Real-Time Analysis and Design of a Dual Protocol Support for Bluetooth LE Devices

Mauro Marinoni* (*IEEE Member*), Alessandro Biondi* (*IEEE Student Member*), Pasquale Buonocunto*, Gianluca Franchino*, Daniel Cesarini†*, Giorgio Buttazzo* (*IEEE Fellow*)
*Scuola Superiore Sant'Anna, Pisa, Italy - email: {*name.surname*}@*sssup.it*
†Tridonic GmbH, Dornbirn, Austria - email: {*daniel.cesarini*}@*tridonic.com*

***Abstract*—Modern distributed embedded systems frequently involve wireless communication nodes where messages have to be delivered within given timing constraints. This goal can be achieved by adopting a suitable real-time communication protocol. In addition, connecting such systems with mobile devices is also desirable for performing configuration, monitoring and maintenance activities. The Bluetooth Low Energy (BLE) protocol would be an attractive solution for this purpose, because it is supported by consumer devices, such as tablets and smart phones, for implementing personal area networks with reduced energy consumption. Unfortunately, however, it cannot guarantee a bounded delay for managing real-time traffic. Modern BLE radio transceivers allow partitioning the network bandwidth between the BLE protocol and another user-defined protocol running on top of the raw radio. This paper exploits this feature to provide an analysis and a design methodology to guarantee the feasibility of a real-time custom protocol that shares the radio with the BLE. Experimental results on a Nordic reference platform show the feasibility of the dual-protocol approach and its capability to support a custom real-time protocol on the raw radio with a bounded overhead.**

***Index Terms*—Real-time Systems, Personal Area Network, Bluetooth, Dual-protocol.**

I. Introduction

Several mobile applications require communication with bounded delay, that is, real-time communication [1]. For example, a team of cooperating mobile robots form a distributed network in which nodes, in order to coordinate and take common decisions, have to continuously exchange messages to converge to a common decision in a bounded amount of time [2]. Relative localization of mobile nodes is another activity that also implies a time constrained communication among nodes to fuse local sensory information available on each node into a shared global view of the system. For instance, a set of data based on the received signal strength indicator (RSSI) acquired by each node can be converted into node positions relative to a common reference frame [3]. A similar need arises in some modern distributed control systems where the stringent timing constraints coming from the control application must coexist with the use of wireless connectivity to overcome the typical drawbacks of wired solutions (e.g., cost, weight, and encumbrance).

This work has been partially supported by Telecom Italia.

For the sake of configuration, operator interface and logging, it is desirable to have compatibility with general purpose mobile devices, such as smart phones and tablets. However, the protocols used by the latter do not support the real-time requirements of the former.

For instance, low-power communication protocols, as BLE [4] or ANT+ [5], represent a possible solution for the user interaction, because they are supported by most mobile devices, as tablets and smart phones, and also enable longer battery lifetime. However, these protocols were not designed for real-time traffic and cannot be used when a bounded transmission time is required. Moreover, extending them with a real-time support can be quite difficult or even impossible without losing compatibility with the standard. Also, most of these stacks for embedded devices are provided by the hardware producer as binary libraries, preventing any modification.

On the other hand, many protocols proposed in the literature (see Section II) for real-time traffic in wireless sensor networks are not supported by commercial mobile devices and do not represent an attractive solution.

The naive solution to this problem is the use of two radio transceivers on each node, one for the real-time traffic and one for the user interaction. Such a naive solution, however, requires more space, cost, and energy consumption, and could be not compatible with the design constraints in some specific systems. The current topology supported by the BLE (rev. 4.0) is seen as a strong limitation. In fact, the Bluetooth Special Interest Group (SIG) is working to incorporate mesh networking into the future BLE specification to allow this kind of topology into the standard. Without such a mesh support, two BLE nodes would communicate through the master, thus introducing extra significant delay and packet overhead. In addition, the communication behavior would strongly depend on the BLE stack of the master, whose connection parameters (e.g., the minimum connection period) are specific for different operating systems. As a consequence, an application-level approach would be platform dependent, poorly scalable, not predictable, and hence not suitable for supporting real-time messages.

In order to already provide working solutions to the market, some vendors started proposing proprietary dual-protocol approaches based on sharing the bandwidth between the standard BLE stack and the transmission using a custom-made protocol implemented on top of the raw radio. For instance, the CSR company proposes a proprietary extension [6] to their BLE

devices that allows them not only to receive and act upon messages, but also repeat those messages to surrounding devices, so extending the range of standard BLE and providing a simple ad-hoc mesh network for Internet of Things applications. Another similar solution is Wirepas Pino [7], which is a fully automatic, self-optimizing, multi-hop mesh networking protocol stack built on top of the Timeslot application programming interface (API) provided by the Nordic's nRF51822 Bluetooth chip. However, none of these solutions consider the problem of providing guarantees for messages delivery times.

To combine the advantages of the dual-protocol devices together with the capability of providing real-time communication, this paper proposes the design and the analysis of a bandwidth reservation mechanism to share the available network bandwidth between the BLE connectivity and a custom protocol dedicated to real-time communication. Such a mechanism exploits the capability of some modern radio transceivers to provide a time slot for raw transmission upon user request. To the best of our knowledge, this is the first approach in which the BLE is integrated in a dual-protocol setting to support real-time traffic.

The proposed dual protocol mechanism can also be exploited in body area networks (BANs) for mobile health applications [8] whenever nodes need to exchange messages among them (e.g., for synchronizing sensory data acquisition to increase precision), while interacting with the user through a mobile device.

Contributions. This paper has three main novel contributions. First, it presents a bandwidth sharing mechanism that allows reserving a fraction of the available network bandwidth to the BLE and a fraction to another user-defined protocol built on top of a meta-protocol that regulates the access to the radio transceiver. Second, it presents a delay analysis that can be used to guarantee real-time traffic over the network, while ensuring no buffer overflow in the BLE communication, assuming that BLE packets are generated in a periodic/sporadic fashion. Third, a design method is presented to determine the maximum bandwidth allocated to the custom protocol for preventing the overflow of the BLE buffer. Experimental results are also presented on a platform based on a Nordic device including both the computation unit and the transceiver, which allows the development of a dual-protocol solution on a single chip.

The rest of the paper is organized as follows. Section II presents the state of the art on the key aspects addressed in the paper. Section III illustrates the system architecture and the related models. Section IV presents the foundations needed to support the proposed dual-protocol approach. Section V presents the analysis for guaranteeing the real-time traffic, the design of the proposed meta-protocol, and the analysis to guarantee the absence of packet loss in the BLE communication. Section VI presents a design method for determining the maximum bandwidth that can be allocated to the real-time traffic preventing the BLE buffer overflow. Section VII reports some experimental results carried out on the system to better characterize its temporal behavior. Finally, Section VIII states our conclusions and future work.

II. State of the art

Classic Bluetooth was developed with the aim of replacing cables on desktop computers in connecting devices such as printers, mobile phones, etc. It covers distances from $1\ \mathrm{m}$ to $100\ \mathrm{m}$, depending on the radio class used in the implementation, with a bit rate of $0.7 - 2.1\ \mathrm{Mbit/s}$. The maximum number of slaves is 7 and the peak current consumption is approximately $30\ \mathrm{mA}$. During the last years, it has been extended into Bluetooth Low Energy (BLE), having a lower application throughput of $0.27\ \mathrm{Mbit/s}$, but supporting a higher number of concurrent communication slaves with lower power consumption. The main difference with respect to the classic Bluetooth is the communication paradigm: while the classic Bluetooth actively keep alive the connection, thus guaranteeing the data transfer between two nodes, the Low Energy version does not, and if needed, it should be implemented at application-level. With respect to Classic Bluetooth (Basis Rate, BR, and Enhanced Data Rate, EDR), the BLE has a much smaller memory requirement in the microcontroller stack [9].

There exist other communication stacks that can be taken into account to develop BAN applications [10], [11]. For instance, ZigBee is a protocol based on the IEEE 802.15.4 standard [12], which can form a mesh network with transmission distances up to $100\ \mathrm{m}$. It supports secure networking and has a defined rate of $250\ \mathrm{kbit/s}$. At present, however, it is not natively supported by mobile phones. ANT+ [5] is an open access multicast wireless sensor network technology that has a negligible consumption when operating in low-power "sleep" mode, and a higher one during transmission. It is currently supported by some mobile phones and is broadly used for sport and well-being devices (e.g., body weight scales and step-counters). It is interesting to observe that Dementyev et al. [13] proved that, for some specific scenarios, the BLE is more energy efficient than ZigBee and ANT+. In addition, the BLE has the advantage of being more widespread in mobile phones, and some Classic Bluetooth devices can be updated to BLE by replacing the software in the chip [14].

On a different side, to guarantee a timely communication at the medium access protocol (MAC) layer in wireless networks, several solutions have been proposed in the literature, as wireless fieldbus protocols [15], [16]. Some other authors proposed to control the channel access by a real-time scheduling algorithm to prevent conflicts in transmission and guarantee message deadlines [17], [18]. Other authors proposed solutions for guaranteeing real-time communication and energy efficiency on IEEE 802.15.4 networks [19], [20].

The development of dual protocols was made possible by new transceivers allowing the co-existence of multiple communication stacks. Examples are the multimode IEEE 802.15.6/Bluetooth Low-Energy WBAN transceiver for biotelemetry applications proposed in [21] and the WiZi-Cloud, an application-transparent dual ZigBee-WiFi radio [22]. Another example is represented by the Nordic nRF51822 [23] device, that supports multiple protocols, such as ANT+ and BLE, other than providing support for custom protocols.

Other industrial projects proposed solutions to integrate multiple protocols in the same transmission device. Zehavi et al. [24] developed a wireless transceiver capable of a joint signal transmit/receive section and a number of signal up/down conversion sections to transmit and receive signals in accordance with two alternate protocols. In some of the developed devices, the transceiver is provided with a processor programmed to implement a time sharing schedule to facilitate the coordinated control and performance of transmit and receive operations. Protocols can be selected among Bluetooth, 802.11x, and Home RF. The wireless device can also be used as a master or gateway of two wireless networks. Bridgelall [25] proposed a time sharing mechanism that allows alternating Bluetooth BR with 802.11. Time frames are reserved for each protocol by exploiting the 802.11 Request-To-Send mechanism to accommodate for Bluetooth transmission windows. Finally, Desai et al. [26] proposed a dual-mode BLE device that identifies idle intervals within Bluetooth BR/EDR traffic communication and uses the dual-mode BLE device to concurrently perform various BLE activities. For example, advertising packet transmissions may be concurrently performed inside identified idle intervals within the Bluetooth BR/EDR traffic communication. However, all these products available on the market aim at integrating standard and proprietary protocols to improve the interoperability among the nodes. They share the bandwidth with the only constraint of maintaining the compatibility with the standard of the implemented protocols (e.g., avoid timeouts, maintain the connection status), but none of them provides the feature of guaranteeing timing constraints on the message delivery time.

Instead, this paper presents an approach sharing the network bandwidth between the BLE and a custom protocol which is able to provide guarantees on timing constraints needed by a real-time protocol.

III. System Architecture and Model

We consider an architecture where each node is composed of an elaboration unit and a radio device. The access to the radio device on each node is regulated by a software component referred to as *radio manager* (RM). Each node is equipped with an implementation of the BLE stack to undertake BLE communications. As enabled by modern BLE-enabled devices (e.g., such as [23]), we also assume that the RM provides a *bandwidth reservation mechanism* for the radio, allowing reserving specific time slots to be used in raw mode, interleaved with the BLE communication.

The rest of this section presents the network topology for both BLE and real-time traffic, summarizes the main characteristics of the BLE stack, and introduces a model for real-time traffic. Such a model will then be used in Section V to instantiate an example of analysis for a specific real-time protocol (namely Weighted Round-Robin [18]).

A. Network Model

This paper considers a network infrastructure composed of n nodes, $N_1, N_2, \ldots, N_n$, providing best-effort traffic through the BLE protocol [4] and delivering messages subject to timing constraints under a real-time custom protocol.

The real-time protocol supports a fully connected network allowing point-to-point communication between any pair of nodes. These links are used by the application to exchange real-time data with timing constraints. This network configuration is typical in most existing real-time protocols, like those presented in Section II. On the other hand, the BLE protocol is used by each node to communicate with a single external device (e.g., an Android tablet), working as a master for the BLE network. This configuration is the simplest BLE network that can be formed and it is referred to as Piconet. This connection allows nodes to send information to the mobile device for configuration, monitoring and logging activities. Each node N_i can switch its network protocol between the BLE and the custom one. It is assumed that the master node is a commercial mobile device that can only communicate by BLE, without providing support for the implementation of a user-defined protocol. Figure 1 shows an example of network composed by 4 wireless nodes and a master device; note that real-time links among nodes and BLE connections to the master device are denoted by different line styles.

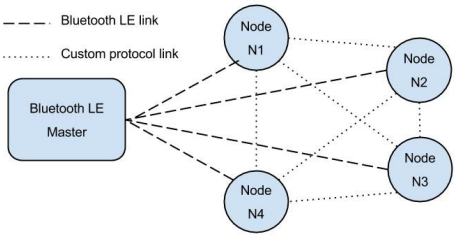

Figure 1: Example of a dual-protocol network with one master and 4 slaves.

Although the BLE protocol is quite flexible in terms of network formation, it is not able to provide support for timing guarantees, and the capabilities to perform multi-hop communication has been introduced only since version 4.1. Moreover, not all devices provide an implementation of the full standard to reduce footprint and power consumption. Typically, the companies producing the transceivers do not provide the code for the network stack, but only a library in binary format (e.g., Nordic SoftDevice [23]) that works as a black box. Although the BLE stack cannot be modified to better integrate real-time communication, the knowledge of the BLE behavior allows providing a tighter analysis. For this reason, the behavior of the BLE protocol is summarized in the next section.

B. BLE protocol and stack

When a BLE node is powered on, it performs the protocol initialization and starts sending the notification packet on a set of dedicated radio channels, listing the node characteristics (e.g., name, type, available data). When the master node (e.g., a tablet) starts acquiring the information provided by the node, it performs a connection sequence sending the communication parameters, including the period between two consecutive

transmissions (T_B) and the maximum number of packets (n_B) that can be sent in each dispatch, called *Radio Event*.

Each sensor node executes a sporadic task τ_S, that is a computational activity whose consecutive activations are separated by a minimum interval T_S. At each activation the task produces a message of n_S packets to be dispatched by the BLE protocol. The BLE stack provides a hardware buffer of size n_H packets to store data produced by the application. The number of packets waiting to be delivered (backlog) is denoted as n_p and to avoid packet loss it cannot exceed the buffer size, that is $n_p \leq n_H$. At the beginning of each Radio Event, an interval of time of length t_{prep} is needed to get all data packets available in the hardware buffer and prepare them for transmission. When ready, each packet is sent within a transmission time t_p.

Figure 2 illustrates the executions of task τ_S. The number of packets (n_p) in the node waiting to be transmitted (backlog) is represented in the second line. The dashed line indicates the maximum backlog (n_H) that can be managed to avoid packet loss. The last line shows the BLE activities performed on the node to complete the transmissions.

C. Real-time traffic model

The dual-protocol approach is suitable for the integration of a wide spectrum of real-time communication protocols. However, a specific traffic model and a real-time protocol are selected here to show the effects of the reservation mechanism on the analyses of the real-time traffic and on the design phase.

As far as the real-time traffic is concerned, each node N_i generates a stream S_i of periodic messages defined by three parameters: the maximum message length n_i (expressed in number of packets), the message generation period T_i and its relative deadline D_i, both expressed in units of time. In this paper deadlines are assumed to be equal to periods. For the purpose of the analysis, the message length will be expressed as the time M_i used by the radio transceiver to send the message on a free transmission channel without any contention.

If a message of stream S_i is transmitted through n_i packets and T_{pkt} is the time needed to transmit a single packet and receive the associated acknowledge, then M_i can be computed as follows:

$$M_i = n_i T_{pkt}. \tag{1}$$

The bandwidth utilization required to transmit a message is denoted as $U_i = M_i/T_i$. Considering the low reliability of the wireless link, each packet is encoded by an appropriate Forward Error Correction (FEC) mechanism [27]. In real-time networks, this method is usually preferred to packet retransmissions due to its higher predictability [28].

IV. Bandwidth reservation mechanism

The proposed bandwidth reservation approach is built on top of the slot request mechanism offered by the RM and provides the following features:

- **BLE communication** capability for each node in the network ensuring no packet loss;
- **Real-time communication** among the nodes based on a *real-time meta-protocol* that guarantees *synchronized* time slots of given bandwidth.

The real-time meta-protocol is not directly capable of supporting real-time communication and serves as an intermediate layer to support a real-time communication protocol within the proposed framework. The specific protocol for managing the real-time communication is a free user choice, under the assumption that it is able to work on top of a known sequence of time intervals. Figure 3 illustrates the layer-based architecture of the proposed dual-protocol approach.

While the BLE communication relies on the BLE stack as it is, supporting the real-time meta-protocol requires the following baseline mechanisms: (i) *reservation of the radio device*, needed to actually reserve the time slots and (ii) *clock synchronization*, which is fundamental to align in time the slots among the nodes in the network. Such mechanisms are discussed in details in the two following subsections.

A. Radio reservation

To share the channel between two protocols, a reservation mechanism is used to allocate time slots according their bandwidth requirements. Note that the slot allocation can follow an arbitrary complex sporadic sequence to fit a specific bandwidth demand generated by the real-time application. To simplify the analysis and the corresponding presentation, this paper assumes that slots are periodically allocated reserving Q time units every P time units. Thus, P denotes the *period* of the real-time communication protocol, while Q denotes the total time *budget* reserved to the real-time traffic in every period.

The periodic sequence considered in the paper is just one of the possible request sequences for which the proposed analysis has been derived.

As discussed in Section III, the RM provides a mechanism for using the radio device in raw mode in specific time slots, interleaved with the BLE traffic. However, such time slots are not provided as soon as they are requested. In fact, in BLE-enabled devices the radio is typically managed by the BLE stack, which needs to complete the ongoing BLE activity—to leave the BLE protocol in a consistent state—before releasing the radio device for raw-mode usage. For instance, Nordic NRF devices explicitly notify this limitation in their application notes [29].

This behavior of the BLE stack determines that *reservation slots are provided with a variable delay* δ, which depends on the current state of the BLE communication.

An upper-bound δ_{max} on the delay δ can be expressed as a function of the number of packets n_B in any BLE radio event as follows:

$$\delta_{max} = t_{prep} + n_B\, t_p + \sigma_{BR} \tag{2}$$

where t_{prep} is the time needed to prepare the BLE packets (i.e., to pack the application data into the specific packets required by the BLE standard), $n_B\, t_p$ is the time to transmit them, and σ_{BR} is the switching time needed to properly save the BLE state and prepare the radio for the raw communication. Once the requested slot terminates, the BLE stack re-acquires the

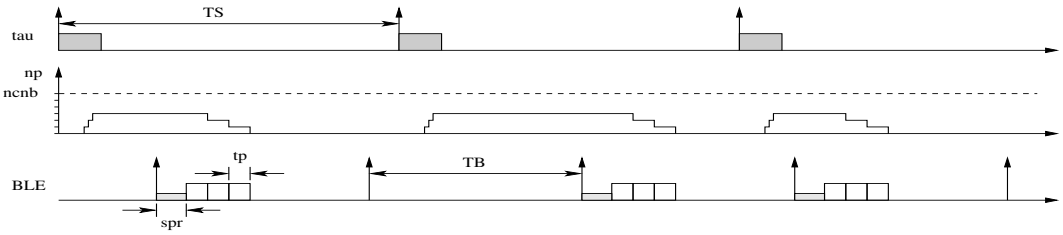

Figure 2: Example of transmission sequence with the BLE protocol.

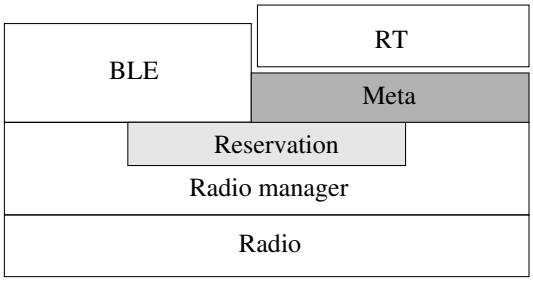

Figure 3: Architecture of the proposed dual-protocol approach.

radio device after a switching time σ_{RB} needed to restore the radio status to support the BLE communication.

Figure 4 illustrates the timing behavior of the radio device under the dual-protocol approach. To better highlight the delay induced by the real-time traffic on the BLE traffic generated in the node, the last line (RT) represents the overall time interval assigned to all the nodes for running the real-time protocol. Note that, in this specific example, the presence of the second protocol affects the BLE communication by saturating the maximum allowed backlog: the conditions under which the BLE communication does not incur in over-backlog will be formally analyzed in Section V-D. In the last time line denoted as RT, the figure also shows two instances of the periodic communication requesting the custom slot, separated by a period P: the first request arrives during the BLE transmission and experiences a delay $\delta < \delta_{max}$, while the second one arrives at the beginning of the BLE transmission and experiences a delay equal to δ_{max}. Also observe that, although the raw radio custom slot can be granted after a delay $\delta < \delta_{max}$, the real-time transmission is started after δ_{max} to make sure that the custom slot is granted by all the nodes in the network.

B. Synchronization

To support the real-time communication, nodes must share a common notion of time to align the communication occurring within the period slots. A crucial issue that must be addressed to obtain a robust dual-protocol communication framework is to bound the differences between local time references inside each node due to the clock drift. Such a drift is due to several causes, including the poor quality of clock generators, possible variations in the supplied voltage, and changes of the environment temperature. For a given clock source, the maximum clock drift rate ξ_{sync} is expressed in parts per million.

To mitigate the clock drift in groups of nodes, a lot of synchronization protocols have been proposed in the literature [30]. In this paper, a simple synchronization protocol has been considered and implemented with the objective of providing a prototype for the dual-protocol system able to validate the proposed analysis and design. The analysis assumes that the synchronization protocol is periodically executed inside the time slot allocated to the raw communication and requires Q_{sync} time units every P_{sync} time units. If ϵ_0^{max} denotes the clock error at the synchronization time, the timing error $\epsilon(\Delta t)$ after a generic interval $\Delta t < P_{sync}$ can be bounded by

$$\epsilon(\Delta t) = \epsilon_0^{max} + \xi_{sync}\Delta t. \tag{3}$$

To safely handle this drift, a guard interval must be inserted at the beginning and at the end of the slot Q. The length of such a guard interval must take into account that two communicating nodes can drift in opposite directions and can be computed as

$$\sigma_{sync} = \epsilon_0^{max} + 2\xi_{sync}P_{sync}. \tag{4}$$

C. System initialization

To correctly boot the system in a dual-protocol mode, the nodes perform the following initialization procedure:

1) nodes start using the radio in raw mode and self-organize in a cluster by executing a network formation procedure [31];
2) the synchronization protocol is executed to align the internal time references of the nodes, still exploiting the transceivers in raw mode;
3) a common time t_0 at which the first real-time slot is requested by all nodes is defined as $t_0 = t + T_{ini}$, where t is the current time and T_{ini} is an interval sufficient for carrying out the operations in the next two phases;
4) the raw communication is disabled, allowing the BLE protocol to perform the initialization phase in which the Piconet is formed;
5) nodes start sending their data to the external device using the BLE protocol;

Table I summarizes all the variables used throughout the paper.

V. Dual-protocol Design and Analysis

This section explains how to guarantee the real-time traffic upon the proposed dual-protocol approach. It begins by deriving the conditions under which it is possible to ensure that a time budget Q is allocated for real-time communication every

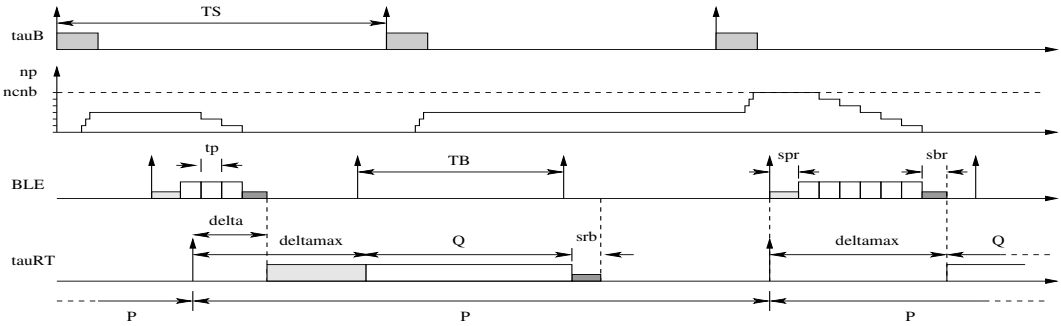

Figure 4: Example of a periodic real-time slot request within the BLE communication.

Variable	Description
T_S	Minimum interarrival time of task τ_S
n_S	BLE packets produced by each job
T_B	Interval between two consecutive Radio Events
n_B	Maximum number of packets delivered in each Radio Event
t_{prep}	Time needed to prepare packets for transmission
t_p	Packet transmission time
n_p	Number of packets in the node waiting to be transmitted
n_H	Size of the hardware transmission buffer
σ_{BR}	Switching time from the BLE stack to real-time
σ_{RB}	Switching time from real-time to the BLE stack
δ	Delay between the slot request and the actual availability
δ_{max}	Maximum delay δ
P	Request period for the real-time slots
Q	Budget assigned to each real-time slot
σ_{sync}	Safe guards intervals due to clock synchronization
P_{sync}	Synchronization period
Q_{sync}	Synchronization bandwidth allocation

Table I: List of all variables used in the model.

period P *for all the nodes in the network* (Section V-A). Based on such an analysis, Section V-B presents the design of the real-time meta-protocol introduced in Section IV.

Please note that the proposed approach is agnostic about the specific real-time communication protocol implemented on top of the meta-protocol. Hence, existing analysis techniques (e.g., [32], [33]) can be adopted to verify the feasibility of real-time traffic once the worst-case behavior of the metal-protocol has been guaranteed.

To illustrate a specific example, Section V-C reports the analysis for guaranteeing real-time traffic using the Weighted Round-Robin (WRR) algorithm on top of the dual-protocol approach.

While timing properties can be guaranteed for real-time communication, the BLE communication is not subjected to temporal constraints. However, Section V-D analyzes the conditions under which the real-time communication does not cause packet loss for the BLE communication due to over-backlog.

A. Guaranteeing real-time slots

As described in Section III, bandwidth partitioning is enforced by a reservation mechanism that provides a time *budget* Q every *period* P for accessing the radio device in real-time. Hence, the bandwidth reserved to the real-time traffic is

$$U^{\text{RT}} = \frac{Q}{P}. \tag{5}$$

However, to actually guarantee the bandwidth utilization U^{RT} for the whole network it is required that *all* the nodes in the network are provided of *at least* Q time units every period P in a *synchronous* manner. Multiple phenomena must be taken into account to correctly ensure this condition.

As discussed in Section IV-A, we recall that, once time slots are requested to the RM, nodes may incur in a variable delay δ before actually accessing the radio device. An additional delay can also be introduced by the synchronization error ϵ in the local clocks of the nodes. The major problem with such delays is that the *actual time interval* in which the nodes have their slots overlapped in time can be lower than the requested slot length Q, hence violating the overall bandwidth requirement of the network.

The reasoning explained above leads to a conclusion that a slot length larger than Q must be requested to the RM to guarantee that all the slots are overlapped in time for at least Q time units. The minimum slot length to be requested to guarantee an overlap of at least Q time units is defined by the following lemma.

Lemma V.1. *To guarantee that all the slots for real-time communication overlap for at least Q units of time, each node must request a slot with length $\widehat{Q}$, where*

$$\widehat{Q} = Q + \Theta = Q + \delta_{max} + 2\sigma_{sync} + \sigma_{RB}. \tag{6}$$

Proof. The start time of the slot allocated to a node can vary due to two reasons: the delay δ and the clock synchronization errors ϵ. Since these two causes are independent, they can be considered separately. Consider an arbitrary time window containing one time slot for each node. The minimum overlap in time of the slots requested by the nodes in the network is given by the worst-case scenario in which (i) *at least* one node N_1 acquires the slot as soon as possible (i.e., with a delay $\delta = 0$) and (ii) *at least* another node N_2 acquires the slot as late as possible (i.e., with the maximum delay $\delta = \delta_{max}$). Because of the clock synchronization error ϵ, the actual time at which a node requests the slots can be shifted by a maximum of σ_{sync} time units, consequently the maximum shift between two nodes requests for the slot can be shifted by $2\sigma_{sync}$. Hence, when both the nodes request slots with length $\delta_{max} + 2\sigma_{sync} + Q$, the slots are overlapped in time for at least Q time units. The lemma follows by considering that σ_{RB} time units are needed to restore the radio device state before terminating the slot, hence an additional interval of

length σ_{RB} must be subtracted from the actual time reserved for the real-time communication. □

Lemma V.1 allows defining the real-time meta-protocol.

B. Design of the real-time meta-protocol

This section states the rules defining the proposed real-time meta-protocol to be implemented on each node. Let t_0 be the the instant of the first slot request and let t_k be a generic time at which a node in the network requests a reservation slot.

- **Rule 0.** At time t_0, each node issues the first request for getting a real-time slot and such a request is periodically repeated every P time units; hence, $t_k = t_0 + kP$ with $k = 0, 1, \ldots$
- **Rule 1.** Each node always requests a slot of length $\widehat{Q}$ to the RM according to Lemma V.1.
- **Rule 2.** A node receiving the slot at time $t_k + \delta$ must wait until time $t_k + \delta_{max}$ before enabling the access to the radio device for the real-time protocol.
- **Rule 3.** At time $t_k + \delta_{max} + Q + 2\sigma_{sync}$, each node concludes the real-time communication restoring the radio status to BLE mode.

The slots temporal alignment is expressed by Rule 2 that forces all nodes to wait till δ_{max} even if the slot for a specific node is granted earlier. Rule 3 is in charge of restoring the BLE stack and reclaiming the remaining part of the requested slot for the BLE traffic that otherwise would be wasted.

An example of the behavior of the real-time meta-protocol is illustrated in Figure 5, where clock synchronization errors are not considered for the sake of clarity. Three nodes (N_1, N_2, N_3) perform a request at time $t_k = 0$ (**Rule 0**) of a slot of length $\widehat{Q}$ (**Rule 1**). Although each node N_i receives the slot at a different times δ_i, it waits until δ_{max} (**Rule 2**) before actually using the custom slot. At time $\delta_{max}+Q+2\sigma_{sync}$ each node requests to restore the radio for the BLE transmission (**Rule 3**), even though the received slot would finish at time $\delta_i + \widehat{Q}$. Note that the actual BLE slot starts after σ_{RB} time units due to a radio switch overhead.

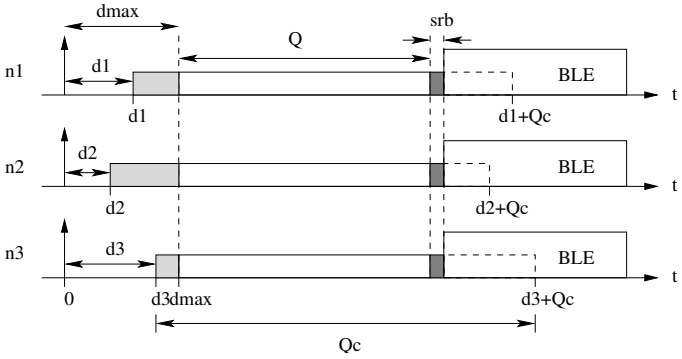

Figure 5: Example of slot requests from different nodes.

Note that in this specific example the slots overlap of $Q + 2\sigma_{sync}$ since all the requests are synchronously issued at time $t_k = 0$, while in the presence of a synchronization error the overlap time is guaranteed to be no smaller than Q, as stated in Lemma V.1.

C. Example of real-time traffic analysis

This section presents the analysis to guarantee the real-time traffic upon the reservation mechanism provided by the RM. Note that the reservation mechanism is independent of the real-time protocol used to regulate the access to the radio in the time slot Q. To make a specific example for the analysis, this section assumes that real-time slots are assigned by the Weighted Round Robin algorithm. In each real-time interval of length Q, the channel is accessed by the nodes according to the protocol, and each node N_i is assigned a time budget Q_i equal to

$$Q_i = \frac{U_i}{U^{\text{RT}}} Q, \tag{7}$$

where $U^{\text{RT}} = \sum_i U_i$ is the total bandwidth utilization required by the real-time streams. Note that, if the budget is assigned according to Equation (7), the sum of the budgets reserved to the nodes is equal to Q and the real-time bandwidth utilization is equal to $U^{\text{RT}} = Q/P$.

The following lemma provides an upper bound on the maximum time needed by node N_i to send a real-time message of length L.

Lemma V.2. *Given a node N_i with an assigned transmission budget Q_i the worst-case response-time for sending a real-time message of length L is upper-bounded by*

$$\mathcal{R}_i(L) = L + \left\lceil \frac{L}{\left\lfloor \frac{Q_i}{T_{pkt}} \right\rfloor T_{pkt}} \right\rceil (P - Q_i). \tag{8}$$

Proof. The worst-case for node N_i occurs when the message is sent ϵ units of time after the beginning of its slot. In this case, in fact, the Weighted Round-Robin scheduler reassigns the slot to the next node, determining a waiting time no larger than $(P - Q_i)$ time units before N_i can start the transmission. Due to the discretization imposed by the packet transmission time T_{pkt}, only $x = \lfloor Q_i/T_{pkt} \rfloor T_{pkt}$ time units can be actually used to transmit messages inside a slot.[1] Hence, the number of slots needed to transmit the message can be computed as $\lceil L/x \rceil$. Before granting the access to each of these slots, the node has to wait for $(P - Q_i)$ time units, thus obtaining a total waiting time of $\lceil L/x \rceil (P - Q_i)$ time units. The lemma follows by noting that the response-time is bounded by the maximum waiting time plus the message length L. □

Each node is in charge of transmitting real-time messages with a maximum length M_i, hence the response-time bound for the message transmission can be computed as $r_i = \mathcal{R}_i(M_i)$. However, this does not hold for all the nodes in the network. Due to the synchronization protocol, one of the nodes is in charge of sending additional periodic synchronization messages. Since the synchronization message is transmitted within the slot allocated to real-time messages with a higher priority, the response-time bound r_i of such messages is inflated. The following lemma allows bounding the response-time in the presence of synchronization messages.

[1] We assume that the packet transmission is skipped if it cannot complete within the end of the slot.

Lemma V.3. *Consider a synchronization message with length Q_{sync} and period P_{sync}. Let N_i be the node in charge of transmitting such a synchronization message. The worst-case response-time for sending a real-time message of length M_i from node N_i is upper-bounded by the least positive fixed-point of the following equation:*

$$r_i = \mathcal{R}_i \left(M_i + \left\lceil \frac{r_i}{P_{sync}} \right\rceil Q_{sync} \right). \tag{9}$$

Proof. The worst-case scenario occurs when the start of the synchronization period is aligned with the time in which the real-time message is issued. Let r_i be the (tentative) response-time of the real-time message under analysis. As long as the transmission of the message is pending, no more than $\lceil r_i/P_{\text{sync}} \rceil$ synchronization messages are issued. Hence, the total payload to be sent within the real-time slots amounts to $L = M_i + \lceil r_i/P_{\text{sync}} \rceil Q_{\text{sync}}$. By Lemma V.2, $\mathcal{R}_i(L)$ upper-bounds the response-time for transmitting such a payload, thus leading to the recursive expression for r_i. Hence the lemma follows. □

Finally, in order to guarantee that the timing constraints on messages generated by any node N_i are met it is sufficient to check that

$$\forall i = 1 \dots n, \quad r_i \leq D_i. \tag{10}$$

D. BLE traffic analysis

A sufficient condition to prevent BLE packet loss can be derived by analyzing the dual-protocol behavior within a single reservation period P. In fact, if the BLE stack is able to send all the messages produced by the node (by task τ_S, as illustrated in Figure 4) within the period P, then, at the end of each reservation period, no backlog is "pushed" in the next period, hence preventing packet loss in the long run.

We begin by deriving a bound on the BLE packets produced in an arbitrary time interval.

Lemma V.4. *As long as task τ_S is guaranteed to complete within its next activation, the maximum cumulative number of BLE packets produced by τ_S in* any *interval of length t is bounded by*

$$pp(t) = \left\lceil \frac{t + T_S}{T_S} \right\rceil n_S \tag{11}$$

Proof. Without loss of generality, assume that the interval of interest starts at time 0 and ends at time t. The jobs of τ_S released *before* time $-T_S$ are completed at time 0, thus cannot produce BLE packets within $[0, t]$. Similarly, the jobs of τ_S released *after* time t can be also discarded. In the interval of time $[-T_S, t]$, no more than $\lceil (t + T_S)/T_S \rceil$ jobs can be released by τ_S. Each of these jobs can produce at most n_S BLE packets. Hence, the lemma follows. □

We recall that BLE packets are transmitted through Radio Events, each of them transmitting n_B packets. Hence, the number of Radio Events needed to transmit the BLE packets produced in the reservation period P is no larger than

$$\left\lceil \frac{pp(P)}{n_B} \right\rceil. \tag{12}$$

It follows that the reservation period P must be large enough to accommodate the real-time budget Q with the associated overhead Θ (see Lemma V.1) and guarantee enough time for the BLE to send the maximum accumulated backlog. Thus, by applying Lemma V.4, we have

$$P \geq Q + \Theta + \left\lceil \frac{pp(P)}{n_B} \right\rceil T_B = \widehat{Q} + \left\lceil \left\lceil \frac{P + T_S}{T_S} \right\rceil \frac{n_S}{n_B} \right\rceil T_B. \tag{13}$$

Considering the properties of the ceiling function (see [34], p. 72), if n_B is a multiple of n_S, Equation (13) can be rewritten as

$$P \geq \widehat{Q} + \left\lceil \left(\frac{P + T_S}{T_S} \right) \frac{n_S}{n_B} \right\rceil T_B. \tag{14}$$

When the reservation mechanism prevents the BLE communication, the produced BLE packets are backlogged in the hardware device buffer. To guarantee that no BLE packets will be lost, it must be that, at any time, the maximum backlog is smaller than the buffer size n_H. The maximum backlog is accumulated in the worst-case scenario in which the first BLE packet is produced at the beginning of a real-time slot request: in this case the radio will not be available for BLE communication for at maximum $\widehat{Q}$ time units; then, after T_B time units, a BLE Radio Event will certainly occur. Hence, $\widehat{Q} + T_B$ represents the maximum time interval in which BLE packets are not processed for transmission. Therefore, the necessary condition to avoid packet loss can be expressed as follows:

$$pp\left(\widehat{Q} + T_B\right) \leq n_H. \tag{15}$$

VI. System Parameters Design

This section proposes a design methodology for determining the reservation parameters (namely, the budget Q and the period P) that maximize the bandwidth U^{RT} reserved for the real-time communication while guaranteeing that no packets are lost for the BLE communication due to over-backlog. In particular, the optimization design problem can be expressed as follows:

Find reservation parameters (Q, P) maximizing the bandwidth $U^{RT} = \frac{Q}{P}$ while ensuring no BLE packets loss.

To simplify the presentation of the design methodology, we assume that the number of packets n_B transmitted in each Radio Event is a multiple of the number of packets n_S transmitted by the task managing the BLE communication. Note that an approach similar to the one presented in this section can also be applied to the general case in which this assumption does not hold. However, the presentation of such a general case would require a non-trivial dedicated discussion, which is out of the scope of this paper, and hence is left as a future work.

Under the above assumption, the conditions to guarantee the absence of packet loss for the BLE are expressed through Equation (14) and Equation (15), which define a region $\mathcal{R}$ in the P-Q plane expressing all possible values for the reservation parameters such that no over-backlog will occur. An example of region $\mathcal{R}$ is illustrated in Figure 6. The goal of the optimization problem is to identify the point $(Q_{\text{opt}}, P_{\text{opt}})$

belonging to this region that maximizes the bandwidth U^{RT}, which corresponds to the slope of the line passing from the point $(Q_{\text{opt}}, P_{\text{opt}})$ and the origin of the plane.

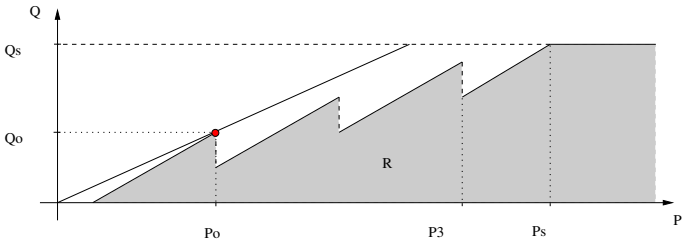

Figure 6: Example of region for the optimization problem.

First of all, note that, due to the ceiling operator, the region includes multiple local maxima. Such maxima can be computed by identifying the points in which the argument of the ceiling function in Equation (14) is an integer, that is, the values of P satisfying the following equation

$$\left(\frac{P+T_S}{T_S}\right)\frac{n_S}{n_B} = k,\ k \in \mathbb{N}. \tag{16}$$

From Equation (14) it results that

$$\widehat{Q} \leq P - \left\lceil \left(\frac{P+T_s}{T_S}\right)\frac{n_S}{n_B} \right\rceil T_B, \tag{17}$$

hence, the family of maxima as a function of $k \in \mathbb{N}$ is

$$\begin{cases} Q_{\max}(k) = P_{\max}(k)\left(1 - \frac{n_S}{T_S n_B} T_B\right) - \frac{n_S}{n_B} T_B - \Theta \\ P_{\max}(k) = T_S\left(k\frac{n_B}{n_S} - 1\right). \end{cases} \tag{18}$$

Now, note that Equation (15) leads to a constant upper-bound on the budget Q, that is

$$Q \leq T_S\left(\left\lfloor \frac{n_H}{n_S} \right\rfloor - 1\right) - T_B - \Theta. \tag{19}$$

Such an upper-bound cuts region $\mathcal{R}$ in the point $(Q_{\text{sat}}, P_{\text{sat}})$ (see Figure 6). Such a point is a candidate for having the maximum bandwidth U^{RT} and can be found by replacing the upper-bound of Equation (19) in Equation (14), so obtaining:

$$\begin{cases} Q_{\text{sat}} = T_S\left(\left\lfloor \frac{n_H}{n_S} \right\rfloor - 1\right) - T_B - \Theta \\ P_{\text{sat}} = Q_{\text{sat}} + \Theta + \left\lceil \left(\frac{P_{\text{sat}}+T_S}{T_S}\right)\frac{n_S}{n_B} \right\rceil T_B, \end{cases} \tag{20}$$

where P_{sat} can be computed by using a *fixed-point iteration*.

Being all the points $(Q_{\max}(k), P_{\max}(k))$ on the same line, candidates for the optimal point $(Q_{\text{opt}}, P_{\text{opt}})$ have to be found by only considering the first point for $k = 1$ and the last point before the saturation, which occurs for

$$\overline{k} = \left\lceil \left(\frac{P_{\text{sat}}+T_S}{T_S}\right)\frac{n_S}{n_B} \right\rceil - 1. \tag{21}$$

In conclusion, the optimal solution can be found by identifying the couple of budget Q and period P giving the maximum bandwidth as follows

$$U^{\text{RT}} = \max\left\{ \frac{Q_{\text{sat}}}{P_{\text{sat}}}, \frac{Q_{\max}(1)}{P_{\max}(1)}, \frac{Q_{\max}(\overline{k})}{P_{\max}(\overline{k})} \right\}. \tag{22}$$

VII. Experimental results

This section discusses some implementation details on a real platform and presents some experiments carried out to characterize some of the parameters needed for the analysis and design of the dual-protocol, as presented in Section V and Section VI, respectively.

We begin by discussing our experimental setup in Section VII-A, then we present the experimental results carried out from three different experiments. The first two experiments, respectively reported in Sections VII-B and VII-C, have been conducted to empirically measure the parameters δ and σ_{sync} required in Equation (6). Such parameters are crucial for the analysis presented in Section V in order to guarantee the real-time slots. Finally, to validate in practice the above mentioned analysis, in Section VII-D we present an evaluation of the proposed slot reservation mechanism. This experiment empirically confirmed that real-time slots are correctly provided by the dual-protocol mechanism when configured according to our analysis.

A. Experimental setup

Experimental data have been derived on a network of wireless nodes built around the nRF51822 device produced by Nordic, combining a low-power ARM Cortex-M0 core with a 2.4 GHz radio transceiver. The core is endowed with 256 kB of flash memory and 16 kB of RAM memory. The transceiver supports the BLE communication stack plus a raw operation mode that can be used to run a custom real-time protocol. Nodes exchange data among each others using the custom protocol and send data to a master unit (consisting of an Android tablet) only using the BLE protocol.

The software in the wireless nodes has been developed in C language on top of the ERIKA Enterprise real-time kernel [35]. The BLE stack is provided by Nordic as a binary library named SoftDevice. The access to the raw radio mode is performed through an asynchronous API of the SoftDevice, passing two parameters Q_{req} and δ_{req}, where Q_{req} is the length of the requested slot and δ_{req} is the delay by which the slot has to be granted. If the slot cannot be granted within δ_{req}, because it is not compatible with the timing of the BLE protocol, the API returns an error code; otherwise, a success code is returned and a slot start signal is sent to the application through a software interrupt. To make sure that a bandwidth U^{RT} is guaranteed for the real-time traffic, Q_{req} must be equal to $\widehat{Q}$ (see Lemma V.1) and each slot request must succeed. This is obtained by specifying a δ_{req} value equal to the upper bound δ_{max} compatible with a correct behavior of the SoftDevice, that is $\delta_{req} = \delta_{max}$.

Table II reports the values of the parameters that characterize the Nordic-based reference platform.

n_B	T_B	t_p	n_H	σ_{BR}	σ_{RB}	δ_{max}	t_{prep}
6	30 ms	967 μs	6	350 μs	10 μs	10 ms	1500 μs

Table II: System constants for the Nordic platform.

B. Experimental delay evaluation

The first experiment has been carried out to characterize the delay δ. Timing measurements were performed on a node

communicating with the master device to profile the length of the interval between the instant at which a time slot is requested (named TS_REQ) and the instant at which the signal notifies that the radio can start transmitting using the custom protocol (named INT_TS_START).

The experiment was repeated with different values of the period P, namely, 20 ms, 35 ms, 50 ms, and 100 ms. The sequence of requests has been issued by a periodic task and the delay δ of each request has been measured. To evaluate the effect of the BLE communication on the delay, the experiment has been done under two load configurations: without data traffic (i.e., transmitting only the message needed for the initialization phase) and with data traffic generated at a frequency of 50 Hz ($T_S = 20$ ms). The traffic is composed by one BLE packet for each transmission ($n_S = 1$) because the requested application payload is 4 bytes, which is smaller that the 20 bytes of the BLE payload.

For each configuration, the delays have been acquired for an interval of 2000 seconds.

Figure 7 reports the delay distribution obtained with a period $P = 100$ ms and the described BLE data traffic. The distributions produced with the other configurations are very similar and are not shown here for lack of space.

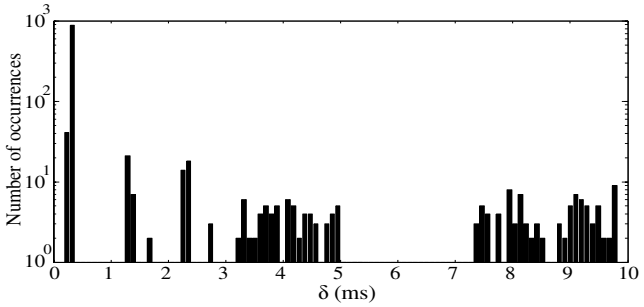

Figure 7: Delay distribution with a request period of 100 *ms and BLE data traffic.*

As it can be easily seen, the delay has a huge variability, mainly due to the different internal states in which the SoftDevice can be when the request is received. In particular, most of the requests are grated in about 0.4 ms, which is the time needed to switch the context when there are no ongoing BLE activities. However, the maximum measured delay is significantly larger and close to 10 ms. This is the time needed to prepare a BLE transmission with the maximum number of packets (n_B), send them and put the stack in a safe state before enabling raw transmissions.

C. Synchronization error evaluation

The performance of the dual-protocol depends on the bandwidth overhead due to the guard intervals (σ_{sync}) that are needed to avoid synchronization issues caused by a relative drift among the nodes clocks. Two experiments have been performed to show that the use of a synchronization protocol can limit the clock drift and hence can bound the maximum displacement between the requests for real-time slots in different nodes. This provides safe guard intervals (σ_{sync}) with an acceptable bandwidth overhead for the dual-protocol approach. This test has been carried out using two nodes. Each node notifies its requests on a digital output and requests from both nodes are acquired with a logic analyzer in order to make the measurement error negligible.

In the first experiment, the nodes execute the synchronization procedure during the initialization step and then operate for an interval of 30 minutes without re-synchronizing their clocks. Slot requests are performed with a period $P = 500$ ms, while the BLE stack is sending one packet every 30 ms. Figure 8 shows that the relative clock error (ΔT) increases linearly, as expected. With the adopted experimental configuration, the achieved guard intervals is $\sigma_{sync} = 3$ ms.

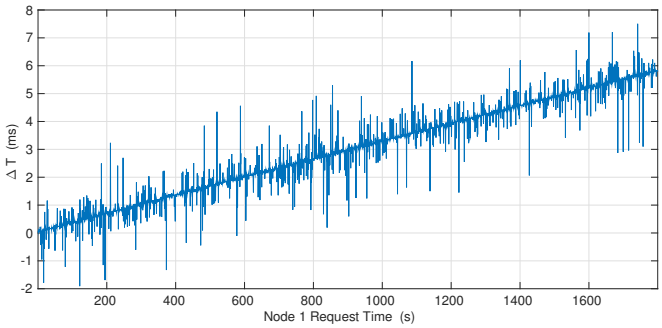

Figure 8: Drift between slot requests performed by 2 nodes with initial synchronization only.

Note that the spikes appearing on the theoretical linear behavior are motivated by the fact that the interrupt service routine in charge of the system clock can be delayed by the execution of the SoftDevice interrupt handler, which runs at the highest priority.

The second experiment uses the same setup as the previous one, with the only difference that the synchronization is periodically executed with a period (P_{sync}) equal to 8 minutes. Figure 9 shows the relative clock error of the two nodes. As clear from the plot, the synchronization protocol is able to bound the drift by a constant that takes into account the drift rate in the nodes, the synchronization period, and the execution time of the SoftDevice interrupt routine. Note that the spikes caused by the delay in the execution of the timer interrupt service routine prevent the synchronization protocol to totally reset the clock error between the two nodes. The protocol is unaware of the delay and translates it in a clock bias for the whole interval P_{sync} till the next synchronization event. However, such an error does not affect the dual-protocol approach, which only requires a bound on the maximum value of the drift.

D. Evaluation of the slot reservation mechanism

This experiment has been carried out to verify that the duration of the slots provided by the BLE stack is sufficient to satisfy the bandwidth requirements needed for the integration of a real-time protocol. To achieve a desired slot duration Q, a slot of duration $\widehat{Q}$ must be requested (see Lemma V.1), taking into account all the overheads and constraints resulting from

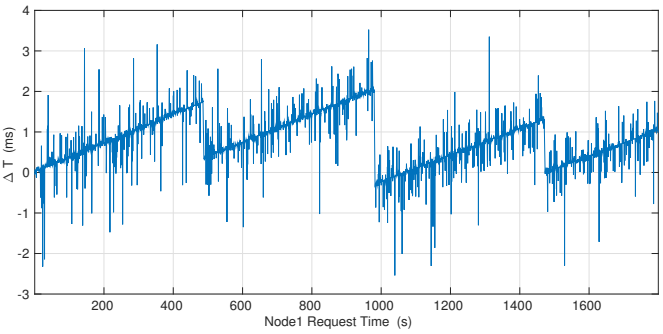

Figure 9: Drift between slot requests performed by 2 nodes when the synchronization is repeated every 8 minutes.

the implementation of the proposed reservation mechanism. As stated in Equation (6), reported below for convenience,

$$\widehat{Q} = Q + \Theta = Q + \delta_{max} + \sigma_{RB} + 2\sigma_{sync}. \qquad (6)$$

the overhead Θ is composed by σ_{RB}, which is constant and depends only upon the SoftDevice execution, and the other two factors that have been measured by the previous two experiments ($\delta_{max} = 10$ ms and $\sigma_{sync} = 3$ ms).

In this experiment, $Q = 30$ ms and $\Theta = 16.01$ ms, so $\widehat{Q} = 46.01$ ms. Slots were periodically requested to the SoftDevice every 100 ms and the actual slot duration Q^* has been measured. Timing measurements were performed by switching an output pin at the beginning and at the end of the Slot, and capturing them by a logic analyzer in order to make the measurement error negligible. In this way, the achieved precision is at the single executed opcode for the used CPU.

Figure 10 shows the distribution of the Q^* measurements. As it can be easily seen, all granted time slots are longer than the minimum desired value Q, thus confirming the feasibility of the proposed approach.

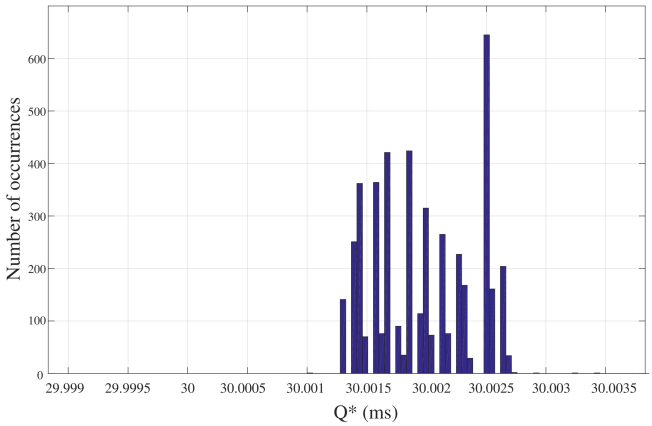

Figure 10: Distribution of the measured slot duration Q^*.

VIII. Conclusions

This paper presented a bandwidth reservation mechanism for partitioning the radio transceiver between two protocols: the Bluetooth LE and a real-time custom protocol. The advantage of this approach is to enable real-time communication among nodes in a distributed embedded system, while preserving a standard connectivity with mobile devices, as tablets and smart phones. The real-time bandwidth partition can be used for internode communication subject to timing constraints and for node synchronization, necessary to trigger the slots allocated to the two protocols and reduce the timing error associated with the data acquired from different nodes.

An evaluation performed on a Nordic reference platform showed the applicability of the proposed approach and the capability to support a custom real-time protocol on the raw radio with a bounded overhead of about 10 milliseconds. This allows guaranteeing, for example, reserving a real-time slot of 30 milliseconds every 100 milliseconds without jeopardizing a BLE communication requiring a bandwidth of 1000 Mbit/s.

The next step will be the analysis of the different real-time protocols and synchronization mechanisms to understand what features are required to maximize the advantages of this new approach.

References

[1] J. J. P. Tsai, Y. Bi, S. J. H. Yang, and R. A. W. Smith, *Distributed Real-Time Systems: Monitoring, Visualization, Debugging, and Analysis*. John Wiley and Sons, Inc., New York, 1996.

[2] T. Facchinetti, G. Buttazzo, M. Caccamo, and L. Almeida, "Wireless real-time communication protocol for cooperating mobile units," in *Proceedings of the 2nd International Workshop on Real-Time LANs in the Internet Age (RTLIA 2003)*, Porto, Portugal, July 1 2003, pp. 1–4.

[3] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*, ser. Springer Series in Statistics. Springer, 2005.

[4] *BlueTooth Specification Version 4.2*, https://www.bluetooth.org/en-us/specification/adopted-specifications, 2014.

[5] *ANT Message Protocol and Usage*, https://www.thisisant.com/resources/Fant-message-protocol-and-usage.

[6] *Bluetooth Smart CSR101x Product Family*, http://www.csr.com/products/bluetooth-smart-csr101x-product-family.

[7] *WirePas PINO*, http://www.wirepas.com/Pino_Product_Brief.pdf, 2016.

[8] P. Buonocunto and M. Marinoni, "Tracking limbs motion using a wireless network of inertial measurement units," in *Proc. of the 9th IEEE Int. Symp. on Industrial Embedded Systems (SIES 2014)*, Pisa, Italy, June 18-20, 2014.

[9] E. Mackensen, M. Lai, and T. M. Wendt, "Performance analysis of an bluetooth low energy sensor system," in *Proc. of the 1st IEEE International Symposium on Wireless Systems (IDAACS-SWS 2012)*, Offenburg, Germany, 2012, pp. 62–66.

[10] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[11] J. Suhonen, M. Kohvakka, V. Kaseva, T. D. Hämäläinen, and M. Hännikäinen, *Low-power wireless sensor networks: protocols, services and applications*. Springer Science & Business Media, 2012.

[12] IEEE 802.15.4 Std-2011, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE-SA Standards Board, Tech. Rep. STDPD-97126, June 2011.

[13] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario," in *Proc. of the IEEE International Wireless Symposium (IWS 2013)*, Beijing, China, Apr. 2013, pp. 1–4.

[14] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[15] A. Kutlu, H. Ekiz, and E. Powner, "Performance analysis of MAC protocols for wireless control area network," in *Proc. of Int. Symp. on Parallel Architectures, Algorithms, and Networks*, 1996, pp. 494–499.

[16] A. Willig, "Polling-based mac protocols for improving real-time performance in a wireless profibus," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 4, pp. 806–817, 2003.

[17] T. L. Crenshaw, A. Tirumala, S. Hoke, and M. Caccamo, "A Robust Implicit Access Protocol for Real-Time Wireless Collaboration," in *Proc. of the 17th IEEE Euromicro Conference on Real-Time Systems (ECRTS 2005)*, Palma de Mallorca, Spain, Jul. 2005.

[18] J. Wu, J.-C. Liu, and W. Zhao, "Utilization-Bound Based Schedulability Analysis of Weighted Round Robin Schedulers," in *Proc. of 28th IEEE Real-Time Systems Symposium*, Tucson, AZ, USA, Dec. 2007.

[19] A. Koubaa, M. Alves, and E. Tovar, "Energy/Delay Trade-off of the GTS Allocation Mechanism in IEEE 802.15.4 for Wireless Sensor Network," *International Journal of Communication Systems*, vol. 20, no. 7, pp. 791–808, July 2007.

[20] E. Toscano and L. Lo Bello, "Multiplechannel Superframe Scheduling for IEEE 802.15.4 Industrial Wireless Sensor Networks," *IEEE Trans. on Industiral Informatics.*, vol. 8, no. 2, May 2012.

[21] A. C. W. Wong, M. Dawkins, G. Devita, N. Kasparidis, A. Katsiamis, O. King, F. Lauria, J. Schiff, and A. J. Burdett, "A 1v 5ma multimode ieee 802.15.6/bluetooth low-energy wban transceiver for biotelemetry applications," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, 2013.

[22] T. Jin, G. Noubir, and B. Sheng, "Wizi-cloud: Application-transparent dual zigbee-wifi radios for low power internet access," in *Proc. of the INFOCOM, 2011*, Shangai, China, Apr. 2011, pp. 1593–1601.

[23] *NORDIC Semiconductor. nRF51822 Product Specification v3.1*, http://www.nordicsemi.com.

[24] E. Zehavi and R. Nevo, "Wireless apparatus having a transceiver equipped to support multiple wireless communication protocols," 2006, US Patent 6,990,082.

[25] R. Bridgelall, "Dual mode wireless data communications," 2005, US Patent 6,895,255.

[26] P. Desai, C. Kwan, A. Polo, S. Qiao, C. Tian, Y. Zhuang, G. Xie, L. Caliaperoumal, and B. Ibrahim, "Method and system for a dual-mode bluetooth low energy device," May 27 2014, uS Patent 8,737,917.

[27] T. K. Moon, "Error correction coding," *Mathematical Methods and Algorithms. John Wiley and Son*, 2005.

[28] H. Karl and A. Willig, *Protocols and architectures for wireless sensor networks*. John Wiley & Sons, 2007.

[29] *NORDIC Semiconductor. SoftDevice Specification v2.0*, http://www.nordicsemi.com.

[30] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, 2005.

[31] P. Kumarawadu, D. Dechene, M. Luccini, and A. Sauer, "Algorithms for node clustering in wireless sensor networks: A survey," in *Proc. of the 4th Inter. Conference on Information and Automation for Sustainability (ICIAFS 2008)*, Sri Lanka, Dec. 2008.

[32] A. Koubaa, M. Alves, and E. Tovar, "Modeling and worst-case dimensioning of cluster-tree wireless sensor networks," in *Real-Time Systems Symposium, 2006. RTSS'06. 27th IEEE International*. IEEE, 2006, pp. 412–421.

[33] J. B. Schmitt and U. Roedig, "Sensor network calculus–a framework for worst case analysis," in *Distributed Computing in Sensor Systems*. Springer, 2005, pp. 141–154.

[34] R. Graham, D. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1994.

[35] *Erika Enterprise RTOS kernel*, http://erika.tuxfamily.org/drupal/.