TSCH meets BLE: Routed Mesh Communication over BLE

Laura Harms[1,2], Olaf Landsiedel[1,2]
[1]*Kiel University, Germany*
[2]*Chalmers University of Technology, Sweden*
{lah, ol}@informatik.uni-kiel.de

***Abstract*—Bluetooth Low Energy (BLE) is the prevalent communication protocol for the Internet of Things. However, for time-critical applications requiring time-synchronized multi-hop networks with often multiple node exchanging data at the same time slot, BLE lacks a solution. Instead, we commonly see IEEE 802.15.4 being used with its Time-Slotted Channel Hopping (TSCH) MAC layer.**

In this work, we build TBLE, which brings the established TSCH protocol to BLE, enabling BLE to be used for time-synchronized routed mesh communication. We show that in experimental testbed deployments, TBLE achieves similar performance to TSCH, with the possibility for lower average latencies of up to 20%. Moreover, due to the higher spectral efficiency of BLE compared with IEEE 802.15.4 (40 vs. 16 channels), more parallel routed communications are possible with TBLE, further reducing latency and increasing throughput.

I. Introduction

With more than 5 billion ($5 * 10^9$) BLE devices estimated to be shipped in 2023 alone [10] and a continuing rise in popularity, Bluetooth Low Energy (BLE) is the prevalent standard for communication in low-power wireless networks. Due to its wide availability, low-cost and energy efficiency, BLE is supported by practically all smart devices we interact with today. In contrast, most wireless industrial devices and many smart home devices use IEEE 802.15.4 instead of BLE. Of the two protocols, the physical layer of BLE is the less complex one using a GFSK modulation scheme in contrast to the O-QPSK modulation scheme of IEEE 802.15.4, allowing for cheaper radios.

Both protocols have limited range and thus rely on, i.e., multi-hop mesh networking for communicating over longer distances. In the field of IEEE 802.15.4, there are several established protocols, including those for flooding-based communication [15], [16] as well as routing-based communication [2]. For routing-based communication, an established technique is Time-Slotted Channel Hopping (TSCH), the standard MAC layer protocol in IEEE 802.15.4. For BLE, the standard mesh protocol is Bluetooth Mesh [7], [8], a protocol using managed flooding on top of BLE advertisements. While flooding-based protocols usually use the entire network for sending a message, routing-based protocols allow multiple parallel communications in the same network at the same point in time. Yet, to our knowledge, there is no time-synchronized routing-based mesh communication protocol for BLE.

Several works looked at the combination of IEEE 802.15.4 TSCH and BLE, either in terms of coexistence [12], [17] or by using a single radio for both BLE and IEEE 802.15.4 and communicating TSCH control information using concurrent BLE transmissions [6]. Especially the latter work raises the question of why there is the need to continue using IEEE 802.15.4 for TSCH communication while only communicating control information over BLE.

In this paper, we combine the BLE PHY with the MAC layer protocol TSCH and call this combination TBLE. TBLE sends standard TSCH packets as part of time-synchronized BLE advertisements, enabling routed mesh communication over BLE with TSCH and replacing IEEE 802.15.4. TBLE enables the use of well established protocols including deadline-based real-time communication protocols on top of BLE. We design TBLE for the use with any BLE radio. We exemplarily implement a BLE driver for the nRF52840 DK [23] for Contiki-NG [24] and adapt it to allow the transmission of valid IEEE 802.15.4 TSCH frames within BLE packets. We study both the coded (125 kbps/500 kbps) and the uncoded (1 Mbps/2 Mbps) PHYs of BLE and experimentally evaluate TBLE's performance on a low-power wireless testbed using the well established autonomous scheduler Orchestra [14] which is included in Contiki-NG and compare its performance to TSCH over IEEE 802.15.4.

Our evaluation on a testbed shows, that especially the coded BLE modes achieve a similar connectivity within a deployment as IEEE 802.15.4. TBLE achieves similar performance to TSCH, with the possibility for lower average latencies of up to 20%. Moreover, due to the higher spectral efficiency of BLE compared with IEEE 802.15.4 (40 vs. 16 channels), more parallel routed communications are possible with TBLE, further reducing latency and increasing throughput.

Overall, we make the following contributions:

- We present TBLE, a protocol closing the gap of routed mesh-communication in BLE. TBLE extends the established TSCH standard.
- We design and implement a BLE driver for the Nordic nRF52840 DK for Contiki-NG and adjust it to be compatible with the Contiki-NG IEEE 802.15.4 TSCH and 6TiSCH stack. We make it publicly available[1] as open source.
- We are the first to run TSCH over BLE, demonstrating TBLE as a practical routed mesh-protocol for BLE.

[1]Available as open-source at: https://github.com/ds-kiel/TBLE.

- We experimentally evaluate TBLE and compare it to IEEE 802.15.4 TSCH, showing its feasibility and a possible performance increase over TSCH without modifying any upper-layer protocols.

We structure the remainder of this paper as follows. Section II gives the necessary background information on IEEE 802.15.4, BLE and TSCH, followed by a detailed dissection of TSCH in Section III. In Section IV we introduce the design of TBLE, followed by our experimental evaluation of TBLE and its comparison to IEEE 802.15.4 TSCH in Section V. In Section VI we discuss a selection of works related to this topic, and conclude our work in Section VII.

II. Background

In this section, we introduce the required background information on IEEE 802.15.4, Time-Slotted Channel Hopping (TSCH), and Bluetooth Low Energy (BLE).

A. IEEE 802.15.4

IEEE 802.15.4 is a low-power personal area radio protocol introduced in 2003 [1], initially for the 2.4 GHz (2400 – 2483.5 MHz) ISM band. It operates at a data rate of 250 kbps and uses a robust modulation scheme of O-QPSK with DSSS (direct-sequence spread spectrum). IEEE 802.15.4 specifies 16 channels that are 2 MHz wide and 5 MHz apart. On the physical layer, an IEEE 802.15.4 packet consists of 4 preamble bytes, a 1 byte start of frame delimiter (SFD), a 1 byte packet length, and up to 127 bytes of payload.

Besides the physical layer, the IEEE 802.15.4 standard also defines the medium access control (MAC) layer. One defined MAC layer is Time-Slotted Channel Hopping (TSCH).

B. Time-Slotted Channel Hopping (TSCH)

Time-Slotted Channel Hopping (TSCH) [2] is a standardized MAC layer protocol (IEEE 802.15.4e) for low-power wireless mesh networks. TSCH combines Time-division multiple access (TDMA) with Frequency-division multiple access (FDMA) and a pseudo-random channel hopping mechanism. Communication occurs in distinct time-frequency-slots, with as many concurrent communications as channels included in the hopping sequence (maximally 16). The channel hopping allows TSCH to withstand narrowband interference.

Slots in TSCH have a standard length of 10 ms and allow the transmission of a single IEEE 802.15.4 packet followed by an optional acknowledgement upon successful reception. Slots can be reserved for sending and receiving Enhanced Beacons (EB). Beacons are sent regularly containing control information for nodes to associate to the TSCH network and to keep the network in sync and alive.

C. Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) [9] is a short-range and low-power communication protocol in the 2.4 GHz ISM band targeting single-hop communication between two devices. BLE uses 40 2-MHz wide frequency channels, which use Gaussian Frequency Shift Keying (GFSK) as a modulation scheme. Three of these channels are reserved for (primary) advertisements and broadcasts, while the other 37 are reserved for connected communication and secondary advertisements. BLE offers an uncoded PHY with data rates of 1 Mbps (standard data rate) and 2 Mbps, and since Bluetooth 5.0 even a long-range coded PHY with data rates of 125 kbps and 500 kbps.

PHY packet format. The physical layer packet format of BLE differs between the uncoded PHY and the coded PHY. The uncoded PHY packet starts with a 1 or 2 byte preamble of alternating ones and zeros, for a data rate of 1 Mbps and 2 Mbps, respectively. It is followed by the 4-byte access address, identifying packets belonging to a connection. For advertisement packets, the advertisement address is fixed to `0x8E89BED6`. Afterward, the packet contains between 2 and 258 bytes payload (PDU) and a 3-byte cyclic redundancy check (CRC) code for error correction. The coded PHY packet generally contains the same components, however, with additional fields for error correction (see Fig. 1). The preamble is uncoded, consisting of 10 repetitions of `0x3C` transmitted with a data rate of 1 Mbps. The first forward error correction (FEC) block is always transmitted with a data rate of 125 kbps containing the access address and the coding indicator (CI). The CI indicates the coding of the second FEC block, deciding whether it uses a data rate of 125 kbps or 500 kbps. The PDU and the CRC are then transmitted with the indicated coding afterward.

Advertisements. BLE has two operation modes: connected and non-connected. In the non-connected mode, BLE devices disclose their presence and advertise their services to nearby devices. These services include, i.e., media control services or weather information (e.g., temperature data). For some of these services, a receiver has to connect to the advertising device and communicate in connected mode.

Advertisement data. In non-connected mode, an advertiser sends data (PDU) consisting of a 2-byte header, followed by one or multiple advertising data (AD) structures. The first byte of the header contains a 4-bit PDU type, 1 bit reserved for future use, a 1-bit flag whether the advertiser supports the LE channel selection algorithm, two 1-bit flags whether the advertiser's and the target device's addresses are random or public, respectively. The second header byte contains the length of the subsequent advertising payload. An AD structure consists of a 1-byte length, n bytes AD type (e.g., an identifier that a list of service UUIDs follows) and $lenght - n$ bytes AD data (e.g., the list of service UUIDs).

III. Dissecting TSCH

After a general introduction of Time-Slotted Channel Hopping (TSCH) in the background, we analyze the inner workings of TSCH. For that, we study the timings within a TSCH timeslot and the time synchronization mechanism of TSCH, especially in the context of the implementation of TSCH in Contiki-NG.

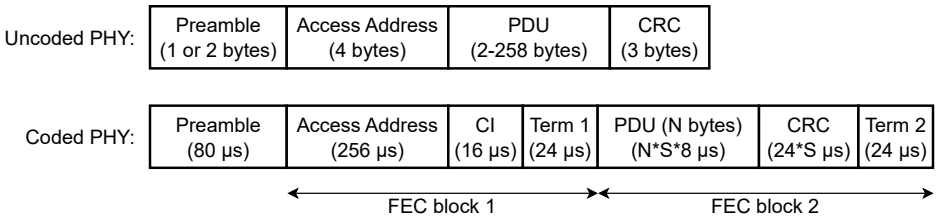


Fig. 1: BLE PHY packet formats

TABLE I: IEEE 802.15.4e TSCH timeslot timings.

Name	Time offset / duration (μs)
CCAOffset	1800
CCA	128
TxOffset	2120
MaxTx	4256
RxOffset	1020
RxWait	2200
RxAckDelay	800
TxAckDelay	1000
AckWait	400
MaxAck	2400
Sum	9776
Timeslot Length	10000

A. TSCH Timeslot Timing

A TSCH timeslot allows the transmission of one packet with a subsequent acknowledgement in case the packet was received. We illustrate the timing within a timeslot for both the sender and receiver in Fig. 2, and provide the timing offsets and durations in Table I.

At the sender's end, a timeslot starts with the transmission offset (`TXOffset`). During this offset, the sender configures its radio, turns it on, and potentially performs Clear Channel Assessment (CCA). Moreover, the sender prepares the packet (i.e., adding headers) and starts transmitting. By the end of the `TXOffset`, the preamble and the Start of Frame Delimiter (SFD) should be transmitted. Following the SFD, the radio transmits the TSCH frame standardized to a maximum length of 128 bytes including 1 length byte. TSCH reserves a time of 4256 μs equating to 133 bytes for this. After the transmission, the sender turns off its radio, and in case of an expected acknowledgement (ACK), it reconfigures the radio into receive mode and turns it on by the end of the `RxAckDelay`. If the sender, which is waiting for the ACK does not receive anything by the end of `AckWait`, it turns off its radio. Otherwise, if it detects an ACK, it receives it and turns off the radio afterward.

The receiver starts with an `RxOffset` during which it configures the radio and turns it on to listen for incoming packets. If it does not start receiving a packet within `RxWait` it turns off its radio. If the receiver receives a valid packet that requires acknowledgement, it prepares its radio for transmission during `TxAckDelay` and by the end of `TxAckDelay` it should have transmitted the packet's synchronization header. The full acknowledgment of maximally 69 bytes (plus 1 length byte) has to be transmitted within the time given by `MaxAck` of 2400 μs (the duration of 75 bytes).

For both the sender and the receiver, we see a mismatch of a time equaling 5 bytes between the reserved transmission times and the maximum packet lengths that can be transmitted. We assume that the given times include the times for transmitting the synchronization header, even though the synchronization header should have been transmitted already during `TxOffset` or `TxAckDelay` for frame and ACK, respectively.

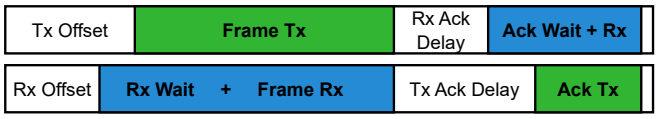


Fig. 2: Simplified TSCH timeslot timing. We omit the optional `CCAOffset` and the `CCA`, which happen during `TxOffset`, if enabled. Please note: the illustrated timing is not to scale. For the correct timing, see Table I.

B. TSCH packet duration

The TSCH timeslot timing provides us with the maximum time a transmission might take (`MaxTx`). However, during operation, we do not wait until the end of `MaxTx` before continuing with the next field of the timeslot if we send packets shorter than the maximum length. Instead, we directly continue once the transmission is over. Instead of probing the radio for the end of the transmission, the implementation of TSCH in Contiki-NG computes the transmission time of the packet. The implementation defines the TSCH packet duration as $duration_{packet} \leftarrow airtime_{byte} * (len + overhead_{PHY})$, with a PHY overhead of 3 for the nRF52840.

C. TSCH time synchronization

On the physical level, a radio can precisely timestamp certain events related to the radio packet, which we can use as markers for time synchronization. For example, in IEEE 802.15.4 mode, the nRF52840 chipset can timestamp events including *framestart*, *Address sent or received*, *Packet payload sent or received*, and *Packet sent or received*. The TSCH implementation in Contiki-NG uses the *start of frame delimiter (SFD)* as the timestamp to synchronize on. While the radio cannot timestamp the start of the SFD, it can timestamp the *framestart*, which is the timestamp right after transmitting or receiving the SFD. Thus, Contiki-NG can derive the necessary timestamp easily from the *framestart* event timestamp. As the SFD is 1 byte long (and takes 32 μs), we can compute the SFD timestamp by subtracting 32 μs from the recorded timestamp.

D. Hopping sequences

For counteracting narrowband interference, TSCH uses channel hopping according to a pseudo-random hopping sequence. TSCH defines a 9-bit linear feedback shift register to determine these hopping sequences. Common hopping sequences include a single-channel hopping sequence (channel 20), a four-channel hopping sequence (channels 15, 20, 25, and 26), and a 16-channel hopping (all IEEE 802.15.4 channels).

IV. Design

In this section, we discuss our design enabling TSCH on top of BLE advertisements, which closes the gap of routed mesh communication in BLE. We discuss the TSCH timeslot timings regarding the four different BLE data rates and show how we achieve time synchronization. Moreover, we discuss our BLE packet format and the channel hopping sequences for TBLE.

A. Overview

The general idea behind TBLE is to replace the IEEE 802.15.4 PHY with a BLE PHY and send standard TSCH packets as part of BLE advertisements. For other BLE devices, these appear like standard BLE packets, while devices running TBLE can recognize them and form a standard TSCH network using a BLE PHY instead of the IEEE 802.15.4 PHY for communication. For that, we have to change the TSCH timing to work with different data rates and embed the TSCH payload into BLE packets. Moreover, BLE offers significantly more radio channels in the same spectrum (40 instead of 16), due to a lower channel spacing, and thus, we can use different and longer hopping sequences allowing both more possibilities to avoid interference and a higher total bandwidth with more parallel communications. Lastly, a BLE radio does not offer the same timestamping capabilities as an IEEE 802.15.4 radio; thus we have to identify a different timestamp for time synchronization.

B. Derived Timing

In Section III-A, we explore the timing of the standard 10 ms TSCH slot (cf. Fig. 2). The different offsets and wait times in the standard TSCH slot are partly dependent on the radio's data rate. The ones dependent on the radio's data rate are the maximum frame length (`MaxTX`), the maximum ACK length (`MaxAck`), the `TxOffset`, and the `AckDelay`. The former two values depend on a packet's maximum time on air, thus on the maximum number of bytes and the radio's data rate. The latter two depend only to a minor extent on the radio as they contain mainly processing and wait times and, in addition, the transmission time of the PHY synchronization header. The `CCA` duration might be dependent on the radio, yet it does not take more time than standardized on the radio we tested it on, which supports both IEEE 802.15.4 and BLE. All other times seem to be independent of the radio. Instead, some of them are dependent on the device's CPU speed. As we only change the physical layer (from IEEE 802.15.4 to BLE) and otherwise keep the same processor capabilities, we only change the times strongly affected by the physical data rate: `MaxTX` and `MaxAck`. As the PHY synchronization header takes less time for either BLE mode than for IEEE 802.15.4, we keep `TxOffset` and `AckDelay` unchanged, which increases the times for data processing by $152\ \mu s$ and $80\ \mu s$ for the uncoded and coded BLE modes, respectively. We also keep the guard times for correctly receiving the beginning of a data packet (`RxOffset` + `RxWait`) the same as in IEEE 802.15.4. Those guard times do not have any effect on the total slot length for IEEE 802.15.4 and do not exceed the derived timeslot lengths for any of the BLE modes.

Contrary to our expectations, our experiments show, that the guard time for beginning to receive an acknowledgment (`AckWait`) is too low to successfully receive an acknowledgment in coded BLE. Thus, we increase `AckWait` from 400 to $1000\ \mu s$ for coded BLE.

From our dissection of TSCH (cf. Section III-A), we know that we need to reserve the time equivalent to 133 bytes and 75 bytes for `MaxTX` and `MaxAck`, respectively. While BLE would allow for packets with a longer payload, we stick to the maximum payload size of IEEE 802.15.4, as this does not require major changes in TSCH. Moreover, packets with a longer time on air are more susceptible to short bursts of interference.

1) Packet Format: After setting the basis for the payload length of a packet and of an acknowledgment, we next discuss the BLE packet structure to be able to calculate the timing. This packet structure differs significantly between the coded and uncoded modes of BLE. While we want to send valid BLE packets in any case, we also want to enable a radio of a device that is not running TBLE to discard the packet as quickly as possible.

For the uncoded modes of BLE (1 Mbps and 2 Mbps), we choose a custom, application-specific access address and transmit the TSCH packet as is as the BLE advertisement packet's PDU. This should enable other BLE devices to discard the packet, as it is not using the standard advertisement access address. Moreover, it is very unlikely that a device has a connection with the same access address, we use for TBLE.

For the coded modes (125 kbps and 250 kbps), we cannot use a custom access address, as the radio we use is not able to receive coded BLE packets with an access address besides the standard advertisement access address. Instead, we have to choose a different way to create a BLE compliant but easily discardable packet. Thus, we create a standard advertisement packet with a PDU with one advertisement data (AD) structure containing the TSCH frame. We set the first byte of the advertisement header to `0x90`, using an undefined PDU type, which might enable other radios to discard the packet right away. The AD structure we send uses the AD type `0xFF`, identifying the subsequent data as manufacturer specific.

Thus, for both modes, we deviate slightly from sending correct BLE packets to enable only devices running TBLE to further process the received data.

2) TBLE Timing: After the considerations regarding the BLE packet structure and the payload length, we can derive the

TABLE II: TSCH/TBLE timeslot timings and effective data rates.

Mode	IEEE 802.15.4	BLE 125k	BLE 500k	BLE 1M	BLE 2M
MaxTX (μs)	4256	9532	2590	1088	548
MaxAck (μs)	2400	5520	1662	624	316
AckWait (μs)	400	1000	1000	400	400
Sum (μs)	9776	7372	18172	4832	3984
Timeslot (μs)	10000	7500	18500	5000	4000
Effective data rate (kbps)	101.6	135.5	54.9	203.2	254

timing for `MaxTX` and `MaxAck`, which we show in Table II. Similarly, to IEEE 802.15.4, we round up the timeslot lengths to the next multiple of 500 μs. The timeslot lengths vary between 4 and 18.5 ms. In addition to the timeslot lengths, we also show the effective data rates, which we calculate for a packet of maximum size (127 data bytes) in kbps with $\frac{127*8}{TimeslotLength}$.

C. Packet duration

To enable TSCH to start the `RxAckDelay` and `TxAckDelay` at the correct time, we need to adjust the TSCH packet duration computation (cf. Section III-B). Setting the byte air time for the payload is straight-forward and can be directly derived from the bitrate. However, the radio PHY overhead is only easily identifiable for the uncoded mode (10 bytes). For the coded modes of BLE, we identify 4 bytes overhead in the payload plus certain fixed time overheads. These are $296\mu s$ for FEC block 1, and $54\mu s$ or $216\mu s$ for CRC and Term 2 at the end of FEC block 2 for a BLE data rate of 500 kbps or 125 kbps, respectively (cf. Fig. 1).

D. Time Synchronization

As we discuss in Section III-C, TSCH uses time stamping functionalities of the radio, for precise time synchronization, especially the timestamp of the start of frame delimiter (SFD). As BLE PHY packets do not contain an SFD between the preamble and the start of the frame, we cannot use the same timestamp for time synchronization as in the case of IEEE 802.15.4. The time stamping points the nRF52840's radio offers in BLE mode are *address sent/received*, *payload sent/received*, and *packet sent/received*. When comparing the accuracy of the different time stamps between two devices located next to each other and connected to a logic analyzer, our results strongly suggest that the timestamp of the *address sent/received* event is most suitable for TSCH time synchronization. Similar to IEEE 802.15.4, we use this timestamp to compute the timestamp of the end of the preamble. For that, we subtract the transmission time for the address from the timestamp. Moreover, our experiments show, that contrary to our assumptions, the radio is not done transmitting its preamble by the end of `TxOffset`. Thus, to synchronize to the intended time, we introduce an additional (negative) timing offset initiating the transmission in TSCH earlier to ensure the preamble being transmitted exactly at the assumed time.

E. Hopping Sequences

For TBLE, we need hopping sequences using the BLE channels. For comparison with IEEE 802.15.4, we replicate the 1, 2, 4, and 16 channel hopping sequences using the BLE channels that best match the used IEEE 802.15.4 channels. In addition, we generate a 40 channel hopping sequence containing all BLE channels and an intermediate one using 32 channels. For the 40 channel hopping sequence, we use the 9-bit linear feedback shift register defined by the TSCH standard. For the 32 channel hopping sequence, we take the 40 channel hopping sequence and remove every fifth channel.

F. Standard-compliance Discussion

To end the description of our design, we want to discuss the standard compliance of TBLE with both TSCH and BLE. To our knowledge, we are fully compliant with TSCH as long as we use the maximum 40-channel hopping sequence. None of the timeslot timing numbers exceed 2 bytes and thus fit into the TSCH Information Elements (IEs) to transmit the timeslot timing as part of an Enhanced Beacon (EB). Moreover, the longer hopping sequence also doesn't exceed the length allowed in IEs of EBs.

For BLE, we deviate somewhat from the standard. The individual packets use an access address not standard for advertisements or use an undefined PDU type. Moreover, we use all BLE channels and do not stick to the primary advertisement channels while sending advertisement packets. Our reasoning is to enable BLE radios to discard our packets as quickly as possible after reception, but still enabling BLE radios running TBLE to join the TSCH network.

V. Evaluation

After discussing the design of TBLE and the necessary adaptions and extensions towards TSCH, we experimentally evaluate its performance. We split the evaluation into two parts. Firstly, we study the connectivity of the same physical network using different PHYs. We compare the four BLE PHY configuration with the IEEE 802.15.4 PHY as our baseline. Afterward, we evaluate and compare the performance of TSCH multi-hop communication over BLE and IEEE 802.15.4. For that, we use the well established autonomous TSCH scheduler Orchestra [14]. Orchestra is integrated into Contiki-NG and is a standard benchmarking solution for Contiki-NG. Orchestra builds upon the routing protocol RPL [5] that builds a routing tree on top of TSCH.

Setup. For our evaluation, we use our local testbed of 20 nodes (see Fig. 3). The testbed spans the top most floor of a university building (500 m^2) with offices and lab rooms and shares the wireless spectrum with other networks including Wi-Fi. Each node is equipped with, i.a., a nRF52840 DK board [23] and a Raspberry Pi 3B+ as observer for collecting and processing our evaluation logs. The nRF52840, which our design targets, is a Cortex-M4 microcontroller with 64 MHz clock speed, 1 MB flash and 256 KB RAM and a radio supporting, i.a., both IEEE 802.15.4 and BLE 5.

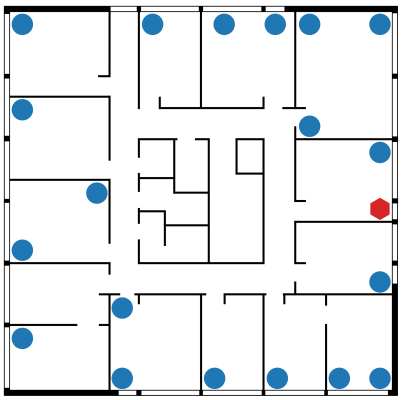


Fig. 3: Local testbed of **500m^2**. Red hexagon: TSCH PAN coordinator/Orchestra root node; Blue circles: network participants

Metrics. Throughout this evaluation, we look at the following metrics. For the connectivity, we compare the single-hop reachability of nodes and the *Expected Transmission Count (ETX)* [13] for a node's neighbors. The *ETX* is the inverse of the *Packet Reception Rate (PRR)*. For our performance evaluation with Orchestra, we look at reliability (*Packet Delivery Rate (PDR)*), latency and radio duty cycle for the different modes.

A. Reachability

Scenario. We investigate the reachable number of nodes for each node in our testbed for the four BLE PHYs and compare it to the IEEE 802.15.4 PHY. We quantify the performance both in terms of neighbors and in terms of average neighbor ETX. We perform this evaluation for 7 different transmit powers between -30 dBm and 0 dBm. For each transmit power we run a 15-minute experiment using one of the available hopping sequences (1 channel, 4 channel, and 16 channels for all PHYs, plus 32, and 40 channels for the BLE PHYs). We use the neighbor discovery mode of the centralized TSCH scheduler MASTER [18].

Results. Fig. 4 and Fig. 5 show the evaluation results for the 5 PHYs. Fig. 4 shows the median number of nodes reachable for the different transmit powers. The plots show a CDF for each of the transmit powers. Each line shows the percentage of nodes in the testbed that has a certain number of neighbors. For example, the left most visible line in Fig. 4d, shows that for the 2 Mbps BLE mode at a transmit power of -20 dBm, 7 nodes can reach a single other node, 6 nodes can reach 2 other nodes, and 2, 3, and 1 nodes can reach 3, 4, and 5 nodes, respectively.

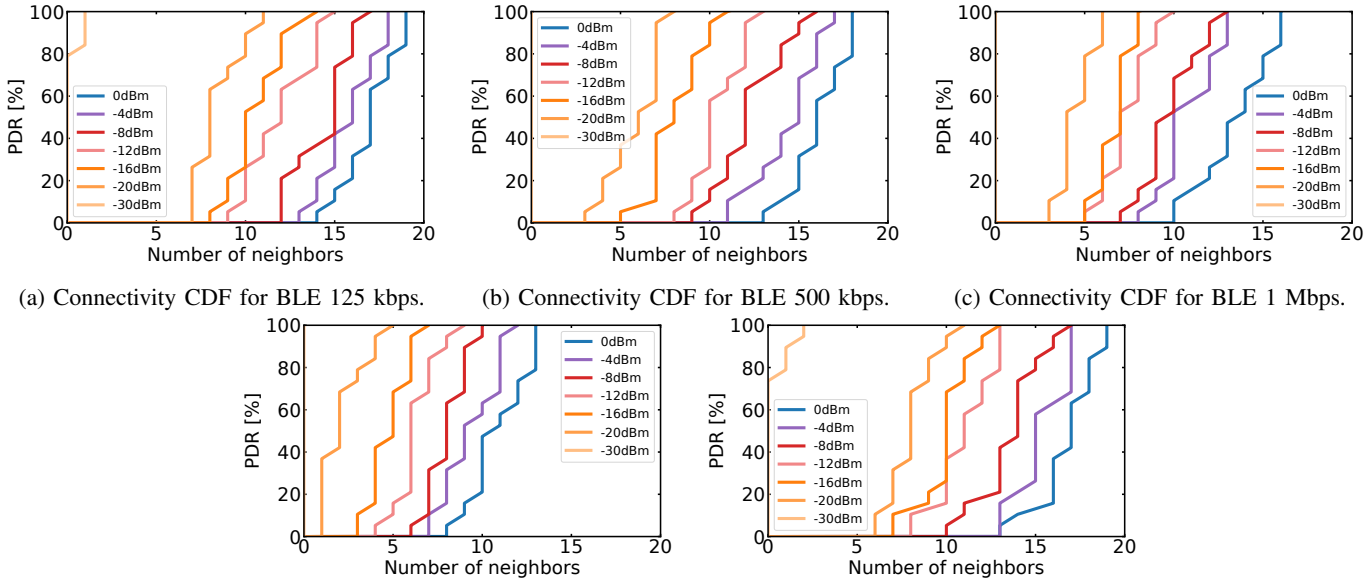
Both the BLE mode with a bitrate of 125 kbps and the IEEE 802.15.4 mode have nodes at a transmit power of 0 dBm that can reach all other 19 nodes in the testbed. Moreover, these are the only modes that can reach any neighbors at all, at a transmit power of -30 dBm. This means that only for these two modes, a TSCH network forms at -30 dBm. Generally speaking, the BLE mode with a bitrate of 125 kbps has a similar number of reachable neighbors as the IEEE 802.15.4 mode. The 500 kbps mode has a slightly lower number of reachable nodes than the 802.15.4 mode. The 2 Mbps BLE mode has with 13 possible neighbors a much lower maximum number of reachable nodes. Moreover, at a transmit power of -20 dBm, TSCH using the 2 Mbps BLE mode can just barely form a mesh network.

Fig. 5 shows the average ETX and the standard deviation for a node's transmission to all of its reachable neighbors for 6 out of the 7 transmit powers. We exclude the transmit power of -30 dBm as it is hardly usable at all. Contrary to the number of reachable nodes, the average ETX to the neighbors is significantly higher for 125 kbps BLE (Fig. 5a) than for IEEE 802.15.4 (Fig. 5e). We suspect this behavior to come from the robustness of the modulation scheme. While both modes can reach similar numbers of neighbors, the BLE modulation scheme (GFSK) should be more affected by interference, leading to the higher expected number of transmissions to successfully reach its neighbors. The other BLE modes (Fig. 5b – 5d) reach a lower number of nodes, but have a better connection to those.

From these results, we derive that -16 dBm (dark orange line) is the lowest usable transmit power to form a proper mesh network for all PHYs. Moreover, -8 dBm (dark red line) is the transmit power with a medium number of neighbors for all PHYs. Therefore, we will use these two transmit powers for our performance evaluation below.

B. Performance Evaluation

Scenario. We evaluate the performance of mesh multi-hop communication of TBLE in comparison with our baseline TSCH over IEEE 802.15.4 using the autonomous scheduler Orchestra at transmit powers of -8 and -16 dBm. We run multiple experiments with a total runtime of 4 hours for each transmit power and mode. As Orchestra builds upon the *Routing Protocol for Low-Power and Lossy Networks (RPL)* [5], we compare different routing networks with each other. Thus, while using the same physical deployment for our experiments, RPL builds a routing network outside our control, optimized for the prevalent situation. Therefore, the results might differ from our intuitive ideas. Yet, using an autonomous scheduler and a routing protocol like RPL assures us to use the best routes for each protocol and mode. This allows us to compare the effective performance of each protocol. Orchestra sends once a second a packet to a random node of the network, and the node sends back a reply on reception of the packet. We evaluate round-trip and one-way performance of this communication using latency and reliability as our metrics. For the round-trip latency, we use the node's measured time between sending and receiving the packet. For the one-way latency we use slot counts, which we convert to time. We base the latter on the slot count, as the timestamping of the serial interfaces of our testbed are not time synchronized to the required degree. Lastly, we also compare the radio duty cycle of the different PHY modes.



(a) Connectivity CDF for BLE 125 kbps.

(b) Connectivity CDF for BLE 500 kbps.

(c) Connectivity CDF for BLE 1 Mbps.

(d) Connectivity CDF for BLE 2 Mbps.

(e) Connectivity CDF for IEEE 802.15.4.

Fig. 4: Evaluation of the nodes' reachability (median reachability CDF for the different hopping sequences). The bitrate has a clearly visible influence on the communication range and thus the number of neighbors.

Results. Fig. 6 shows the performance of Orchestra for all modes. Fig. 6a and 6d show that the round-trip latency for most BLE modes is most of the time better than for IEEE 802.15.4 with an average improvement of 10 to 20%. For 125 kbps BLE (dark red line) this is most visible. While this mode has the longest slot lengths, it seems that RPL uses fewer hops for the routing, which benefits the latency in most cases, but also leads to an increased latency for some packets. We can see a similar routing benefit for 500 kbps BLE and 1 Mbps BLE in Fig. 6a.

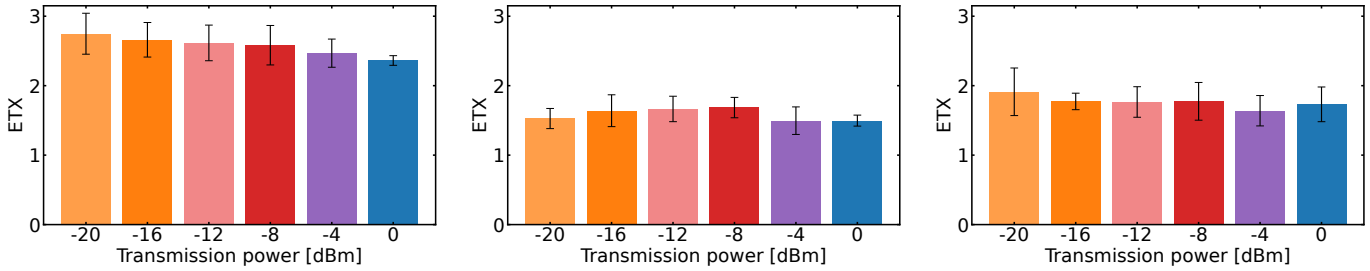
Overall, reliability is comparable for most modes. However, at a transmit power of -16 dBm, both the 125 kbps BLE mode and the 2 Mbps BLE mode do not reach maximum reliability. While we can expect that for the former the routing is not favorable to reach highest reliability, this should not be the case for the latter. For the latter (2 Mbps), we observe that the network formation takes already half an hour and therefore, we can expect the network to be not as stable as in the other modes and thus is unable to achieve maximum reliability.

In Fig. 6c and 6f we compare the duty cycle of TBLE and TSCH over IEEE 802.15.4. We can see that the duty cycle increases with the bitrate. This can be expected, as a higher bitrate allows for a shorter slot duration to transmit the same amount of data. In Orchestra, a receiver listens in every slot. If the receiver does not encounter a packet, it keeps its radio on for the `RxWait` guard time. Therefore, if we have twice the number of slots (e.g., 1 Mbps BLE vs. IEEE 802.15.4), we have a doubling in radio-on-time for the majority of slots, those in which no communication takes place. When we multiply the duty cycle with the respective slot length, the resulting values are almost the same. When comparing the radio duty cycle between a network with a transmit power of -8 dBm (Fig. 6c) and a transmit power of -16 dBm (Fig. 6f), we see a minor increase for all modes of 0.03 to 0.5 percantage points. This confirms that the predominant factor for the duty cycle are slots without communication.

VI. Related Work

In recent years, several works looked into extending the use of Time-Slotted Channel Hopping (TSCH) into the field of other radio PHYs. Brachmann et al. [11] study the possibility of using TSCH with subGHz PHYs. The authors show its feasibility when adapting the TSCH timeslot timings. Moreover, they show the possibility of combining multiple PHYs in the same TSCH schedule. Rady et al. [27] build with g6TiSCH another work combining multiple PHYs in a single TSCH network. They perform modifications along the 6TiSCH stack to allow an intelligent choice which PHY to use in a TSCH slot. Carhacioglu et al. [12] study the co-existence of TSCH and BLE and propose a system with a common TSCH and BLE orchestrator to overcome cross-technology interference. Hajizadeh et al. [17] build a simulation framework analyzing the coexistence and amount of expectable collisions for coexisting BLE and TSCH networks.

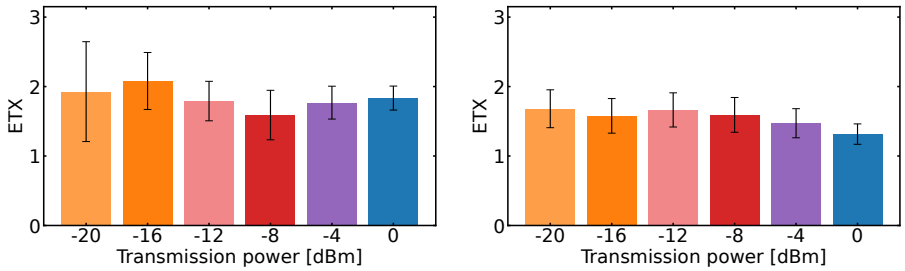
Baddeley et al. [6] take an approach of combining BLE and TSCH. They propose 6TiSCH++ which uses the standard TSCH slots over IEEE 802.15.4 for data communication, but replaces the beaconing slots with concurrent transmissions over BLE. In 6TiSCH++, multiple subsequent concurrent BLE transmissions fit into one TSCH slot and allow for a faster transmission of control information for the TSCH network. Concurrent Transmissions (CT) are a well explored communication paradigm in IEEE 802.15.4 and their feasibility for



(a) Average ETX to neighbors for BLE 125 kbps.

(b) Average ETX to neighbors for BLE 500 kbps.

(c) Average ETX to neighbors for BLE 1 Mbps.



(d) Average ETX to neighbors for BLE 2 Mbps.

(e) Average ETX to neighbors for IEEE 802.15.4 (250 kbps).

Fig. 5: Evaluation of the nodes' reachability showing the average ETX to a node's neighbors. IEEE 802.15.4 has the best connectivity to its neighbors.

multi-hop communication on top of BLE were shown by BlueFlood [4], [22].

On the side of pure BLE networking, Patti et al. [25] devise a connection-oriented protocol for real-time mesh communication on top of BLE that uses subnetworks, each with a central node and several peripheral nodes. The networks are linked through peripheral nodes shared between two subnetworks. Leonardi et al. [21] extend and implement that solution. In contrast, we build a single mesh network allowing communication between any two nodes. With Bluetooth Mesh [7], [8], the Bluetooth SIG standardized a mesh networking protocol on top of BLE using managed flooding. Aijaz et al. [3] experimentally study its performance using the same hardware we use for TBLE. Leonardi et al. [20] propose RESEMBLE, a protocol for Bluetooth Mesh enabling TDMA-based communication with time slots and clock synchronization over Bluetooth Mesh to allow for real-time communication in Bluetooth Mesh networks. Contrary to that solution, we create a routing-enabled solution utilizing a well established time-slotted and time-synchronized MAC layer protocol. Petersen et al. [26] extend BLE to enable efficient multi-hop IPv6 over BLE and Lee et al. [19] bring the RPL routing protocol to BLE.

Our approach to mesh networking on top of BLE is to some extent in line with these networking approaches, but differs in certain aspects. On the one hand, we bring TSCH to another PHY and study a mesh networking approach on BLE. On the other hand, our approach differs from the approaches above in that we combine an established MAC layer (TSCH) for multi-hop routing with a widespread radio communication technology (BLE), enabling routed mesh communication on top of BLE.

VII. Conclusion

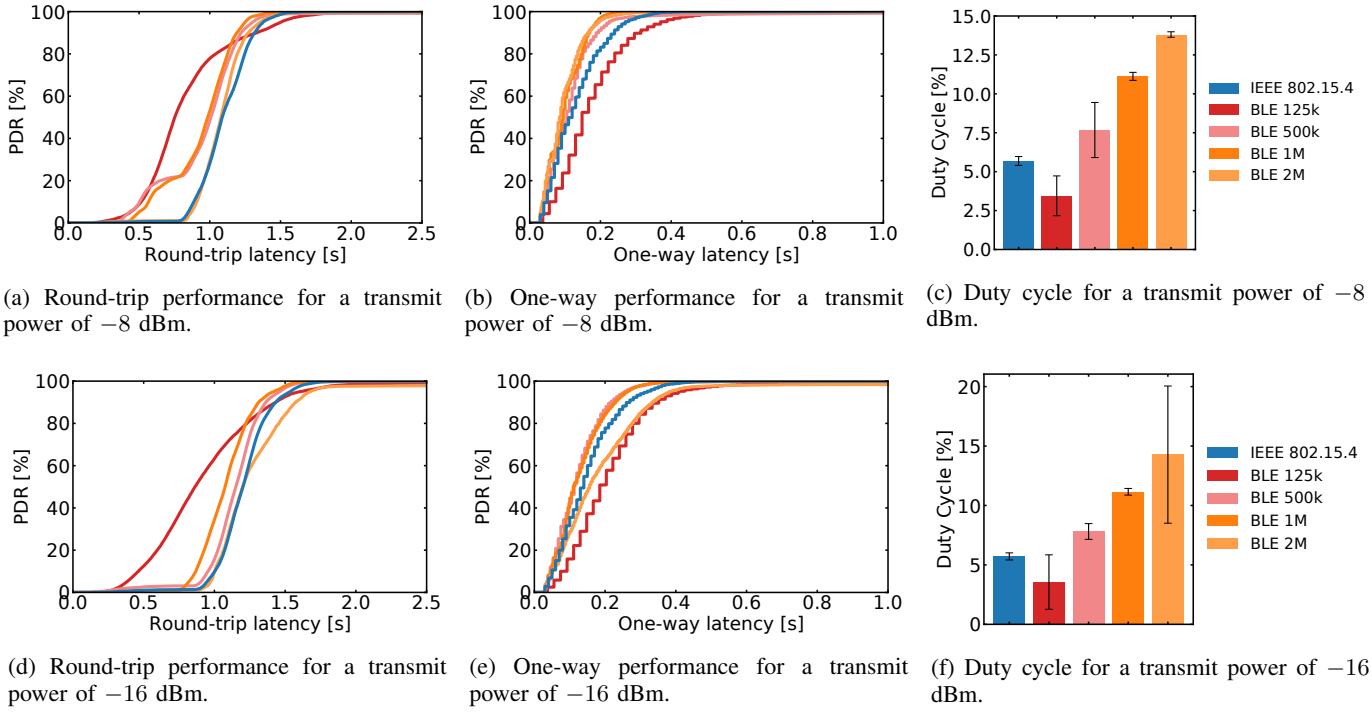
Bluetooth Low Energy is a widely used communication protocol in the IoT. For advanced communication systems covering larger areas, mesh communication is necessary. While BLE offers Bluetooth Mesh, it lacks a routed mesh communication protocol. With this work, we introduce TBLE, a combination of BLE and TSCH, and a replacement for IEEE 802.15.4. We show that TSCH and the 6TiSCH network stack are a viable candidate for routed mesh communication over BLE. Moreover, with the larger amount of available frequency slots in comparison with IEEE 802.15.4, and the possibility for shorter time slots due to higher bit rates, BLE and TBLE might even be favorable over IEEE 802.15.4 and TSCH in latency-critical applications. Moreover, BLE supports longer packets, which could increase the effective bit rate even further and lead to an even more efficient use of the wireless spectrum.

VIII. Acknowledgments

The authors would like to thank Tobias Schramm for his help implementing the BLE driver, this work builds upon.

References

[1] "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN)," *IEEE Std 802.15.4-2003*, 2003.



(a) Round-trip performance for a transmit power of -8 dBm.

(b) One-way performance for a transmit power of -8 dBm.

(c) Duty cycle for a transmit power of -8 dBm.

(d) Round-trip performance for a transmit power of -16 dBm.

(e) One-way performance for a transmit power of -16 dBm.

(f) Duty cycle for a transmit power of -16 dBm.

Fig. 6: Evaluation of the performance of Orchestra for all PHY layers. We show the performance for two transmit powers (-8 dBm and -16 dBm) and the respective radio duty cycles during operation. We show the legend for all plots in Fig. 6c and 6f. The BLE modes usually achieve better round-trip latency performance with a slight instability of the highest bitrate (2 Mbps) at a transmit power of -16 dBm.

[2] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," IEEE, Tech. Rep., 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6185525/

[3] A. Aijaz, A. Stanoev, D. London, and V. Marot, "Demystifying the Performance of Bluetooth Mesh: Experimental Evaluation and Optimization," in *IEEE Wireless Days (WD)*, 2021.

[4] B. Al Nahas, S. Duquennoy, and O. Landsiedel, "Concurrent Transmissions for Multi-Hop Bluetooth 5," in *EWSN*, 2019, pp. 130–141.

[5] R. Alexander, A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6550

[6] M. Baddeley, A. Aijaz, U. Raza, A. Stanoev, Y. Jin, M. Schuß, C. A. Boano, and G. Oikonomou, "6TiSCH++ with Bluetooth 5 and Concurrent Transmissions," in *EWSN*, 2021.

[7] Bluetooth SIG, "Mesh Model 1.0," 2017. [Online]. Available: https://www.bluetooth.com/specifications/specs/mesh-model-1-0/

[8] ——, "Mesh Profile 1.0," 2017. [Online]. Available: https://www.bluetooth.com/specifications/specs/mesh-profile-1-0/

[9] ——, "Bluetooth Core Specification v5.2," 2019.

[10] ——, "2021 Market Update," 2021. [Online]. Available: {https://www.bluetooth.com/wp-content/uploads/2021/01/2021-Bluetooth_Market_Update.pdf}

[11] M. Brachmann, S. Duquennoy, N. Tsiftes, and T. Voigt, "IEEE 802.15.4 TSCH in Sub-GHz: Design Considerations and Multi-band Support," in *IEEE LCN*, 2019.

[12] O. Carhacioglu, P. Zand, and M. Nabi, "Cooperative Coexistence of BLE and Time Slotted Channel Hopping Networks," in *IEEE PIMRC*, 2018.

[13] D. S. J. De Couto, "High-Throughput Routing for Multi-Hop Wireless Networks," PhD thesis, MIT, 2004. [Online]. Available: https://pdos.lcs.mit.edu/papers/grid:decouto-phd/thesis.pdf

[14] S. Duquennoy, B. A. Nahas, O. Landsiedel, and T. Watteyne, "Orchestra," in *ACM SenSys*, 2015.

[15] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient network flooding and time synchronization with Glossy," in *ACM/IEEE IPSN*, 2011.

[16] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, "Low-power wireless bus," in *ACM SenSys*, 2012.

[17] H. Hajizadeh, M. Nabi, M. Vermeulen, and K. Goossens, "Coexistence Analysis of Co-Located BLE and IEEE 802.15.4 TSCH Networks," *IEEE Sensors Journal*, vol. 21, no. 15, pp. 17 360–17 372, aug 2021.

[18] L. Harms and O. Landsiedel, "MASTER: Long-Term Stable Routing and Scheduling in Low-Power Wireless Networks," in *IEEE DCOSS*, 2020.

[19] T. Lee, M.-S. Lee, H.-S. Kim, and S. Bahk, "A synergistic architecture for rpl over ble," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, 2016.

[20] L. Leonardi, L. L. Bello, and G. Patti, "RESEMBLE: A Real-Time Stack for Synchronized Mesh Mobile Bluetooth Low Energy Networks," *Applied System Innovation*, vol. 6, no. 1, p. 19, jan 2023.

[21] L. Leonardi, G. Patti, and L. L. Bello, "Multi-Hop Real-Time Communications Over Bluetooth Low Energy Industrial Wireless Mesh Networks," *IEEE Access*, vol. 6, pp. 26 505–26 519, 2018.

[22] B. A. Nahas, A. Escobar-Molero, J. Klaue, S. Duquennoy, and O. Landsiedel, "BlueFlood," *ACM Transactions on Internet of Things*, vol. 2, no. 4, pp. 1–30, jul 2021.

[23] Nordic Semiconductor, "nRF52840 DK." [Online]. Available: https://www.nordicsemi.com/Products/Development-hardware/nrf52840-dk

[24] G. Oikonomou, S. Duquennoy, A. Elsts, J. Eriksson, Y. Tanaka, and N. Tsiftes, "The Contiki-NG open source operating system for next generation IoT devices," *SoftwareX*, vol. 18, p. 101089, jun 2022.

[25] G. Patti, L. Leonardi, and L. L. Bello, "A Bluetooth Low Energy real-time protocol for Industrial Wireless mesh Networks," in *IEEE IECON*, 2016.

[26] H. Petersen, T. C. Schmidt, and M. Wählisch, "Mind the gap: Multi-hop ipv6 over ble in the iot," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21, 2021.

[27] M. Rady, Q. Lampin, D. Barthel, and T. Watteyne, "g6TiSCH: Generalized 6TiSCH for Agile Multi-PHY Wireless Networking," *IEEE Access*, vol. 9, pp. 84 465–84 479, 2021.