII gives our conclusions and hints for future work.

II. RELATED WORK

Bluetooth Low Energy (BLE) is one of the most interesting solutions for short-range communications, thanks to its low-cost, high throughput, broad compatibility with mobile devices, and low-power properties. BLE is gaining ground in specific sectors, for instance, energy management for smart homes [9], [10], food manufacturing and traceability [11], [12], and also in applications, such as machine monitoring and structural health monitoring, in which other technologies, e.g., the IEEE 802.15.4 standard, proved to be effective and efficient too [13], [14]. In [15] the IEEE 802.15.4 and BLE protocols are evaluated in terms of service ratio, delay, and energy efficiency under IPv6 traffic.

Some studies analyzed the BLE system behavior in wireless environments under realistic operating conditions. In particular, the work in [16] addresses the coexistence of IEEE 802.15.4, BLE and IEEE 802.11 through analysis and experiments to determine the effect of cross-technology interference on the wireless network reliability. The paper concludes that the MAC layer mechanisms of both IEEE 802.15.4 and BLE improve reliability and that cooperative solutions are required to achieve coexistence. In [17] an extension of the BLE channel access scheme in connection state that is based on the listen-before-talk mechanism is proposed. The numerical results in the paper show that the proposed scheme reduces the average transmission delay.

Several studies investigated the BLE discovery phase, which plays a critical role in the networks with dynamic topology. For instance, Cho *et al.* [18] address the influence of parameter settings on the latency and energy performance of the discovery process. A mechanism to enhance the discovery phase for BLE devices is proposed in [19]. The mechanism aims at avoiding collisions during the advertisement process, so as to achieve lower latency and energy consumption.

Modern BLE radio transceivers allow to partition the network bandwidth between the BLE and another user-defined protocol. Marinoni *et al.* [20] show the feasibility of a dual-protocol approach and its capability to support a custom real-time protocol running on top of the raw radio layer. The proposed real-time protocol introduces a bounded overhead and can be implemented in specific devices able to support two different protocols.

In order to increase the communication coverage of BLE, Kim *et al.* [21], Jung *et al.* [22], Zenker *et al.* [23], and Hortelano *et al.* [24] proposed different methods prior to the publication of the Bluetooth mesh networking specifications. For instance, Kim *et al.* [21] proposes a method to implement a BLE network with a tree topology to extend the coverage to other wireless sensor networks. A cluster-based on-demand routing protocol to support multihop communication in Bluetooth low energy ad hoc networks is proposed in [22]. In [23] the applicability of BLE for mesh-enabled applications is addressed. The results, obtained using a prototype of the proprietary BLE-based CSRmesh protocol [25], show that BLE mesh is a promising technology for mesh applications, but that more studies are needed to fully exploit its potential. Hortelano *et al.* [24] present a new mesh proposal based on the proprietary BLE-based CSRmesh protocol, that exploits a connectionless network and tries to overcome some limitations of the CSRmesh network (e.g. the scalability, the high number of retransmissions). Finally, Darroudi and Gomez [26] survey state-of-the-art BLE mesh network solutions and discuss their advantages and drawbacks.

The IP support has recently turned BLE into a potential candidate for a broad range of applications including healthcare, wearable devices, home automation, and Internet of Things (IoT) [27]–[29]. The delay performance of BLE for time-critical applications is addressed in [30]. The work analytically models the delay for connection-oriented applications under different bit error conditions. The works in [31] and [4] explored the BLE suitability for industrial and process automation communications and for time-critical Industrial IoT (IIoT) applications, respectively. In particular, the last one [4] presents a retransmission scheme able

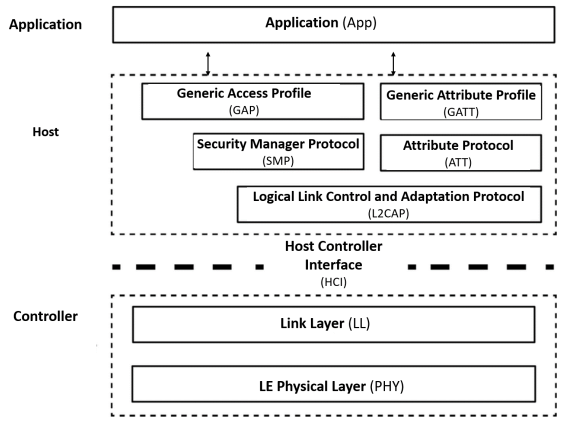

FIGURE 1. The BLE protocol stack.

to fulfill the reliability and timeliness requirements of IIoT applications.

The above mentioned literature indicates that it is worthwhile investigating novel extensions and configurations of BLE to make it more suitable for industrial environments. For instance, in [7], a configuration method for the BLE standard that guarantees bounded packet latencies with star topologies was proposed, together with a new protocol for BLE-based mesh topologies that is able to provide bounded latencies. However, Patti *et al.* [7] leave for future work both the development of the proposed solution on COTS devices and the protocol assessment in a realistic multi-hop scenario. Comparing with the work in [7], that simply outlines the MRT-BLE idea, this paper turns the idea into a protocol, characterizing in detail the mechanisms to guarantee bounded latencies on multihop communications and their timing. This paper also provides a timing analysis for multihop communications to compute the worst-case end-to-end delay of periodic packets. In addition, this work addresses the MRT-BLE protocol implementation on COTS devices and assesses the end-to-end delay performance in a realistic multi-hop scenario. The measured delay values are compared with the computed worst-case end-to-end delays obtained through the analysis in Section V. As it will be shown, the theoretical upper bounds are found compliant with the measured values.

The proposed MRT-BLE protocol differs from the Bluetooth mesh networking specifications [8] as the latter are defined as connectionless communications, based on the GAP Broadcaster and Observer roles, while the MRT-BLE protocol is connection-oriented and offers support for real-time communications over multi-hop mesh networks.

III. BLE SUMMARY

The Bluetooth Low Energy core system architecture includes an RF transceiver and a protocol stack, shown in Fig. 1, that enable BLE devices to connect and exchange data.

The lowest system layers are grouped into a subsystem known as the Controller. This is a common implementation that uses a standard interface, called the Host Controller Interface (HCI), which enables two-way communications with the upper layers of the protocol stack that realize the so-called Host in Fig. 1. Applications are developed on top of the Host. The following subsections summarize the most important features of the BLE Link Layer, the Attribute Protocol and the Generic Attribute Profile. Such features are maintained in the BLE versions 4.1 [1], 4.2 [5] and 5.0 [6].

A. LINK LAYER

The operation of the BLE Link Layer (LL) can be described through a state machine with the following states: Standby, Advertising, Scanning, Initiating, and Connection. The Link Layer may run multiple state machine instances and each Link Layer state machine allows only one active state at a time. When two devices are engaged in a connection, they can play the master or the slave role. The master is the device that initiates the connection and coordinates the medium access, periodically polling the slaves in a Time Division Multiple Access (TDMA) way.

The master controls the timing of the connection events (CE), i.e., the synchronization points for the master and the slave. Since two devices must be tuned to the same RF channel to communicate and the Link Layer may use one physical channel at a given time, the master and slave shall also determine the data channel index for each connection event to reduce the collisions. A physical channel collision is possible because many BLE devices can independently operate within the same spatial and temporal area, and the number of physical channels (up to 37) is limited, thus two independent BLE devices may have their transceivers tuned to the same physical channel.

The start time of a connection event, called an Anchor Point (AP), is the instant at which the master shall start to transmit a packet to the slave. During a connection event, the master and the slave send and receive packets alternately. Consecutive packet transmissions are separated by an Inter Frame Space (IFS), whose duration is 150 μs. A connection event is considered open as long as both devices keep sending packets. If none of the devices has data to transmit, the slave switches to the sleep mode until the next AP.

The BLE specifications define two timing parameters for connection events, i.e., the Connection Interval (CI) and the Connection Slave Latency (CSL). The CI represents the time interval that regularly spaces the start times of connection events. The second parameter, the Connection Slave Latency, defines the number of consecutive connection events that a slave has to skip, while remaining in sleep mode, when it does not need to communicate with the master.

Several BLE implementations [32] require a guard band (GB) between the connection events to cope with the synchronization accuracy, and the master shall ensure that a connection event closes at least one guard band before the anchor point of the next connection event. Moreover, such BLE implementations provide configurable parameters to set the maximum duration of each connection event. The master

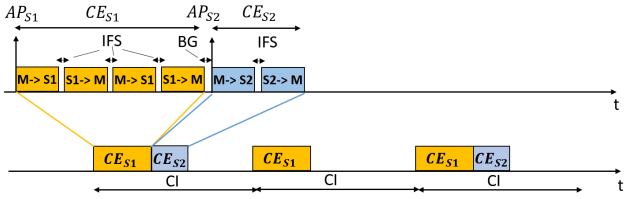

FIGURE 2. Example of the BLE master scheduling for multiple connections.

has the ability to schedule the connection event anchor point at a time of its choice to efficiently schedule connection events for the multiple connections it is involved in. This way, the master basically splits the connection interval into as many connection events as the number of connections.

Fig. 2 shows an example in which there are two slaves (S1 and S2) and one master (M). In Fig. 2, the lower time axis gives the schedule of the connection events for the two connections, while the upper time axis shows the details of each connection event.

At the anchor point of the slave S1 (AP_{S1}) the master starts the polling, transmitting a packet to S1, thus beginning the alternating transmission sequence (M to S1, S1 to M, etc.). The connection event for the slave S1 periodically repeats with a period equal to CI. At the end of the first CE for S1, here called CE_{S1}, after the guard band the master starts the connection event for the slave S2 (CE_{S2}), which periodically repeats with a period equal to 2*CI.

While the connection interval CI is the same for all the slaves connected to the same master, the anchor points of the relevant slaves are shifted, so the master polls one slave at a time. Every Link Layer connection uses a mechanism of acknowledgement and flow control. Consequently, the Link Layer guarantees the retransmission of the unacknowledged packets.

B. ATTRIBUTE PROTOCOL

The Attribute Protocol (ATT) [1] allows a device, named the server, to expose a set of attributes and their associated values to a peer device, named the client. An attribute is a value that has three properties associated with it, i.e., the attribute type, the attribute handle and a set of permissions. The client can discover, read, and write the attributes exposed by the server. The server can indicate and notify its attributes. For example an ATT server can expose three attributes that represent the data detected by three sensors (e.g., temperature, pressure and humidity) and periodically notifies the connected clients with new measured data. In another scenario, these attributes can be read by the clients when necessary. A client may send an Attribute Protocol request to a server and the server shall respond to all the received requests. A device can implement both the client and the server roles, which can concurrently run in the same device, but each device shall run only one instance of a server.

The Attribute Protocol procedures typically use a sequential request-response protocol. An Attribute Protocol

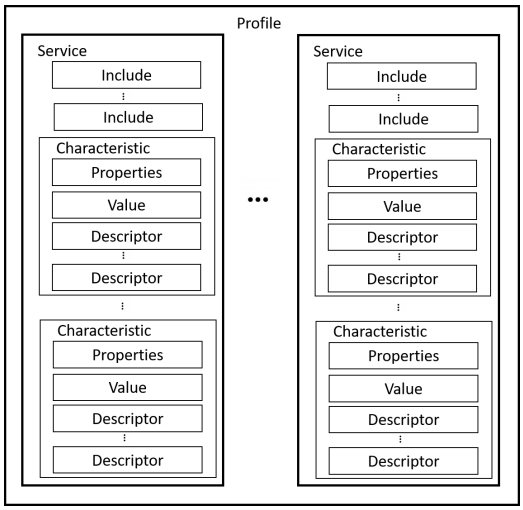

FIGURE 3. GATT profile hierarchy.

{request, response} pair is considered a single transaction, i.e., once a client sends a request to a server (e.g., a Write request to write a server attribute), that client shall not send other requests to the same server until a response Protocol Data Unit (PDU) has been received (e.g., a Write response sent by the server to inform the client of the result of the Write request). Conversely, there are procedures, e.g., the notification (a server-initiated procedure) that do not have a response PDU. For example, a server notifies an attribute that contains the temperature detected by a sensor and updates its value. This value is received by the connected clients that have enabled the receiving of notification packets after the update of a specific attribute.

C. GENERIC ATTRIBUTE PROFILE

The Generic Attribute Profile (GATT) [1] defines a service framework built on top of the Attribute Protocol. The GATT profile is designed to be used by an application or another profile so that a client can communicate with a server. The GATT defines the hierarchical data structure, shown in Fig. 3, that a device exposes to the connected devices and also defines the way two BLE devices exchange standard packets.

The server contains some attributes and the GATT Profile defines how to use the Attribute Protocol to discover, read, write, notify, indicate, and obtain indications about these attributes.

The top level of the hierarchy is the Profile, which is composed of one or more services that are needed to fulfill a use case. A service is generally composed of characteristics or references to other services. Each characteristic consists of various fields, i.e., a set of properties, a value, and one or more optional descriptors. An important descriptor is the Client Characteristic Configuration Descriptor (CCCD) that configures for the client a characteristic on the server (e.g., enabling/disabling of notifications and/or indications).

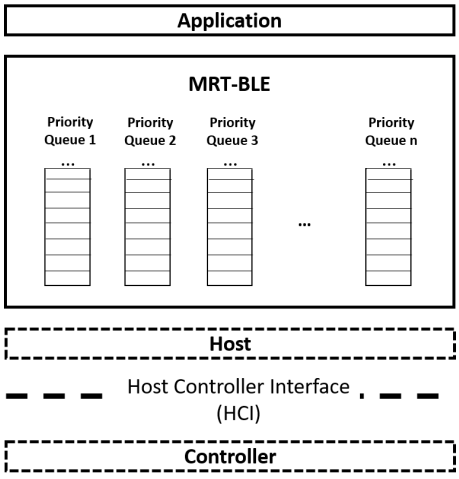

FIGURE 4. The BLE protocol stack including the MRT-BLE.

For example [33] a profile may expose two services, i.e., the acceleration service and the environmental service. The acceleration service contains a characteristic called free-fall that cannot be read or written, but can be notified. The application will send a notification on this characteristic if a free-fall condition is detected by a sensor. Notifications can be enabled or disabled by a client writing on the related CCCD. The environmental service contains three characteristics (with read-only properties) that expose data from some environmental sensors, i.e., temperature, pressure and humidity. Each characteristic of the environmental service has a characteristic format descriptor that describes the type of data contained inside the characteristic.

The basic elements used in a profile, i.e., services and characteristics, are contained in the attributes used in the Attribute Protocol.

The Attribute Protocol Maximum Transfer Unit (ATT_MTU) is defined as the maximum size of any packet exchanged between a client and a server. The GATT client and server implementations shall support an ATT_MTU not smaller than the default value (23 bytes).

IV. PROTOCOL DESIGN

The Multi-hop Real-time Bluetooth Low Energy (MRT-BLE) protocol for Industrial Wireless Sensor Networks proposed in this paper allows the creation of BLE-based mesh networks able to provide bounded packet delays on multi-hop data transmissions, thus offering support for real-time communications. Fig. 4 shows the positioning of the MRT-BLE protocol in the BLE protocol stack.

The main idea behind the MRT-BLE protocol is to subdivide the network into a number of sub-networks, each one coordinated by a master. Two sub-networks are linked by a device that is either a shared slave or a master/slave device. These devices act as "bridges" among the sub-networks. As the masters of the two linked sub-networks are not synchronized, connection events may overlap at the bridge nodes, thus determining a collision that does not make it possible to guarantee transmission.

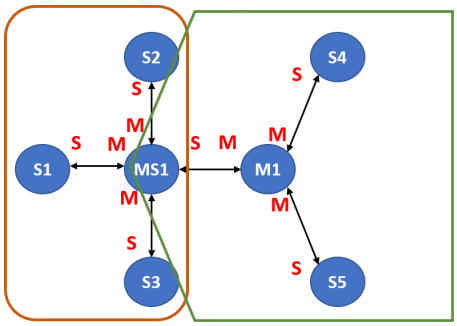

FIGURE 5. Example of topology with a master/slave device.

Fig. 5 shows an example of network topology in which two sub-networks are linked by a master/slave device.

In Fig. 5 the sub-networks are coordinated by the nodes MS1 and M1, respectively. The node MS1 is a master/slave device, as it plays the role of master for the nodes S1, S2 and S3, and the role of slave for the node M1. Since M1 and MS1 are not synchronized, the communications in which MS1 acts as a slave may overlap with those in which it plays the master role. While MS1 is acting as a slave, its timing is defined by the master M1, therefore the timing of the connection event in which MS1 acts as a slave is not synchronized with the time base mechanism of the MS1 master connections.

However, a viable solution to avoid the above-mentioned problem is to give MS1 a proper schedule so that, while MS1 is communicating with the nodes of its subnetwork (i.e., S1, S2 and S3), M1 avoids transmitting to MS1. On the contrary, S4 and S5 may transmit simultaneously with M1, as MS1 is not intended to communicate directly with M1. The approach here proposed is based on the latter consideration. In general, the problem of enabling adjacent subnetworks to communicate can be solved by scheduling adjacent subnetwork transmissions in two alternate timeslices, so that when a shared node (e.g., MS1) schedules the communication with the nodes of its subnetwork, its adjacent master is prevented from communicating with the shared node.

To achieve this result, thus avoiding communication overlap, in the proposed approach for each connection the duration of one CE is configured offline, so as to allow the transmission of one packet between the slave and the master and of one packet in the opposite direction. The connection intervals for all connections have to be set with the same duration. Moreover, CIs have to be sized so as to allow the communication between all slaves and the master of a sub-network once. This way, the duration of one timeslice is equal to the duration of one CI and the shared node alternates its connections with a sub-network at a time.

In the example shown in Fig. 5 the proposed solution consists in alternating the connections of the node MS1 with its master, with those with its slaves. This means that, in a given timeslice (according to the proposed solution a timeslice is

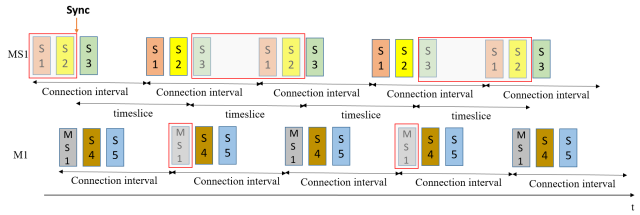

FIGURE 6. Solution to the overlapping CE problem with a master/slave shared device.

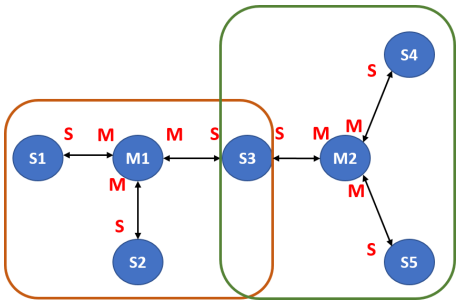

FIGURE 7. Example of topology with a shared slave device.

a multiple of one CI) the node MS1 will disable the connection with the node M1 and enable the connections with the nodes S1, S2, and S3, while in the next timeslice it will operate in the opposite way. This solution can be easily realized enabling/disabling a connection.

Each device, before sending data, has to check that the connection is enabled. If this is not the case, the master will insert the packet in a queue managed by the MRT-BLE protocol and transmit it in the next connection event. This way the CE overlap is avoided.

Fig. 6 shows the timing of the masters MS1 and M1 relevant to the topology shown in Fig. 5. The semi-transparent rectangles represent the timeslices during which some connections are disabled.

Fig. 6 shows as the node MS1, acting as a GATT client, can enable/disable the connections with the devices that act as a GATT server. In the initial phase, MS1 enables the connection with the master M1 (shown in the lower timeline), while keeping disabled the master connections. Once MS1 has received the first packet from M1 (considered a synchronization packet), it disables the connection with the master M1 and enables the master connections with the nodes S1, S2, and S3 (shown in the upper timeline). After a time equal to the connection interval (in the example, the timeslice duration is equal to one CI), the node MS1 disables the master connections and enables the connection with M1, for one timeslice. This mechanism repeats over time.

Fig. 7 shows an example of network topology in which there are two sub-networks linked by a shared slave device.

In Fig. 7 the sub-networks are coordinated by nodes M1 and M2, respectively.

As the node S3 is a shared slave it has to communicate with two masters that are not synchronized with each other, therefore the connection events of this node may overlap.

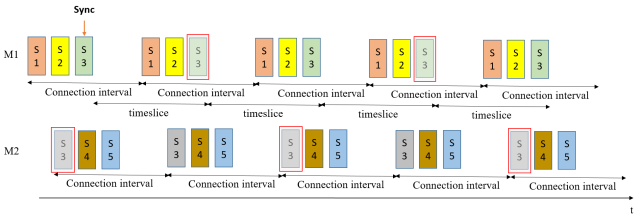

FIGURE 8. Solution to the overlapping CE problem in the shared slave topology.

In the example shown in Fig. 7 the proposed solution consists in alternating the connections of the node S3 with the masters. This means that, in a given connection interval the node S3, acting as a GATT client, will disable the connection with the node M1 and enable the one with the node M2, while for the next connection interval the node S3 will enable the connection with the node M1 and disable the one with the node M2.

In this case, the shared slave S3 starts alternating its connections, for example, upon receiving a synchronization packet. Assuming that S3 receives the synchronization packet from M1, the shared slave will start the mechanism at this very time instant. First, the node S3 disables the connection with the node M1 and enables the connection with the node M2. After a time equal to one timeslice, the node S3 will enable the connection with the node M1 and disable the connection with the node M2. The described mechanism repeats over time. This way the CE overlap problem is avoided.

Fig. 8 shows the timing of the masters M1 and M2 in the topology shown in Fig. 7. The semi-transparent rectangles represent the intervals during which the node S3 disables the connection with one of the two masters.

Fig. 8 shows that the node S3 in the initial phase enables the notification only for the connection with the master M1. Upon receiving the sync packet, the node S3 disables the connection with the master M1 and enables the connection with the master M2. After one timeslice (in this case, equal to one connection interval) the node S3 makes the opposite action, i.e., disables the connection with M2 and enables the connection with M1, for another timeslice. This mechanism repeats over time.

Note that Fig. 6 and Fig. 8 show connections alternating every connection interval (i.e., in these examples, the timeslice duration is equal to one connection interval).

In order to achieve bounded delays, the routing is static and configured offline. To improve the fault-tolerance, it is possible to provide multiple paths to be enabled in case of faults.

In order to deploy a mesh network without connection event overlap, the MRT-BLE imposes the following configuration rules:

1) A node not acting as a master can establish a connection with up to two masters.

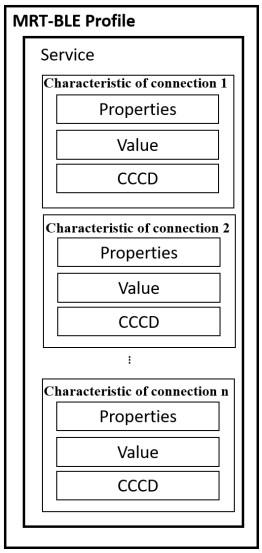

FIGURE 9. MRT-BLE Profile.

2) A node (A) acting as a master can establish a connection with at most another master (B). In this connection, the node A shall play the slave role.
3) The connection intervals of the different sub-networks must have the same duration.

According to the BLE specifications [1], a device can act both as a master and a slave for different connections and can also establish connections with multiple masters.

To realize the described mechanisms, the solution proposed in this paper is based on the transmissions from the GATT servers that notify their Characteristic Values. Each device is both a GATT client and a GATT server and defines a profile, called the MRT-BLE Profile, shown in Fig. 9.

The MRT-BLE profile is composed of a service that includes a number of characteristics equal to the number of connections. Each characteristic is used to communicate on a specific connection (according to a programmer-defined static configuration) and consists of a set of properties, a value, and the client characteristic configuration descriptor (CCCD). In particular, the *value* field contains the data to be sent, the CCCD is a bit field where a set bit indicates an enabled action (e.g., notification), while a cleared bit indicates a disabled action. *Properties* is a bit field that determines how the *value* can be used and how the CCCD can be accessed (e.g., read, write without response, notify, etc.). The *characteristic of the connection* indicates the specific characteristic that is used to send packets on a specific connection.

In the following of the paper, when the *enabling/disabling of a connection* is mentioned, it means that the GATT client uses a {request, response} GATT procedure to write the CCCD of the characteristic of the connection on the server, so as to enable/disable the notification procedure of the GATT server. Consequently, a *disabled connection* indicates that the notifications of the characteristic of the connection are disabled, so the GATT servers involved in the connection cannot notify the value of this characteristic. An *enabled connection* will indicate the opposite case.

TABLE 1. Summary of notation.

Symbol	Definition
l_{xy}	The link between a node x acting as a master and a node y acting as a slave.
sl_{xy}	The shared link between a master node x and a slave node y, i.e., the link in which at least one of the nodes x and y is shared between two sub-networks.
i	The index of the i-th link.
Xci	The number of connection intervals within the two timeslices that can be used to transmit data packets over one shared link.
Tsw	The time that a node shared between two sub-networks takes to disable the shared links with a sub-network and to enable the other ones.
Tci	The duration of one connection interval.
$NL(i)$	The maximum between the number of shared links of the master and the slave, respectively, for the i-th shared link.
$Tc(i)$	The cycle time, i.e., the time interval after which the entire schedule for the i-th link repeats.
$prio_f$	The priority of a packet of the f-th flow.
P_f	The period of a packet of the f-th flow.
R_f	The routing path of the f-th flow.
h^f	The index of the last link to the destination of the flow f.
RT_f	The maximum response time of a packet of the f-th flow.
$Q_t(l_i^{(f)})$	The maximum time that a packet of the f-th flow waits to be transmitted over the link l_i.
T_{tx}	The maximum time taken by the BLE controller to transmit a packet.
$s_i(t)$	The minimum number of CI start times dedicated to the transmission of data packets in the i-th link which may occur in any interval of length t.
$w_i(X)$	The longest time the i-th link has to wait to see X consecutive start times.
$I_f(t)$	The number of packets generated by the f-th flow in any interval of length t.
Tce	The duration of one connection event.

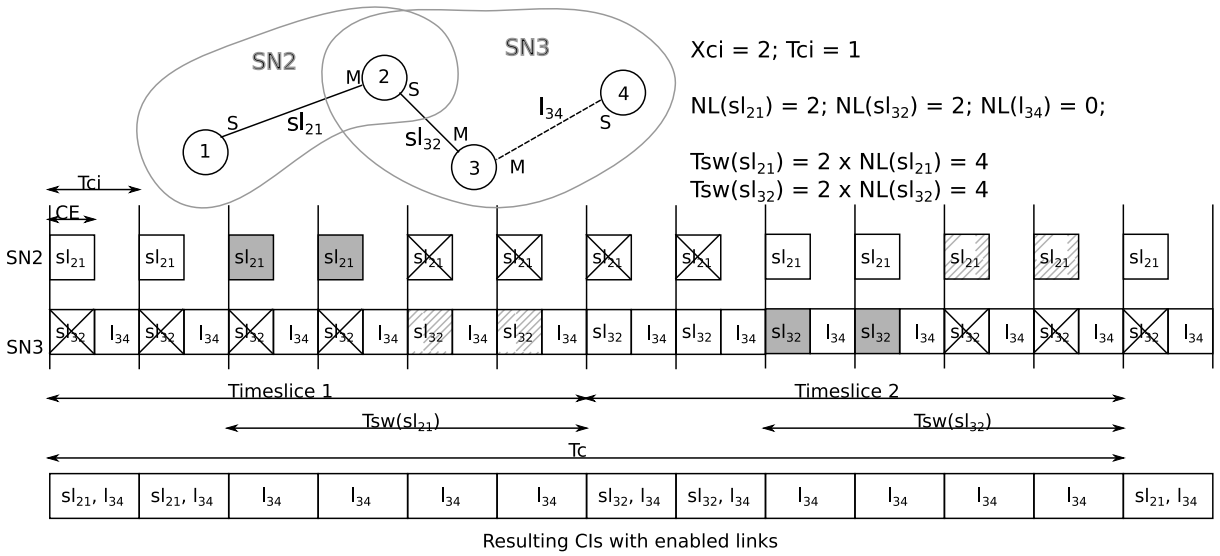

FIGURE 10. Network timings: an example.

The proposed protocol does not implement a mechanism to dynamically configure the network topology, it makes the configuration offline. This aspect will be investigated in future works.

V. TIMING ANALYSIS

The analysis here proposed deals with the calculation of the worst-case end-to-end delay (hereinafter worst-case response time) of periodically transmitted packets belonging to a flow. The resulting worst-case end-to-end delay for each flow represents a lower bound for the minimum deadline that allows a feasible schedule for each flow. The analysis can be used to assess the feasibility, under a static priority assignment, of a flow set comparing the worst-case response time of each flow with the corresponding relative deadline. Table 1 summarizes the notation used in the analysis.

A connection between a master and a slave is defined as a link. A link is a bidirectional connection denoted by l_{xy} (where x is the index of the node acting as a master for the link and y the index of the slave). In the proposed approach for each link the duration of one CE is offline configured so as to allow the transmission of one packet between the slave and the master plus one packet between the master and the slave. The connection intervals for all links have to be set with the same duration and allow the communication between all slaves and the master of a sub-network once. A link in which at least one of the two connecting nodes is shared between two sub-networks is defined as a *shared link* (sl_{xy}). As discussed in Sect. IV, in the proposed approach, the time is divided into two timeslices. In each timeslice, a node with shared links alternates the connection with one sub-network at a time. Fig. 10 shows an example of the network timings. In the example the Node 2 acts as a master for the shared link sl_{21} and as a slave for sl_{32}. l_{34} is a link between two nodes that are connected to only one sub-network, so l_{34} is always active as no slot collisions can occur. The number of CIs in the two timeslices that a node can use to transmit data packets on a shared link is denoted by *Xci* and is equal for all the shared links in the network. In the example *Xci* is set to 2 and in fact sl_{21} and sl_{32} are active for only two CIs (sl_{21} in timeslice 1 and sl_{32} in timeslice 2).

To disable one link and enable the other one, the transmission of four packets is needed, i.e., two to disable one link (write_request and write_response) and two to enable the other one. According to the BLE specifications [1] , request and response may be scheduled within a single CE. Many implementations, such as the ST BlueNRG-MS devices [33], take four connection intervals to complete the entire procedure, as they transmit only one request or response in each CI. In the analysis such an interval is called *Tsw*. For example, in the ST BlueNRG-MS devices the *Tsw* of a node to enable/disable the connection of the i-th shared link is calculated as

$$Tsw(i) = 2 \times NL(i) \times Tci, \tag{1}$$

where $NL(i)$ for the i-th shared link is defined as the maximum between the number of shared links of the master and the number of shared links of the slave. The *NL* value for no shared links is 0. Equation (1) evaluates two times $NL(i)$, because the transmission of two packets, i.e., one write_request and one write_response packets, is required to disable/enable one link. Note that, the ST BlueNRG-MS devices [33] take two connection intervals to transmit one request and one response (i.e., only one request or response is transmitted in each CI).

The time interval in which the entire schedule, for the i-th link, repeats is called Cycle Time (Tc), and it is calculated as

$$Tc(i) = \begin{cases} NL(i) > 0 & \Rightarrow 2 \times Xci \times Tci + 2 \times Tsw(i) \\ NL(i) = 0 & \Rightarrow Tci. \end{cases} \tag{2}$$

In the MRT-BLE protocol, each node can generate multiple packets belonging to different flows (f) with different (or equal) priorities ($prio_f$). If two packets have the same priority, they are transmitted in *First-In First-Out* (FIFO) order. Packets are periodically transmitted with a period P_f. The routing path (R_f) for each flow is fixed and configured offline. To improve fault-tolerance, each node has to maintain a routing table with a backup path for each flow (this way the analysis can be repeated for each path). Each flow is therefore characterized by the tuple $(P_f, prio_f, R_f)$, where $R_f = (l_i^f, \ldots, l_h^f)$ is the vector of the links or shared links that a packet of the flow f has to traverse to reach the destination, while h is the index of the last link. The time RT_f taken by a packet of the f-th flow sent from a node to arrive to the destination node is calculated as

$$RT_f = \sum_{i=0}^{h} \left(Q_t(l_i^{(f)}) + T_{tx} \right), \quad \forall n_i \epsilon R_f \tag{3}$$

where $Q_t(l_i^{(f)})$ is the queuing time that a packet of the f-th flow has to wait in the controller at each link to be transmitted and T_{tx} is the maximum time taken by the BLE controller to transmit a packet. T_{tx} has a fixed bound that is equal, in the worst case, to the duration of one connection interval, so in the worst case response-time analysis we assume that $T_{tx} = Tci$. In fact, the analysis here presented assumes that in each CI a node can transmit only one packet and the CE is sized so as to also accommodate retransmissions. So in one Tci the packet is transmitted and retransmitted as long as the CE for the node is not expired. In the following, a timing analysis for calculating the worst case $Q_t(l_i^{(f)})$ in each hop is presented. Our aim as a future work is to deal with a stochastic analysis of flows characterized by a known distribution of the arrival times, as the one proposed in [34].

A. RESPONSE-TIME ANALYSIS

To determine the worst case $Q_t(l_i^{(f)})$ for each link, the worst case response-time (WCRT) analysis is made calculating both the resource availability and the worst case for the resource request. In MRT-BLE the resource availability for the i-th link is given by the number of CIs in the two timeslices (Fig. 10) that a node can use to transmit over the link i, while the resource request is the number of CIs needed to transmit the packets generated and/or forwarded by a node over the link i. The analysis here presented, which applies to fixed priority non-preemptive scheduling, is based on the worst case response-time (WCRT) analysis that was proposed in [35] and extended with the "busy period" approach in [36] and [37]. Consequently, the response times of all the packets of a flow within a busy period have to be examined. The busy period is defined as the maximum interval during which any packet of priority lower than the priority of the f-th flow is unable to start transmission. The busy period starts at the time t^s in which a packet with a priority higher than or equal to the one of the f-th flow is enqueued and there are no packets with priority higher than or equal to the f-th flow, enqueued strictly before t^s, that are waiting to be transmitted. The busy period ends at the earliest time t^e at which there are no packets of priority equal to or greater than the f-th flow, enqueued strictly before time t^e, that are waiting to be transmitted. Hence, all the packets with the same or higher priority than the f-th flow that were enqueued before the end of the busy period are transmitted during the busy period [37].

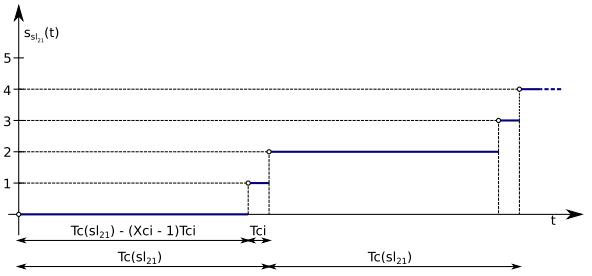

FIGURE 11. Example of the minimum number $s_{sl_{21}}(t)$ of CI start times dedicated to the transmission of data packets in the link sl_{21} for the network in Fig. 10.

In the MRT-BLE protocol the worst case for the resource availability is determined by computing $s_i(t)$, i.e., the minimum number of CI start times dedicated to the transmission of data packets in the i-th link which may occur in any interval of length t. As illustrated in Fig. 11, the worst-case interval (that is the one containing the minimum number of start times) begins when the last CI, dedicated to data packet transmission, has just started. From this instant (please refer to Fig. 11 where the white dots denote the start time of data packets over the time), the time the i-th link has to wait before another data packet starts is $Tc(i) - (Xci - 1)Tci$. Then Xci data packet transmissions start, spaced at intervals of length Tci, and the pattern will repeat with period $Tc(i)$. In the example in Fig. 10, the worst-case interval for the link sl_{21} begins when the second CI has just started. Then the time the link sl_{21} has to wait before another data packet starts is $Tc(sl_{21}) - (Xci - 1)Tci = 12\ Tci - (2 - 1)Tci = 11\ Tci$.

The formal equation of $s_i(t)$ is

$$s_i(t) = \begin{cases} NL(i) > 0 & \Rightarrow \sum_{j=1}^{Xci} \left\lfloor \dfrac{t + (j-1)Tci}{Tc(i)} \right\rfloor \\ NL(i) = 0 & \Rightarrow \left\lfloor \dfrac{t}{Tci} \right\rfloor . \end{cases} \tag{4}$$

where, if $NL(i) > 0$ (i.e., it is a shared link), the start times of Xci connection intervals dedicated to the transmission of data packets are separated by Tci, start times repeat with period Tc and, in the worst case, the first start time occurs after $Tc(i) - (Xci - 1)Tci$. If $NL(i) = 0$ (i.e., the link is not shared) data packets can be transmitted every CI, so the start times repeat with period Tci.

As the link transmissions, thanks to the mechanism here proposed, do not collide with each other, for the $Q_t(l_i^{(f)})$ calculation only the interference of the packets generated and forwarded within the same node has to be analyzed.

To calculate $Q_t(l_i^{(f)})$, the longest time $w_i(X)$ the i-th link has to wait to see X consecutive start times is calculated as

$$w_i(X) = \begin{cases} NL(i) > 0 & \Rightarrow (S+1)Tc(i) - (Xci - 1 - O)Tci \\ NL(i) = 0 & \Rightarrow X \times Tci. \end{cases} \quad (5)$$

where S is quotient of the Euclidean division of $X-1$ by Xci and O is the remainder, i.e., $X - 1 = SXci + O$. For instance, in the case of Figures 10 and 11, the link sl_{21} has $Xci = 2$ and $Tc(sl_{21}) = 12\ Tci$, so for $X = 3$ we have $S = 1$ and $O = 0$. Therefore, the link sl_{21} has to wait for $w_{sl_{21}}(3) = 2Tc - (2-1)Tci = 23Tci$. In fact, looking at the example in Fig. 10, if a packet to be transmitted in sl_{21} arrives when the second CI has just started, it will be transmitted after 11 Tci, i.e., $Tc(sl_{21}) - (Xci - 1)Tci$.

To apply the busy period approach we first calculate the number of packets of the f-th flow generated in any interval of length t, i.e.,

$$I_f(t) = \left\lceil \frac{t}{P_f} \right\rceil. \quad (6)$$

Then, we calculate the largest number (X_j^f) of start times that are needed to transmit the i-th packet, which is given by the smallest value of X_j^f that satisfies the following equation

$$X_j^f = 1 + \sum_{\text{prio}(k)>\text{prio}(j)} I_k(w_i(X_j^f)) + \sum_{\substack{\text{prio}(k)=\text{prio}(j),\\ k\neq j}} 1. \quad (7)$$

In equation (7), 1 is the start time required to transmit the packet, while the parameter X_j encompasses the following sources of interference:

1) The k-th higher priority packets (i.e., $\text{prio}(k) > \text{prio}(j)$) that compete with the j-th packet for the transmission in the i-th link.
2) The packets with the same priority as the j-th packet that compete with it for the transmission in the i-th link (in this case a FIFO policy is adopted).

Equation (7) has not simple solution, as the X_j term appears both in the Left-hand-side (LHS) and in the Right-hand-side (RHS) of the equation under the ceiling operator. Thus, the calculation of X_j is performed by the following iterations

$$\begin{cases} X_j^{f\,(0)} & = 1 \\ X_j^{f\,(v+1)} & = 1 + \sum_{\text{prio}(k)>\text{prio}(j)} I_k(w_i(X_j^{f\,(v)})) \\ & \quad + \sum_{\substack{\text{prio}(k)=\text{prio}(j),\\ k\neq j}} 1. \end{cases} \quad (8)$$

Iteration starts with $X_j^{f\,(v)} = 1$, with $v = 0$, as the packet to be transmitted requires one start time. Then $X_j^{f\,(v)}$ is iteratively calculated until the LHS is equal to the RHS, i.e., until the interference stops growing. Equation (8) is proved to converge if the number of start times required to transmit every packet on the i-th link is lower than or equal to the number of start times available for the i-th link [35].

Once the X_j^f value has been found, the worst-case $Q_t(l_i^{(f)})$ is calculated as

$$Q_t(l_i^{(f)})_{\text{worst}} = w_i(X_j^f). \quad (9)$$

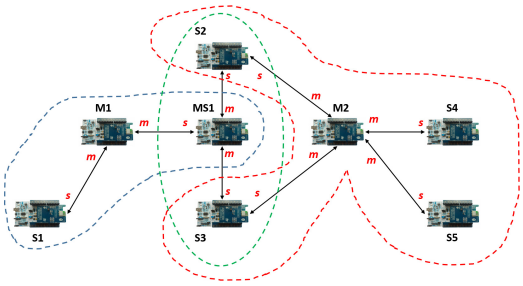

FIGURE 12. The implemented topology.

VI. IMPLEMENTATION OF THE MRT-BLE PROTOCOL ON COTS DEVICES

In order to show the feasibility of the proposed protocol on COTS hardware, we implemented it on the X-NUCLEO-IDB05A1 devices produced by STMicroelectronics. These devices are equipped with communication modules that are compliant with the Bluetooth Specifications v4.1, i.e., the SPBTLE-RF BlueNRG-MS ones.

The testbed is composed of eight devices, as shown in Fig. 12.

Each device acts as both a GATT client and a GATT server. Bidirectional communication consists in transmissions initiated by the GATT servers, which update their exposed attributes and send them through notifications. In the implemented topology, there are three linked sub-networks and the devices S2, S3 and MS1 are shared nodes. As the masters of the linked sub-networks are not synchronized, connection events may overlap at the shared nodes. The mechanism described in Section IV starts after the node MS1 has received a synchronization packet from node M1.

Any time the shared slaves, i.e., S2 and S3, note (following a GATT event) that MS1 has modified one of their exposed attributes (the descriptor of one of their characteristics), they will enable or disable the connections with M2. In particular, if MS1 has enabled the connections with S2 and S3, both of them will disable the connections with M2. Conversely, if MS1 has disabled the connections with S2 and S3, both of them will enable the connections with M2.

Fig. 13 shows the timing of the masters M1, MS1 and M2 in the topology in Fig. 12. The semi-transparent rectangles represent the timeslices during which the depicted connections are disabled.

Fig. 13 shows that the node MS1 in the initial phase enables only the connection with the master M1. Once the sync packet, sent from node M1, is received by the node MS1, the mechanism of enabling/disabling the connections starts. Each device implements as many priority queues as the number of established connections managed by the

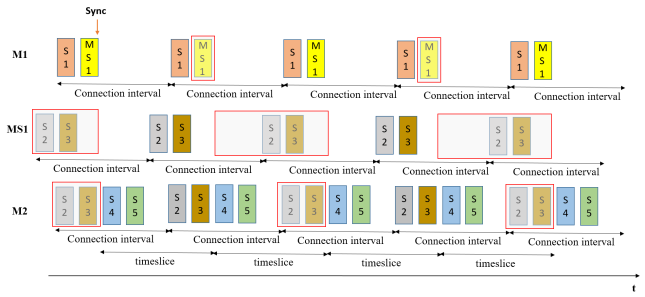

FIGURE 13. Timing of the masters.

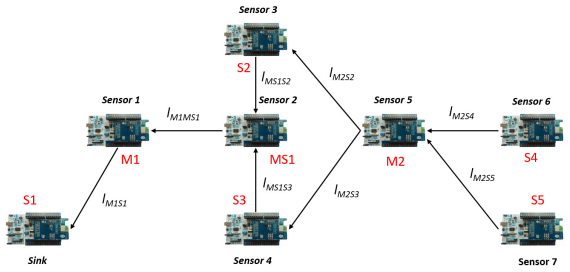

FIGURE 14. The considered scenario.

MRT-BLE sublayer. Each priority queue contains the outbound packets for a specific connection. The insertion into a given queue occurs, using a static routing function, when a packet is generated or when a node receives a packet to forward. A backup path can be provided for each flow. The priority of packets can be assigned according to a configurable criterion. In this scenario we dynamically assigned the priority of packets according to the hop count, i.e., the number of hops that the packet has traversed. This way, we can favor the packets that have to traverse more links to reach the destination, thus avoiding to excessively increase their end-to-end delay. For example, on the node MS1, a packet with source S4 (two hops) has a higher priority than a packet with source M2 (one hop). When, on a device acting as a GATT server, a GATT event reports that a connection has been enabled, the device is allowed to send one packet for each enabled connection every connection interval. Before sending a packet, each device has to check which connection is enabled and then transmits the highest priority packet from the queue of the enabled connection.

The realized testbed, shown in Fig. 14, consists of seven sensors (M1, MS1, S2, S3, M2, S4 and S5) and one sink (S1) that collects the data from all the sensors. Our implementation does not include the application layer of the sink that receives and processes the received data.

The arrows in Fig. 14 show the direction of the data transmission flows. As it can be seen, the nodes M1 and MS1 are a potential bottleneck in the realized mesh network.

In the simulated scenario, the sensors generate a new packet every second. Furthermore, the used devices permit to set the maximum duration of each connection event (*Tce*). For each connection, the duration of the connection interval *Tci* was set to 30ms and the *Tce* was set to 5ms to allow

TABLE 2. Testbed parameters.

Parameter	Value
Tci	30 ms
Tce	5 ms
Xci	4

up to five master connections and one advertising or scanning interval (they are mutually exclusive in a CI). In each connection interval, the connected devices exchange only one packet in their CE. The *Tce* is sized so as to take into account the time to transmit one packet and any possible Link Layer retransmission.

On the final destination (i.e., the actuator S1) a specific check is implemented to verify that no packet is lost.

The following subsections present the results of the measurements performed to assess the feasibility of the MRT-BLE approach on COTS devices and to analyze the application level end-to-end delay experienced by multi-hop real-time data exchanges.

A. PACKET END-TO-END DELAY

This subsection presents the results of experimental measurements of the packet end-to-end (e2e) delay as a function of the number of hops from the source to the final destination. Note that, in the following, the hop count is obtained summing up the number of traversed links and the node S1 is assumed to be the final destination for all the packets.

The relevant network parameters are summarized in Table 2. The *Tce* parameter was set to accommodate the transmission of a packet plus the potential retransmissions, while *Tci* is equal to the duration of five connection events plus the scanning interval. As the calculated $Tsw(i)$ for enabling/disabling the connection is really high (e.g., the *Tsw* for the link sl_{MS1S2} in Fig. 14 is 180ms according Eq. (1)), *Xci* was set to 4 to reduce the network end-to-end delay. In fact, as S2, S3, S4, S5, M2, and MS1 generate one packet with destination S1 every second, the node MS1 should forward six packets to M1 (the only device connected with S1). Consequently, an overhead of 180ms for each packet would cause a large end-to-end delay increase. Setting *Xci* equal to 4 in this case significantly reduces the overhead of the proposed mechanism, i.e., 180ms every four packet transmissions.

We defined seven flows (one from each node, except the sink) generated with a period of 1s and with a 30-byte payload. The parameters of the flows are shown in Table 3. The routing paths determine which packets actually interfere with the packets originated by a given node. The interfence is due to the waiting times in the outbound queues of the intermediate nodes that the packets traverse to reach the final destination.

The assessed metric here adopted is the packet end-to-end delay at the application layer, defined as the time difference

TABLE 3. Parameters of the flows.

Flow ID	Source	Period	Routing Path
0	S5	1s	$l_{M2S5}, sl_{M2S3}, sl_{MS1S3}, sl_{M1MS1}, l_{M1S1}$
1	S4	1s	$l_{M2S4}, sl_{M2S2}, sl_{MS1S2}, sl_{M1MS1}, l_{M1S1}$
2	M2	1s	$sl_{M2S3}, sl_{MS1S3}, sl_{M1MS1}, l_{M1S1}$
3	S3	1s	$sl_{MS1S3}, sl_{M1MS1}, l_{M1S1}$
4	S2	1s	$sl_{MS1S2}, sl_{M1MS1}, l_{M1S1}$
5	MS1	1s	sl_{M1MS1}, l_{M1S1}
6	M1	1s	l_{M1S1}

TABLE 4. WCRT values.

Source	WCRT value
$M1$	240 ms
$MS1$	2010 ms
$S2$	2460 ms
$S3$	2490 ms
$M2$	1770 ms
$S4$	1710 ms
$S5$	1710 ms

between the packet generation time at the source and the reception time at the receiver, measured at the application layer. The end-to-end delay is measured by a third device, hard-wired to both the source and the destination nodes, as the difference between the packet reception time (i.e. when the receiver node sets up a pin) and the packet generation time (i.e., when the source node sets up a pin).

According to the timing analysis in Section V, the worst case response-time (WCRT) values are shown in Table 4. These values represent the upper bound of the expected end-to-end delay for each flow.

Fig. 15 gives the end-to-end delay distribution for the packets generated by node M1 (i.e., single-hop), that shows the percentage of packets that experienced the plotted end-to-end delay values.

As it is shown in Fig. 15, every packet generated by the node M1 was delivered to the destination (S1) within 120ms. The measured delay is only due to the waiting time that a packet experiences in the queue before transmission. Note that the node M1 can transmit a packet every connection interval, as the connection with the node S1 is always enabled. The delay experienced by the packet with source M1 is a multiple of *Tci* (depending on the number of higher priority packets enqueued in the priority queue) plus the waiting time

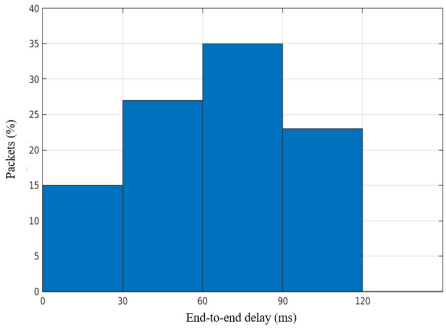

FIGURE 15. End-to-end delay distribution - single hop.

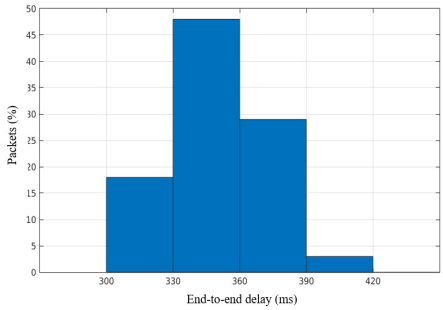

FIGURE 16. End-to-end delay distribution - two hops.

for the assigned Link Layer slot. If both the application queue and BLE queue are empty, the packet will be delivered to destination within 30ms (*Tci*).

The experimental results show that the path of these packets in the considered scenario (i.e., the one in Fig. 14) and the priority policy bound to three connection intervals the waiting time for the Link Layer transmission of the packets originating from M1. Consequently, the end-to-end delay for these packets is lower than 120ms (i.e., 4*Tci).

Fig. 16 shows the end-to-end delay distribution for the packets originating from the node MS1 (i.e., two-hop packets).

Almost all these packets reach the final destination in less than 390ms. Fig. 16 shows that a low percentage of packets (lower than 4%) with source node MS1 reach the final destination within 420ms. This delay mainly consists of the waiting times in the queues due to the interference of higher priority packets, i.e., the packets originating from the other source nodes (i.e., all of them but M1), that accumulated more hops on their way.

Fig. 17 shows the end-to-end delay distribution for the packets with source node S3 (i.e., three-hop packets).

Almost all these packets (i.e., 95% of them) reach the final destination within 810ms. A very limited percentage of packets (about 1%) arrived to destination within about 1170ms.

Fig. 18 shows the end-to-end delay distribution for the packets with source node M2 (i.e., four-hop packets).

Almost all of the packets (about 95%) were delivered to the final destination within 1050ms. An end-to-end delay

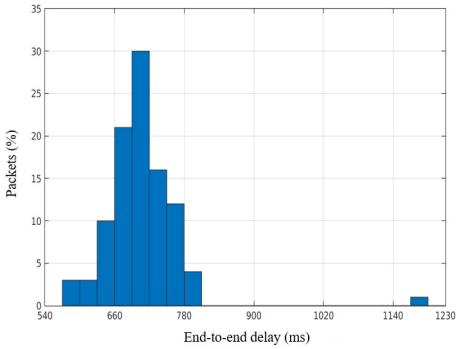

FIGURE 17. End-to-end delay distribution - three hops.

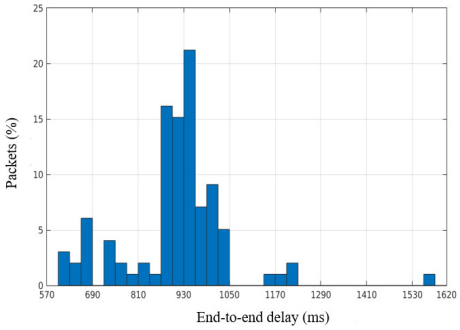

FIGURE 18. End-to-end delay distribution - four hops.

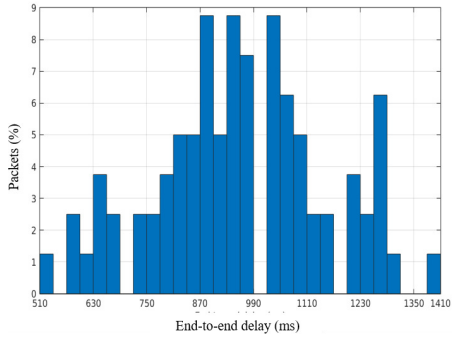

FIGURE 19. End-to-end delay distribution - five hops.

of 1580ms was experienced by a limited percentage of packets (lower than than 2%).

The end-to-end delay peak values shown in Figures 17 and 18 are due to the longer queuing delays that the packets generated by the nodes S3 and M2 experience due to the interference of higher priority packets. More specifically, the interfering packets for the ones generated by the node S3 are those originating from the nodes S5 and M2, while the interfering packets for the node M2 are the ones generated by node S5.

Fig. 19 shows the end-to-end delay distribution for the packets with source node S4. These packets experience a five-hop delay.

These packets arrived at destination within 1400ms. They experienced a waiting time in their source node lower

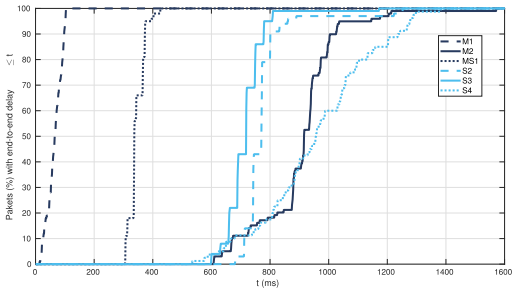

FIGURE 20. Cumulative distribution of the packet end-to-end delay.

than 30ms, as the node S4 is not a forwarding node in the considered scenario and the node M2 always keeps enabled the connections with node S4. Consequently, packet transmission occurs within a connection interval since the packet generation. Similar results were found for the packets generated by the node S5, that have the same hop count and thus the same priority (i.e., the highest in the network) as the packets generated by S4. This is in accordance with the analysis, that obtained the same worst-case response times for the packets originated by S4 and S5.

In all the considered cases, the measured end-to-end delays are always lower than the worst case response-time values shown in Table 4, as the analysis is safe.

Fig. 20 summarizes the cumulative distribution of the packet end-to-end delay versus the source node and, therefore, versus the number of hops between the source and the final destination.

Fig. 20 shows that the end-to-end delays of the packets that have the same hop count to the final destination, for example, the nodes S2 and S3, are centered around a given range of values. In fact, almost all the packets generated by the nodes S2 and S3 reach the destination within about 700 ms. Consequently, the cumulative distributions grow fast to reach a specific range of end-to-end delay values. This consideration is especially true for the packets that at most experience a three-hop delay. For example, the cumulative distribution of end-to-end delay for the packets generated by MS1 grow very fast around 330-360 ms. When the number of hops from the source to the final destination increases, more factors affect the end-to-end delay, therefore some packets will experience a higher variable delay and the cumulative distributions grow more slowly. For instance, the packets generated by the node S4 experience an end-to-end delays distributed between 510 ms and 1410 ms.

VII. CONCLUSIONS

This paper proposes the MRT-BLE, a protocol working on top of BLE that provides for bounded delays over IWSN with mesh topologies. The paper addressed in detail the MRT-BLE protocol and its configuration and implementation on COTS devices. To assess the performance of the protocol, a worst-case timing analysis is also presented and its outcomes are compared with the packet end-to-end delays

obtained in experiments on a real testbed. To the best of our knowledge, in the literature there are no other mechanisms supporting real-time communication over BLE mesh networks. Other approaches addressing mesh topologies over BLE (e.g., Mesh BLE [8]) work in a best-effort way, therefore, they can be more bandwidth-efficient and may also provide better average performance than MRT-BLE, but they cannot temporally isolate time-critical flows and cannot guarantee end-to-end delay bounds to individual flows. As a result, these approaches cannot cope with the real-time requirements of industrial communications. For this reason, no meaningful performance comparison between MRT-BLE and such approaches can be made for the purposes of this paper. The MRT-BLE protocol is a connection-oriented protocol, as the main aim here was on ensuring a bounded packet delivery delay over multihop mesh IWSN. The protocol therefore foresees that the network topology is offline configured. This feature limits the applicability of MRT-BLE to IWSN that are not characterized by the presence of mobile nodes. Future works will therefore investigate a dynamic configuration mechanism to allow for both more flexibility and the free movement of nodes. Another line of investigation includes a dynamic topology management mechanism, as the one proposed in [38], and a stochastic response-time analysis able to take into account also aperiodic traffic flows.

REFERENCES

[1] *Bluetooth Core Specification Version 4.1*, Bluetooth SIG, Kirkland, WA, USA, Dec. 2013.

[2] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is Bluetooth low energy? Comparative measurements with ZigBee/802.15.4," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2012, pp. 232–237.

[3] E. Toscano and L. L. Bello, "Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios," in *Proc. 9th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, May 2012, pp. 115–124.

[4] R. Rondon, M. Gidlund, and K. Landernas, "Evaluating Bluetooth low energy suitability for time-critical industrial IoT applications," *Int. J. Wireless Inf. Netw.*, vol. 24, no. 3, pp. 278–290, Sep. 2017.

[5] *Bluetooth Core Specification Version 4.2*, Bluetooth SIG, Kirkland, WA, USA, Dec. 2014.

[6] *Bluetooth Core Specification Version 5.0*, Bluetooth SIG, Kirkland, WA, USA, Dec. 2016.

[7] G. Patti, L. Leonardi, and L. Lo Bello, "A Bluetooth Low Energy real-time protocol for industrial wireless mesh networks," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Firenze, Italy, Oct. 2016.

[8] *Mesh Profile Specifications 1.0*, Bluetooth SIG, Kirkland, WA, USA, Jul. 2017.

[9] M. Collotta and G. Pau, "A solution based on Bluetooth low energy for smart home energy management," *Energies*, vol. 8, no. 10, pp. 11916–11938, 2015.

[10] M. Collotta and G. Pau, "A novel energy management approach for smart homes using Bluetooth low energy," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2988–2996, Dec. 2015.

[11] L. Lo Bello, O. Mirabella, and N. Torrisi, "Modelling and evaluating traceability systems in food manufacturing chains," in *Proc. 13th IEEE Int. Workshops Enabling Technol., Infrastruct. Collaborat. Enterprises*, Jun. 2004, pp. 173–179.

[12] B. B. Sánchez, R. Alcarria, D. Martín, and T. Robles, "TF4SM: A framework for developing traceability solutions in small manufacturing companies," *Sensors*, vol. 15, no. 11, pp. 29478–29510, 2015. [Online]. Available: http://www.mdpi.com/1424-8220/15/11/29478

[13] J. Medina-García, T. Sánchez-Rodríguez, J. A. G. Galán, A. Delgado, F. Gómez-Bravo, and R. Jiménez, "A wireless sensor system for real-time monitoring and fault detection of motor arrays," *Sensors*, vol. 17, no. 3, p. 469, 2017.

[14] L. Leonardi, G. Patti, F. Battaglia, and L. Lo Bello, "Simulative assessments of the IEEE 802.15.4 CSMA/CA with priority channel access in structural health monitoring scenarios," in *Proc. IEEE 15th Int. Conf. Ind. Informat. (INDIN)*, Emden, Germany, Jul. 2017, pp. 375–380.

[15] P. Trelsmo, P. Di Marco, P. Skillermark, R. Chirikov, and J. Östman, "Evaluating IPv6 connectivity for IEEE 802.15.4 and Bluetooth low energy," in *Proc. Wireless Commun. Netw. Conf. Workshops (WCNCW)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.

[16] R. Natarajan, P. Zand, and M. Nabi, "Analysis of coexistence between IEEE 802.15. 4, BLE and IEEE 802.11 in the 2.4 GHz ISM band," in *Proc. 42nd Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Firenze, Italy, Oct. 2016, pp. 6025–6032.

[17] M. H. Dwijaksara, W. S. Jeon, and D. G. Jeong, "A channel access scheme for Bluetooth low energy to support delay-sensitive applications," in *Proc. 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Valencia, Spain, Sep. 2016, pp. 1–6.

[18] K. Cho, G. Park, W. Cho, J. Seo, and K. Han, "Performance analysis of device discovery of Bluetooth low energy (BLE) networks," *Comput. Commun.*, vol. 81, pp. 72–85, May 2016.

[19] J. Seo, K. Cho, W. Cho, G. Park, and K. Han, "A discovery scheme based on carrier sensing in self-organizing Bluetooth low energy networks," *J. Netw. Comput. Appl.*, vol. 65, pp. 72–83, Apr. 2016.

[20] M. Marinoni, A. Biondi, P. Buonocunto, G. Franchino, D. Cesarini, and G. Buttazzo, "Real-time analysis and design of a dual protocol support for Bluetooth LE devices," *IEEE Trans. Ind. Informat.*, vol. 13, no. 1, pp. 80–91, Feb. 2017.

[21] J. Kim, S. Kang, and J. Park, "Bluetooth-based tree topology network for wireless industrial applications," in *Proc. 15th Int. Conf. Control, Autom. System*, Busan, South Korea, Oct. 2015, pp. 1305–1308.

[22] C. Jung, K. Kim, J. Seo, B. N. Silva, and K. Han, "Topology configuration and multihop routing protocol for Bluetooth low energy networks," *IEEE Access*, vol. 5, pp. 9587–9598, May 2017.

[23] P. Zenker, S. Krug, M. Binhack, and J. Seitz, "Evaluation of BLE Mesh capabilities: A case study based on CSRMesh," in *Proc. 8th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Vienna, Austria, Jul. 2016, pp. 790–795.

[24] D. Hortelano, T. Olivares, M. C. Ruiz, C. Garrido-Hidalgo, and V. López, "From sensor networks to Internet of Things. Bluetooth low energy, a standard for this evolution," *Sensors*, vol. 17, no. 2, p. 372, 2017.

[25] *Qualcomm Bluetooth Low Energy Solutions, CSRmesh and CSR1010 (Datasheet 87-CE855-1 Rev A)*, Qualcomm Int., San Diego, CA, USA, Sep. 2016.

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

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

[28] T. Zhang, J. Lu, F. Hu, and Q. Hao, "Bluetooth low energy for wearable sensor-based healthcare systems," in *Proc. IEEE Healthcare Innov. Conf. (HIC)*, Oct. 2014, pp. 251–254.

[29] G. Pau, M. Collotta, and V. Maniscalco, "Bluetooth 5 energy management through a Fuzzy-PSO solution for mobile devices of Internet of Things," *Energies*, vol. 10, no. 7, p. 992, 2017.

[30] R. Rondón, K. Landernäs, and M. Gidlund, "An analytical model of the effective delay performance for Bluetooth low energy," in *Proc. 27th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Valencia, Spain, Sep. 2016, pp. 1–6.

[31] M. Grover, S. K. Pardeshi, N. Singh, and S. Kumar, "Bluetooth low energy for industrial automation," in *Proc. 2nd Int. Conf. Electron. Commun. Syst. (ICECS)*, Coimbatore, India, Feb. 2015, pp. 512–515.

[32] *BlueNRG, BlueNRG-MS Stacks Programming Guidelines (Programming Manual PM0237)*, STMicroelectronics, Geneva, Switzerland, Dec. 2016.

[33] *BlueNRG-MS Development Kits (User Manual UM1870)*, STMicroelectronics, Geneva, Switzerland, May 2016.

[34] G. A. Kaczynski, L. Lo Bello, and T. Nolte, "Deriving exact stochastic response times of periodic tasks in hybrid priority-driven soft real-time systems," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Patras, Greece, Sep. 2007, pp. 101–110.

[35] M. Joseph and P. K. Pandya, "Finding response times in a real-time system," *Comput. J.*, vol. 29, no. 5, pp. 390–395, Oct. 1986.
[36] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proc. IEEE Real-time Syst. Symp. (RTSS)*, Lake Buena Vista, FL, USA, Dec. 1990, pp. 201–209.
[37] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Syst.*, vol. 35, pp. 239–272, Jan. 2007.
[38] E. Toscano and L. L. Bello, "A topology management protocol with bounded delay for Wireless Sensor Networks," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2008, pp. 942–951.

LUCA LEONARDI received the bachelor's and master's degrees *(summa cum laude)* from the University of Catania in 2013 and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Department of Systems, Energy, Computer and Telecommunications Engineering. Since 2015, he has been focusing in the field of real-time industrial networks, wireless sensor and actuator networks, and industrial Internet of Things.

GAETANO PATTI received the M.S. and Ph.D. degrees from the University of Catania, Italy, in 2013 and 2017, respectively. He is currently a Post-Doctoral Researcher with the Department of Electrical, Electronics, and Computer Engineering, University of Catania.

He has co-authored over 20 papers published in the proceedings of international conferences and in important international journals. Since 2012, he has been focusing in the field of real-time industrial networks, automotive networks, wireless sensor and actuator networks, powerline communications and networks for mobile robot applications. He is a member of the IEEE IES Technical Committee on Factory Automation (Subcommittee on In-Vehicle Embedded Systems).

LUCIA LO BELLO (M'02–SM'09) received the M.S. degree in electronic engineering and the Ph.D. degree in computer engineering from the University of Catania, Italy, in 1994 and 1998, respectively. During 2000–2001, she was a Visiting Researcher with the Department of Computer Engineering, Seoul National University, South Korea. In 2014, she joined the University of Malardalen, Sweden, as a Guest Professor. She is currently a tenured Associate Professor with the Department of Electrical, Electronic and Computer Engineering, University of Catania. She has authored or co-authored over 150 technical papers in the area of industrial networks, automotive communications, real-time embedded systems, and wireless sensor networks. She is responsible for several international and national projects in the area of real-time embedded systems and networks. She was the Chair of the IES Technical Committee on Factory Automation from 2014 to 2015 and from 2016 to 2017.