IPv6 over Bluetooth Advertisements: An alternative approach to IP over BLE

Hauke Petersen[1,3], János Brodbeck[1], Thomas C. Schmidt[2], Matthias Wählisch[3,4,1]
[1]Freie Universität Berlin, [2]HAW Hamburg, [3]TU Dresden, [4]Barkhausen Institut
hauke.petersen@fu-berlin.de, janos.brodbeck@posteo.de, t.schmidt@haw-hamburg.de, m.waehlisch@tu-dresden.de

Abstract

The IPv6 over Bluetooth Low Energy (BLE) standard defines the transfer of IP data via BLE connections. This connection-oriented approach provides high reliability but increases packet delays and introduces substantial overhead to manage BLE connections. To overcome these drawbacks, we present the design and implementation of IPv6 over BLE advertisements, a standard-compliant connection-less approach. We deploy our proposal on low-power IoT hardware and comparatively measure key network performance metrics in a public testbed. Our results show that IP over BLE advertisements offers network performance characteristics complementary to IP over connection-based BLE, trading lower reliability for shorter latency.

1 Introduction

The Internet of Things (IoT) is highly fragmented [5]. In the low-power wireless IoT, heterogeneous link layer technologies compete, each requiring dedicated (smart) gateways to connect to the Internet. Bluetooth Low Energy (BLE) is the most deployed low-power radio technology today [18] and the IP over BLE standard [14, 15, 30] allows to seamlessly connect BLE devices to the Internet. Furthermore, BLE offers best in class low-power characteristics [9, 27, 40] as well as reliable network performance [35], making it a promising default link layer in the low-power IoT. Enabling BLE as go-to lower layer for low-power wireless IP networks has the potential to mitigate fragmentation in the IoT.

IP over BLE, however, works on top of BLE connections, which leads to the following three disadvantages. First, before exchanging IP data, any node must open BLE connections to one or more adjacent peers. Managing these connections automatically poses overhead on BLE nodes. Furthermore, there are no protocols for automated BLE connection management available yet. Second, the number of concurrent BLE connections is typically limited to ≈15 peers due to restrictions in memory and radio scheduling. Third, the current IP over BLE standard [30] increases packet delays because BLE connections are time-slotted. Typical latencies of IP over BLE networks are substantially larger compared to networks based on carrier-sense multiple access (*e.g.,* IEEE802.15.4) [35].

In order to mitigate the disadvantages of the current IP over BLE standard, we propose exploring the transfer of IP data using the connection-less mode of BLE. We do not aim for replacing the existing connection-oriented IP over BLE design but offer an alternative based on the same technology, to allow IoT developers to optimize deployments depending on requirements. Possible new scenarios include use cases where nodes suffer from the overhead and difficulty of establishing BLE connections. One use case is the provisioning of devices. IP over connection-less BLE can be used to effortlessly connect devices to remote provisioning services without the overhead of managing BLE connections. Another possible scenario is the massive deployment of nodes in small areas that would suffer from the limited connection count per peer. One example for this is the self-tracking of products on high density assembly lines that at certain stages need to report their state. A third field of use cases evolves around applications that demand low-latency interaction. An example of this is building lighting as targeted also by Bluetooth Mesh [17]. Here, fast user feedback, for example when using light switches, is desirable. As both connection-less and connection-based designs are using the same software (BLE and IP stacks) and hardware (radios) they can be deployed and run simultaneously, which finally will increase IoT use cases for BLE networks allowing a flexible mapping of services to communication mode at runtime.

In this work, we present the protocol design and prototype implementation of IPv6 data over connection-less BLE. We utilize the extended advertisements, which are introduced in Bluetooth version 5.0 [16]. Extended advertisements have the advantage of offering a MTU of up to 65 Kbytes through packet fragmentation capabilities provided by a BLE controller. They are, thus, able to carry full IPv6 packets with a minimum MTU of 1280 bytes on top of a lean software system. In contrast to this, legacy BLE advertisements would allow only for a maximum payload of 31 bytes per packet. This small MTU would require complex fragmentation schemes implemented on an intermediate layer between

IP and BLE in addition to 6LoWPAN-based header compression [7, 29].

We systematically measure key performance metrics in practice based on 15 low-power BLE nodes in the FIT IoTlab testbed [3]. We analyze reliability, latency, and energy consumption in different single- and multi-hop network topologies and compare them to the performance of connection-based IP over BLE networks. Our results show that using (connection-less) BLE advertisements offers lower latency (on average 1.5× to 5× lower for comparable configurations) but less reliability (1% to 80% packet loss vs. <0.01%) and increased power consumption (radio always on).

Currently, Bluetooth Mesh [17] is the only standard to transfer (proprietary) data over connection-less BLE. In contrast to our proposal, however, Bluetooth Mesh does not support arbitrary IP packets but is limited to the flooding of specific, small data frames (<20 bytes) and does not support fragmentation. Bluetooth Mesh aims for vendor-specific simplified scenarios, whereas our proposal targets flexible Internet-like deployments.

In summary, we make the following contributions:

1. The first, standard-compliant design to transfer IPv6 data over BLE extended advertisements. (§ 3.1)
2. A publicly available, open source implementation based on the operating system RIOT and the BLE stack NimBLE. (§ 3.2)
3. Reproducible experiments conducted on real-world hardware. (§ 4)
4. A comparative performance evaluation including network and system measures to show protocol mechanics in contrast to connection-based IP over BLE networks. (§ 5–§ 6)

All artifacts, including implementations and raw data, are publicly available, details see Appendix A.

2 Background

BLE supports three modes to transfer data: the connection-less *legacy advertising mode*, the connection-less *extended advertising mode*, and the *connection-based mode*. Connection-less communication is usually used to enable the discovery of services and to broadcast data for further processing to unknown peers. Connection-based communication aims for communication between direct peers, *e.g.,* in the IP over BLE standard [15,30]. This section briefly presents core background on all three modes with a focus on embedding data.

2.1 Connection-less BLE Communication

Legacy advertising is used in Bluetooth Mesh [17] and the extended advertising mode was introduced in Bluetooth 5.0 [16].

Legacy Advertising. Legacy advertising supports a maximum payload of 31 bytes. The advertisement packets are sent periodically in so-called *advertising events* during an *advertising interval*, depending on the configuration between 20ms and 10.48s, see Figure 1. To counteract unwanted synchronization between nodes, a random jitter between 0ms and 10ms is added between each connection event. Each

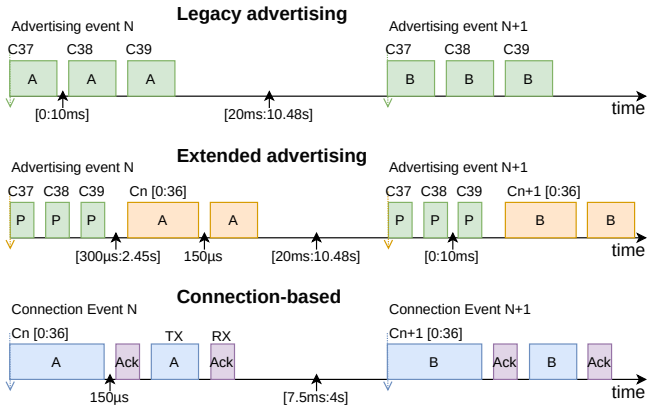


Figure 1: BLE packet flow when transferring payloads A and B using legacy and extended advertisements as well as connection-based communication.

advertisement packet is sent via the *primary advertisement channels*, three predefined channels that are exclusively reserved for advertisements to achieve some level of robustness. This mode is unidirectional (no link layer acknowledgments) and unmanaged (no CSMA scheme).

To receive advertisements, nodes listen periodically on one of the primary advertising channels (*scan event*) based on the *scan interval*. An active radio in RX mode is expensive in terms of energy. The Bluetooth standard, thus, allows the receiver during each scan event to only activate the radio during the *scan window*. If the scan window is shorter than the advertising interval, advertising packets might get lost. There exist a number of approaches on how these parameters can be optimized to balance energy usage and delivery probability [26, 37, 38].

In common advertising use-cases (*e.g.,* beaconing), devices use a fixed payload. When considering less predictable application data such as carried in IP packets, this data should preferably be sent within a single advertising event. Since there is no guarantee that an advertising event is received, a single payload is typically transmitted in multiple advertising events, hence implementing a fixed number of link layer packet retransmissions. This is, for example, applied in Bluetooth Mesh, which defaults to carry the same payload in 5 connection events [17].

When transmitting IP data using legacy advertisements, the limited payload becomes a major bottleneck. To encapsulate IP data into the payload of advertising packets, an additional advertising data header of at least 6 bytes is required, leaving only 25 bytes for IP data. Even when using header compression techniques (*e.g.,* defined in 6LoWPAN [7, 23]), packet fragmentation would then be needed.

Extended Advertising. Extended advertising allows for larger payloads and is based on legacy advertising. Instead of carrying data in (very limited) packets via the primary channels, extended advertising uses these packets to refer to one of 37 data channels and a start time. The actual payload is then sent at the specified start time on the given data channel in one or more chained data packets (see Figure 1).

By containing only a short pointer, the packets sent on the 3 advertising channels need less air time for transmission. This reduces collision probability on those potentially crowded channels. The collision probability for data packets is reduced by utilizing all 37 data channels for their transmission.

By splitting the payload over multiple chained packets, extended advertising allows to transfer up to 65 Kbytes in a single advertising event. Fragmentation and reassembly into link layer data packets is done by the Bluetooth controller, which relieves higher layers from implementing fragmentation schemes to transfer full IPv6 MTUs.

2.2 Connection-based BLE Communication

In contrast to advertisements, which are transmitted in the broadcast domain, BLE connections are always point-to-point. In a connection between two nodes, one node acts as *connection coordinator* while the peer node is the *connection subordinate*.[1] Similar to advertising events, the communication in the connection-based mode is structured into *connection events*. Each connection event consists of at least a single data packet exchange between the coordinator and subordinate. This can be repeated multiple times in the same event until no payload is left to send or the next connection event starts. If one of the peers has no data to send, it will send empty packets. Each connection event takes place on one of the 37 available data channels.

BLE connections provide a point-to-point service guaranteeing first-in-first-out, in-order, and complete data delivery. To achieve this, data packets are retransmitted on the link layer until they are acknowledged. If by either side no valid packet is received during a specific amount of time, the connection is considered lost and is closed. Consequently, as long as connections are active, there is no packet loss on the link layer [35]. The IP over BLE standard [13,15,30] is using this connection-based mode.

3 IPv6 over BLE Extended Advertisements

This section describes our design to enable connectionless IPv6 communication over BLE. The core idea is to carry IPv6 packets in the payload of BLE extended advertisements, using either directed advertisements or undirected advertisements to transmit unicast or multicast data, respectively.

In the remainder of this paper, we will denote our proposed design *IP-BLE-Adv*, while the standardized IP over connection-based BLE is denoted *6BLEMesh*.

3.1 Protocol Design

Requirements. Wherever applicable, our proposal shall comply with the *6BLEMesh* standard. In detail,

1. the support of an MTU of $\geq$ 1280 bytes across all links to prevent the fragmentation of IPv6 packets [10] and to utilize the built-in functionality of extended advertisements.
2. the use of 6LoWPAN header compression [29].
3. the support of unicast and multicast messages, where multicast messages can be transmitted via broadcast (similar to IEEE 802.15.4) but unicast messages should be filtered on the link layer.

[1]The terms "coordinator" and "subordinate" used in this paper diverge from Bluetooth specifications, to support non-discriminatory language.

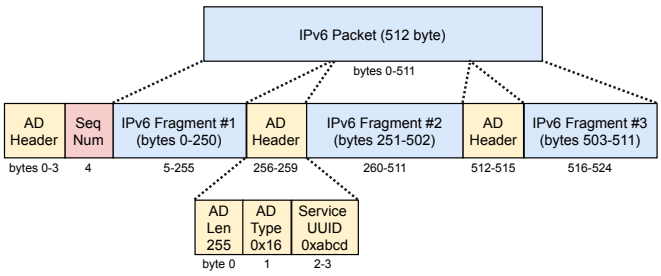


Figure 2: Encoding and fragmentation of an IPv6 packet (blue) into a BLE PDU consisting of Advertising Data (AD) segments, each starting with a dedicated AD header (yellow). The first segment contains the custom sequence number (red) to detect duplicates.

Advertising Types. All data is transferred using BLE extended advertisements. BLE supports different types of extended advertisements. In our design, all advertising packets are non-connectable and non-scannable. To maximize the use of built-in functionality of the BLE controller, we suggest to use non-directed packets to transfer multicast messages and directed advertising packets for unicast messages. The latter allows to utilize built-in packet filtering of an BLE controller, which prevents the need to filter on the upper layer and thus saves processing and energy.

Advertising Data Encoding. The Bluetooth standard requires the payload of advertising packets to be encoded in the advertising data (AD) format. This format comprises a list of one or more generic length-type-value fields with a 1 octet length field, a 1 octet type field, and a variable length data field. The structure of the data field depends on the value of the type field [19].

Encoding IP data packets into this format poses two challenges: (*i*) there is no predefined AD type to carry IP data and (*ii*) the maximum length of a single AD field is limited to 254 bytes. To address (*i*), we opted to use the AD type `Service Data - 16 bit UUID` (`0x16`). In this type, the data section of the AD field starts with a 2 octet Bluetooth service UUID and is followed by the IP payload including a prepended sequence number (see below). Due to the lack of a standardized service identifier for *IP-BLE-Adv* we use a custom, non-standard 16-bit UUID (*e.g.,* `0xabdc`).

Subtracting the 2 octet UUID, this design leaves 252 bytes of IP data in each AD segment. If an IP packet exceeds this size, it is split into multiple chained AD segments of the same type, where each segment is filled with the maximum possible amount of data before a new segment is started. The encoding of a 512 byte IPv6 packet is illustrated in Figure 2. As a result, each IP packet to be transferred, including a custom sequence number, is encoded into a continuous list of chained AD segments. This list is passed as a single block to the BLE stack. Fragmentation if this block into link layer packets and their reassembly is carried out trans-

parently by the BLE stack, based on configured HCI and link layer packet sizes. This BLE-level fragmentation and reassembly is independent of the structure of AD segments.

Data Reception. All nodes are expected to constantly listen for incoming packets to maximize reception reliability. This behavior is also specified for Relay nodes in Bluetooth Mesh or for nodes in CSMA/CA-based IEEE802.15.4 modes. It implies that the scan interval and scan window are equal.

Operating nodes in an always-on radio state per default conflicts with energy requirements, but it is sufficient to gain insights on the network performance of *IP-BLE-Adv* networks. Looking into concepts to improve energy efficiency by duty cycling the radio, like the friend role defined by Bluetooth Mesh, would be desirable. This is, however, not in scope of this work.

Data Transmission. A single extended advertising event consists of 3 advertising packets sent on each of the three advertising channels, as well as 1 or more chained packets containing the actual payload sent on one of the 37 data channels (see § 2). Ideally, each IP packet is sent in a single advertising event. Due to packet losses on the advertising channels, radio switching delays on the receiver, and the unidirectional nature of advertising packets, there is no guarantee that peer nodes receive a packet. Repeating the same payload in multiple consecutive advertising events does significantly improve the packet delivery ratio. Our design offers to configure this number of static retransmissions in the same style as Bluetooth Mesh does this for Relay nodes. In § 5.2 we present our findings towards optimizing this parameter.

Duplicate Detection. By retransmitting IP packets through multiple connection events, nodes potentially receive the same IP data packet multiple times. As we do not require duplicate detection on upper layers, BLE stacks must be able to detect these duplicates. The Bluetooth standard defines means for duplicate detection on advertising packets in the BLE controller. In practice, this duplicate detection is implemented using buffer memory holding the latest received packets to be compared with newly received ones. Especially on memory constrained devices, this buffer space is limited. We noticed that in environments with a high amount of advertising traffic, these buffers are not able to hold a sufficient amount of data and hence will fail to detect duplicates reliably.

As the built-in duplicate detection is unreliable, we introduce a 1 octet sequence number that is prepended before the IP payload. This sequence number is incremented for each IP payload that is transmitted. Each receiver maintains a table of link layer source addresses and last received sequence number for each neighbor. Incoming packets are then filtered by comparing the included sequence number against the last seen sequence number of packets from the same source address. If the sequence numbers are equal, the incoming packet must be a duplicate and is dropped.

3.2 System Design and Implementation

High-level Idea. Our proposed solution consists of a 6LoWPAN-enabled IP stack connected to a BLE stack, including a wrapper module between the stacks, taking care of forwarding and converting IP in both directions (see Figure 3). In the context of the IP stack, the wrapper module acts as a plain network interface exposing the 6-byte BLE addresses as link layer addresses following the specification in the IP over BLE standard [30]. The interaction between the wrapper module and the BLE stack is restricted to the *Generic Access Profile* (GAP). Next to the BLE controller, on the host side, our design requires only the *Logical Link Control and Adaption Layer Protocol* (L2CAP) layer to be implemented. In *6BLEMesh*, the presence of a defined *Generic Attribute Profile* (GATT) service is required by the *Internet Support Profile* (IPSP) [15]. The service aids the establishment of BLE connections, and hence is not needed in our implementation.

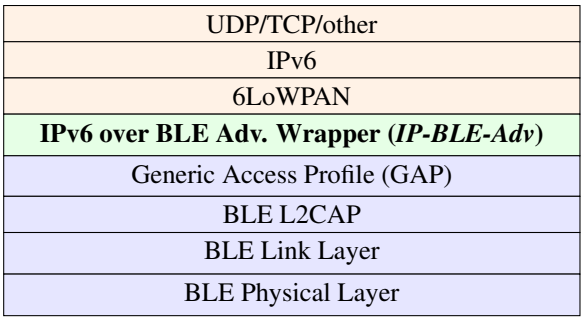


Figure 3: *IP-BLE-Adv* network stack. Our proposal represents a wrapper (green) between a common BLE stack (blue) and a common 6LoWPAN/IPv6 stack (orange).

Implementation. We implemented the proposed design in a fully open-source platform based on the RIOT operating system [2, 6] and the Apache NimBLE BLE stack [1]. For IPv6 networking, we utilize GNRC, the default IPv6 stack in RIOT [28].

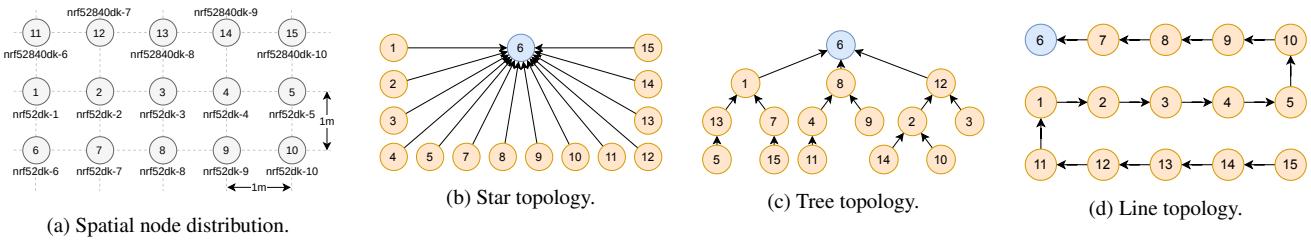
All IP data forwarding as well as link layer address handling is implemented in a single, separate software module, which allows to easily include or exclude *IP-BLE-Adv* in a RIOT image. In the context of the GNRC network stack, this module acts as a network interface by implementing the GNRC `netif` interface. To interact with the NimBLE BLE stack, the proposed module uses the NimBLE GAP API, in particular the `ble_gap_ext` family of functions.

On data reception, the NimBLE GAP API may, depending on the overall packet size, fragment incoming data into chunks and pass those chunks sequentially to the API user. The received IP data is encoded into one or more BLE advertising data format segments (see § 3.1). However, in order to be able to extract the IP packet, access to the full packet is needed. To achieve this, our implementation introduces an intermediate receive buffer that holds at least one full IPv6 MTU plus the overhead generated by the advertising data field headers, leading to an additional RAM usage of 1.3 Kbytes.

For *6BLEMesh*, we use the `nimble_netif` implementation that is included in RIOT [35].

4 Experiment Setup

We evaluate our proposal (see § 3) based on low-power hardware. This section describes the hardware and the



(a) Spatial node distribution.

(b) Star topology.

(c) Tree topology.

(d) Line topology.

Figure 4: Mapping of nodes in the FIT IoTlab testbed to network topologies deployed in our experiments, highlighting producer (orange) and consumer (blue) nodes.

testbed we use to deploy the different scenarios we analyse.

4.1 Hardware and Software Configurations

Hardware. We use the `nrf52840dk` and `nrf52dk` development boards from Nordic Semiconductor. Both feature 64 MHz ARM Cortex-M4F SoCs with on-chip BLE 5.2 support. In the context of this evaluation, the only relevant differences between both platforms are the RAM and ROM sizes: the `nrf52840dk` offers 1 Mbytes ROM and 256 Kbytes RAM while the `nrf52dk` offers 512 Kbytes ROM and 64 Kbytes RAM. Both SoCs, especially the `nrf52dk`, offer common memory and performance characteristics of modern IoT platforms [8].

Testbed. All raw data analyzed in this paper is the output of multi-node experiments conducted in the FIT IoTlab testbed [3] (`https://www.iot-lab.info`). IoTlab is an open testbed that anyone can access by creating an account free of charge.

We perform all experiments at the Saclay site of the FIT IoTlab using 15 nodes (10×`nrf52dk` and 5×`nrf52840dk`). All nodes are within radio range of each other and are located in the same room in a 1 m × 1 m two dimensional grid. This room is located on the ground floor of a typical office building without dedicated shielding, making it subject to radio interferences in the 2.4 GHz band used by, *e.g.,* Bluetooth, WiFi, DECT. Figure 4(a) shows the spatial distribution of the nodes in the testbed, which we deploy in three different topologies (details see § 4.2).

Software configuration. The base of our implementation is RIOT version *2021.09*, commit `c739516`, and NimBLE version *1.4*, commit `b9c20ad`. We use the BLE default 1 MBit mode and, if not stated otherwise, the default parameters defined by RIOT and NimBLE.

For GNRC we enable *6LoWPAN* [29] as well as `gcoap` to support the *Constrained Application Protocol* (CoAP) [39]. We use the default GNRC packet buffer size of 6144 bytes and set the `gcoap` buffer size to 5120 bytes. Router solicitations are turned off as they are not needed. In GNRC, the link layer packet queue is configured to hold 4 IP packets. IP routes are configured statically by using the RIOT shell commands to create the analyzed network topologies. For *6BLEMesh* the BLE connections are also setup statically according to the configured IP routes. Dynamically created network topologies are out of scope of this work because there are no protocols for automated connection management in *6BLEMesh* networks available.

The NimBLE configuration of *IP-BLE-Adv* is based on the default configuration of *6BLEMesh* but with the major difference that extended advertisements are enabled. Both are configured to provide an MTU of 1280 bytes while the NimBLE buffer size is configured to 8.9 Kbytes for both setups. The link layer data length extension is enabled in the controller and the HCI interface is configured to transfer chunks up to 257 bytes, the maximum that NimBLE allows. The maximum number of chained auxiliary packets per advertising event is configured to 10. Furthermore, we allow a maximum number of 10 concurrent advertising instances.

4.2 Scenarios

We want to model network performance that resembles real-world deployments and also take the characteristics of low-power embedded hardware into account. For this reason, we focus on analyzing network metrics on the application layer by generating network load by sending CoAP packets [39].

In common IoT scenarios, numerous nodes periodically send their data to a gateway. To reflect this, we deploy multiple producers and a single consumer. Producers periodically send data based on non-confirmable CoAP PUT messages (*i.e.,* no application layer retransmission) to a consumer node. To prevent burst traffic, producers add a random jitter to their periodic producer interval. If not stated differently, each CoAP PUT message contains a payload of 100 bytes while the CoAP reply messages do not contain any payload.

Our experiments use a many-to-one scenario with a single consumer and 14 producer nodes. The scenario is deployed in three different topologies: 1-hop star, 3-hop tree, and 14-hop line. The star and tree topologies resemble setups that are likely to be encountered in real-life. The line topology is used to understand the network behavior under worst case conditions. All topologies are created using static IP routes such that the consumer is located in the center (star), root (tree), or edge (line). The specific mappings of nodes is illustrated in Figure 4.

5 Results

We compare network performance (*i.e.,* reliability and latency) and system performance (*i.e.,* power consumption and memory usage) for *IP-BLE-Adv* and *6BLEMesh* in different network setups.

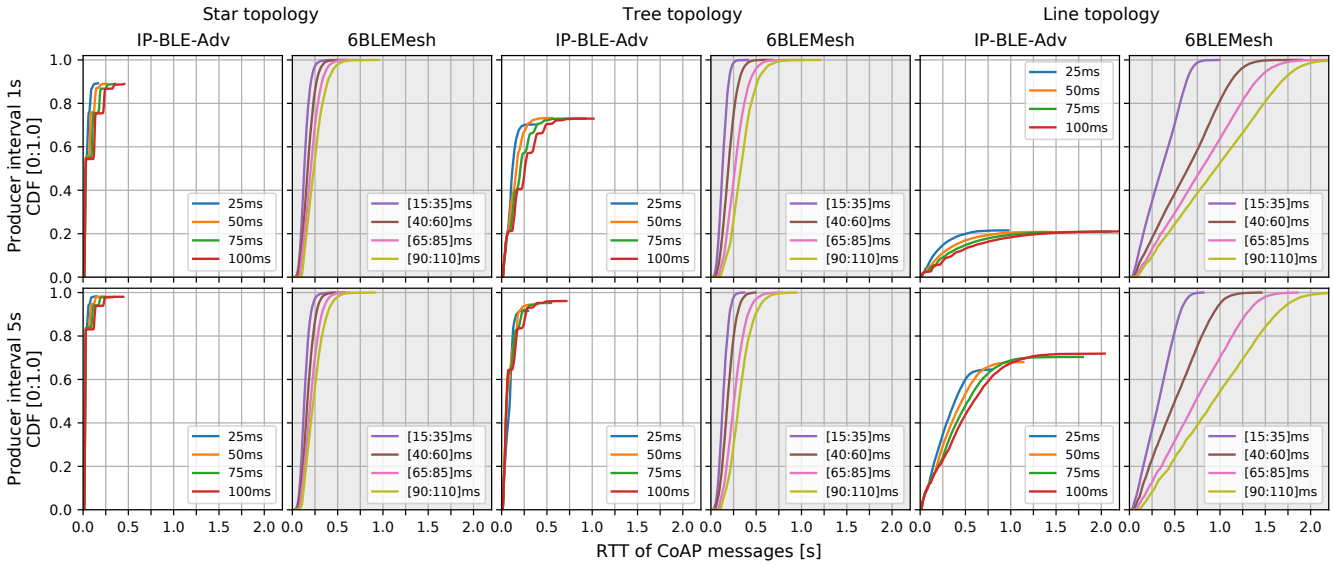


Figure 5: Distribution of CoAP round trip times for different producer intervals (1s, 5s) and varying advertising and connection intervals in three network topologies (star, tree, line).

5.1 Basic Performance Characteristics

Figure 5 exhibits the latency of CoAP messages when deploying *IP-BLE-Adv* and *6BLEMesh* in a star, tree, and line topology. We measure the latency as the time difference between sending a CoAP message and receiving the corresponding (empty) ACK, which is sent even in non-confirmable mode. Any packet loss leads to an infinite RTT. We consider high and low network load (*i.e.,* producer interval of 1s ±.5s and 5s ±2.5s). Each configuration runs for 1h.

In *IP-BLE-Adv* experiments, we configure four different advertising intervals (25ms, 50ms, 75ms, 100ms) and a static packet retransmission of 2 packets, which results in 3 advertising events for each IP packet (see § 5.2).

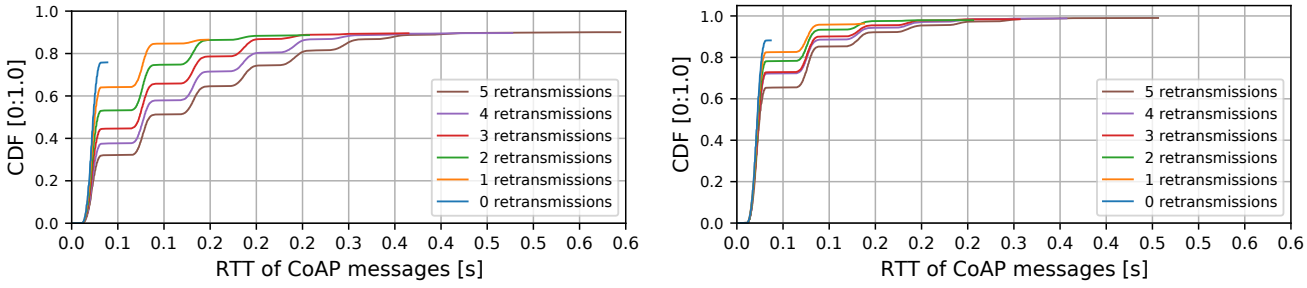
In *6BLEMesh* experiments, to account for high reliability, we use randomized connection intervals [35] such that these intervals match the advertising intervals used in *IP-BLE-Adv* (*i.e.,* [15:35]ms, [40:60]ms, [65:85]ms, [90:100]ms). To be independent of background noise, we conduct all *IP-BLE-Adv* experiments outside of office hours. We consider background noise in detail in § 5.3.

IP-BLE-Adv. The reliability of *IP-BLE-Adv* differs greatly depending on the network topology and load but, in general, significantly decreases when the network load increases. For example, using an advertising interval of 50ms and increasing the load from one packet every 5s to every 1s, reduces the CoAP packet delivery rates from 98.2% to 89.1%, 94.5% to 73.2%, and 67.9% to 20.8% in star, tree, and line topology, respectively. The reason for these losses is twofold. First, a higher network load increases the chance of packet collisions in the physical domain. Second, more packets increase radio scheduling conflicts, especially at the consumer node. In multi-hop tree and line topologies, these effects become multiplied because the overall number of packets sent is increased due to hop-wise packet forwarding along longer paths. For example, in our setup, the overall number of packets sent among all nodes is 6× higher in the line topology compared to the star topology. Furthermore, even if we configured multi-hop topologies based on IP routes, packets are sent in the same radio domain. Given that all nodes are in radio range of each other, packet collisions even via independent (IP) routes are observed.

The CoAP packet latency shows a stair effect. These steps are caused by the static retransmissions of advertising packets and the width of the steps is defined by the advertising interval. The delay transitions are most pronounced in the star topology. In the tree and line topology, processing and queuing delays along intermediate nodes smoothen transitions. In all configurations, we found that over 90% of the successful CoAP acknowledgments are received after 2.5× of the average hop count.

Comparison to *6BLEMesh*. The results of *6BLEMesh* (see Figure 5) are in line with prior work [35]. In all 24 experiments, no CoAP packet was lost. The network load caused by a producer interval of 1s does not lead to an overload in *6BLEMesh*, only the delay slightly increases in multi-hop topologies.

In terms of reliability, *6BLEMesh* is superior compared to *IP-BLE-Adv*. Under relaxed network conditions, though, *IP-BLE-Adv* achieves packet delivery rates that are acceptable for a wide range of IoT applications. In terms of latency, we observe a different picture: time-sliced channel hopping, which enables reliability in *6BLEMesh*, comes to the price of increased packet delay. In *IP-BLE-Adv*, packets are always sent immediately, *i.e.,* when they are handed to the Bluetooth Stack. Then, packet delays are only affected by (relatively small) processing and retransmission delays. Especially in scenarios with short paths (*e.g.,* star topology), *IP-BLE-Adv*



(a) Producer interval 1s.

(b) Producer interval 5s.

Figure 6: CoAP round trip time CDFs for different packet retransmission counts and producer intervals in a star topology network and an advertising interval of 50ms.

experiences significant shorter round-trip times compared to *6BLEMesh*.

5.2 *IP-BLE-Adv* Configuration Parameters

We identified two major configuration parameters that predominantly effect the network performance of *IP-BLE-Adv* networks, the **retransmission count** and the **advertising interval**.

Retransmissions. The unidirectional transmission of BLE advertisements challenges a reliable link layer because it prevents the implementation of acknowledgments to confirm successful messages. To increase the reliability of data transmissions, we apply a static retransmission scheme that defines how often an advertising event is replicated, similar to Bluetooth Mesh [17].

Figure 6 illustrates the impact of the retransmission, deployed in a star topology network and an advertising interval of 50ms. A higher a number of retransmissions increases the overall reliability, as expected, but few retransmissions have surprisingly notable impact. Focusing on the high load scenario (1s producer interval), the PDR significantly increases with a single retransmission (from 75.8% to 86.5%). Additional retransmissions have lower impact. A PDR of 90.0% requires 5 retransmissions. This behavior can be reproduced in different topologies and under different network loads.

The results in Figure 6 further illustrate a drawback of fixed retransmissions. The reliability of the first advertising event reduces with the number of retransmissions and thus packet delays increase. The reason for this is that nodes need to spend more time transmitting data while spending less time listening for incoming data, thus, the chance to miss incoming packets grows. In our configuration, this can be observed particularly on the consumer node (not shown), which experiences the most IP traffic. In our scenario, two retransmissions provide the best tradeoff between reliability and packet delay.

Advertising interval. The advertising interval defines the amount of time between two consecutive advertising events and thus the retransmission delay. It is worth noting that a Bluetooth controller does not apply the configured value directly when scheduling advertising events, because the Bluetooth standard defines to add a random jitter of 0 to 10ms between two advertising events. This jitter leads to less sharp transitions and a slight slope in the distribution of retransmissions (see Figure 6).

5.3 Noise Resilience

IP-BLE-Adv uses the three primary advertising channels to deliver data. These three channels can also be used by other BLE applications (*e.g.,* BLE beacons [25], Covid Warn Apps [4]) or interfere with external sources (*e.g.,* WiFi, DECT). This leads to the assumption that *IP-BLE-Adv* is less resilient to non-related radio activity compared to *6BLEMesh*, which benefits from 37 typically less crowded data channels.

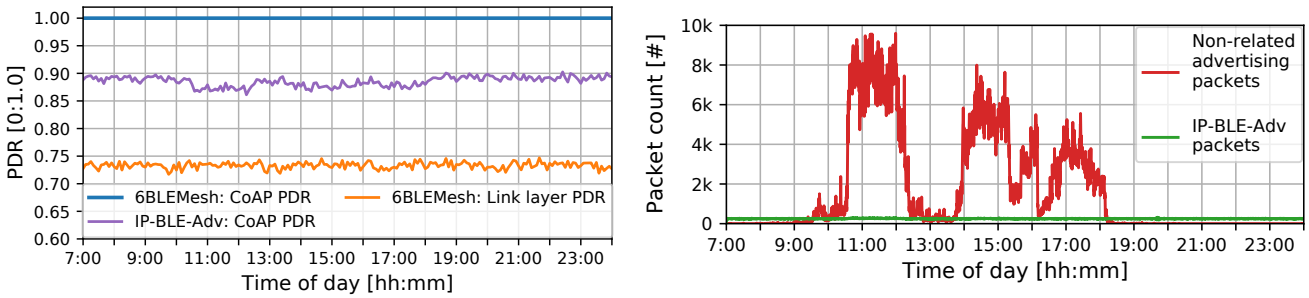
All used testbed nodes are located in an office building and are thus subject to diurnal radio background noise. To illustrate the impact of such noise on the network performance, we conducted 18h producer-consumer experiments in a star topology during working hours, 7AM – 12AM (UTC+2). The experiments were conducted with 2 retransmissions and an advertising interval of 50ms for *IP-BLE-Adv* and a connection interval of [40:60]ms for *6BLEMesh*.

Figure 7(a) compares the CoAP packet delivery rates for both network setups and shows additionally the link layer PDR of the *6BLEMesh* network. The *6BLEMesh* PDR is constant during the entire experiment and no impact of the external radio activities during office hours is visible. The time-sliced channel hopping makes *IP-BLE-Adv* robust against external interferences.

In *IP-BLE-Adv*, the CoAP PDR exhibits a different picture: during office hours successful CoAP packet delivery decreases. To shed light on the reason, Figure 7(b) shows the aggregated number of unrelated BLE advertising packets received by all *IP-BLE-Adv* nodes during the experiment. Even though these packets are only a subset of the external noise, a correlation between increased noise and decreased CoAP PDR in the *IP-BLE-Adv* network is visible. We conclude that *IP-BLE-Adv* is not critically sensitive to external noise (*i.e.,* 5% additional packet loss) but environments with high radio activity have a negative impact on packet delivery rates. This is not the case in *6BLEMesh*.

5.4 Energy Consumption

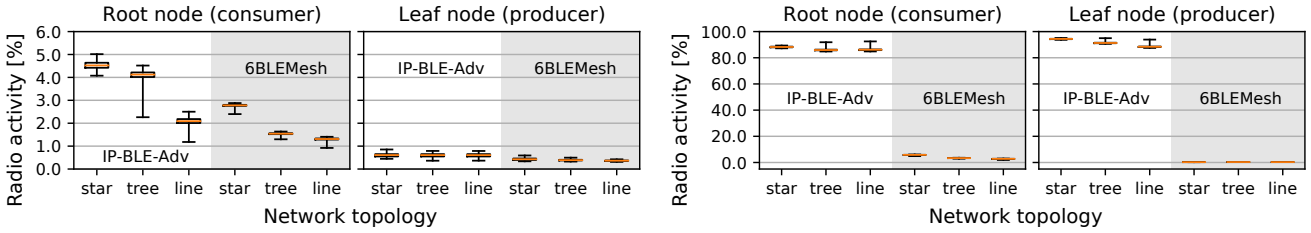
Energy consumption is a key metric in low-power IoT networks. As the FIT IoT-LAB does not provide any functions to measure the energy consumption directly, we derive this



(a) Packet delivery rates, binsize 300s.

(b) Total number of received advertising packets, binsize 10s.

Figure 7: Impact of radio interference and background traffic on the network performance of *IP-BLE-Adv* and *6BLEMesh* in star networks.



(a) Radio usage for TX.

(b) Radio usage for RX.

Figure 8: Comparison of radio utilization of *IP-BLE-Adv* and *6BLEMesh* for the root node (consumer) and one selected leaf node (producer) and different network topologies. Advertising interval for *IP-BLE-Adv* is 50ms, connection interval for *6BLEMesh* is [40:60]ms.

metric based on radio usage. Assuming that a node is only active when sending or receiving network data, such as in our experiments, the radio activity does provide a close estimate of the actual energy consumption. To measure the radio activity of each node, we count the time a radio is active in receive or transmit mode by inserting software counters directly in the low-level radio driver code.

Figure 8 shows the radio usage relative to the experiment runtime for the root of the network and a single selected leaf node. All experiments have been running for 1h with an advertising interval of 50ms and 2 retransmissions in *IP-BLE-Adv* and a connection interval of [40:60]ms in *6BLEMesh*.

IP-BLE-Adv nodes experience more transmit events compared to the *6BLEMesh* nodes (see Figure 8(a)). Although in each network topology the same number of CoAP packets traverse the network, the overall number of BLE packets is significantly larger in the *IP-BLE-Adv* networks because multiple packets are needed for each advertising event and the static packet retransmission. Differences become even more significant when measuring the amount of receiving time (see Figure 8(b)) since *IP-BLE-Adv* requires that nodes always listen. The aggregated radio activity of *IP-BLE-Adv* nodes is below 100%, though, due to radio switching and CPU overhead.

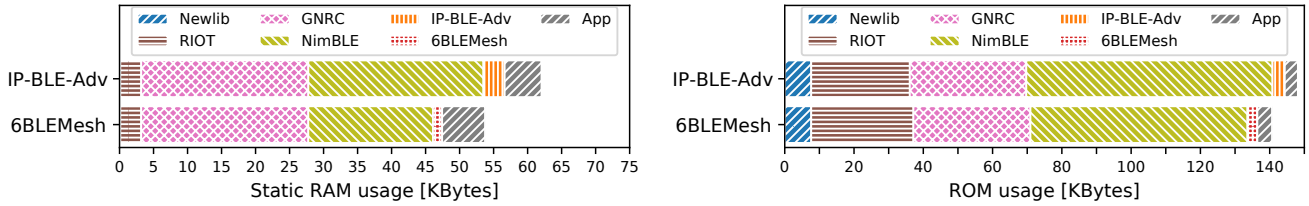
If we assume an average current consumption of 4.6mA while the radio is active, as given in the datasheet of our IoT hardware, the node lifetime would last at most 50h using a 230mAh coin cell battery. In practice, this value would be smaller due to TX activity and energy consumed by other CPU activity. In comparison, the same node in a *6BLEMesh* network with a radio activity of 0.5% has an average current consumption of 23µA and would last for 416 days on the same battery.

Without optimization our *IP-BLE-Adv* design does not allow for the same level of low-power deployments as possible with using *6BLEMesh*. However it is important to note, that we are comparing two extremes here. The energy consumption of *IP-BLE-Adv* is at its upper bound due to the permanently active radio. Duty cycling the radio and defining different roles could significantly improve the energy consumption (see § 7). The numbers for *6BLEMesh* in contrary present a lower bound as they depict a leaf node with only a single BLE connection. Router nodes will need to run neighbor discovery and connection management protocols involving additional BLE advertising and scanning which will increase power consumption in practice.

However, the energy consumption of non optimized *IP-BLE-Adv* is on par with other popular low-power WPAN technologies, such as unslottet IEEE802.15.4, and is a magnitude lower than the use of WiFi on embedded devices.

5.5 Memory Usage

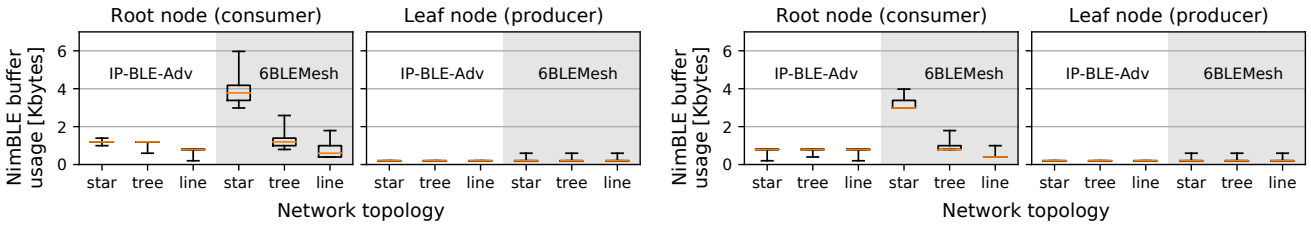
When analyzing memory requirements we consider two aspects to provide a complete picture: (*i*) the static RAM and ROM usage reserved and computed at linking time, and



(a) RAM requirements.

(b) ROM requirements.

Figure 9: Compile-time memory usages of the *IP-BLE-Adv* and *6BLEMesh* benchmark binaries, separated into different system components.



(a) Producer interval 1s.

(b) Producer interval 5s.

Figure 10: NimBLE packet buffer usages for a set of 1h experiments deployed in different network topologies, using an advertising interval of 50ms in *6BLEMesh* and 2 retransmissions and a connection interval of [40:60]ms in *IP-BLE-Adv*.

(*ii*) the actual amount of RAM that is used at runtime. Note that it is common to use only preallocated static memory in low-end embedded systems. This is also the case in RIOT and NimBLE.

Figure 9 compares the static memory usage between *IP-BLE-Adv* and *6BLEMesh*. Although the *IP-BLE-Adv* Bluetooth stack is less complex, the resulting firmware image is 7.5 Kbytes larger compared to the *6BLEMesh* build. The reason for this is that NimBLE currently cannot be configured to support extended advertisements without BLE connection handling functions. Hence, enabling extended advertisements adds additional RAM and ROM usage to *IP-BLE-Adv*, while *6BLEMesh* does not use that feature.

In *6BLEMesh*, the RAM usage depends on the preconfigured number of BLE connections a node can maintain simultaneously. For each BLE connection an additional RAM block of 1056 bytes needs to be allocated.

We want to emphasize that the measured code sizes can be decreased significantly when moving to production code. Our experiment applications contain a number of large software modules, such as the RIOT shell and our custom event logging, that can be dropped in user applications. Furthermore, to limit the impact of overflowing buffers, the NimBLE and GNRC packet buffers are configured to fill the unused RAM space, which can also be decreased in practice.

Figure 10 exhibits the memory used in the NimBLE packet buffer during runtime of different 1h experiments. We choose an advertising interval of 50ms with 2 retransmissions in the *IP-BLE-Adv* setup and a connection interval of [40:60]ms in the *6BLEMesh* setup. We show the buffer usage of the root node (consumer), which experiences the most network traffic, and a selected leaf node (one producer), experiencing the least traffic. Other leaf nodes show the same behavior.

The buffer usage of the root node in the *IP-BLE-Adv* network slightly increases with increasing network traffic. In contrast to this, in *6BLEMesh* networks, the root node uses significant more buffer space, not only when the number of open connections grow (star vs. tree vs. line), but also when network load increases (producer interval 5s vs. 1s). The buffer usages on the leaf node are comparable in both approaches but are slightly more volatile in *6BLEMesh*.

6 Discussion

Relevance of Bluetooth. The number of annually shipped BLE devices is expected to reach 6.4 billion in 2025, of which 770 million are smart home devices [20], with an estimated market size surpassing USD 16.7 billion by 2026 [24]. This will ensure a steady availability of both BLE-enabled end-user products and BLE-enabled micro controller platforms including software support. After its introduction, BLE was attested the potential to disrupt the IoT [36]. Improving IP over BLE by a lean connection-less mode enables additional low-power IoT network scenarios. For example, mobile nodes, which will benefit from reduced overhead because connection management is not needed, or massive deployments since nodes do not need to keep additional states for each neighbor, which is particularly important in the context of memory-constrained devices.

Bluetooth Mesh vs. *IP-BLE-Adv*. Similar to *IP-BLE-Adv*, Bluetooth Mesh is connection-less but, in contrast, limited in exploiting BLE channels and forwarding because of two restrictions. First, Bluetooth Mesh uses only the primary advertising channels, whereas *IP-BLE-Adv* leverages

both three primary and 37 secondary channels to reduce the load on the primary channels. Second, Bluetooth Mesh floods packets, whereas *IP-BLE-Adv* can leverage efficient IP-based routing. Hence, *IP-BLE-Adv* performance properties describe an upper bound for Bluetooth Mesh. Furthermore, *IP-BLE-Adv* allows for arbitrary IP packets instead of small packets that follow a specific data model, granting more freedom when implementing Internet services and applications.

802.15.4 vs. *IP-BLE-Adv*. Our results show that *IP-BLE-Adv* networks offer performance properties that are on par with common IEEE 802.15.4-based networks [21] and thus sufficient to efficiently run advanced IoT protocols and applications in multi-hop environments. Especially in scenarios of relaxed network loads and topologies with short paths, data transmission is reliable (<10% packet loss). From a systems perspective, the biggest advantage of *IP-BLE-Adv* compared to IEEE 802.15.4 is reduced complexity. The same radio hardware and software can be used to implement both connection-less and connection-based communication. Finally, *IP-BLE-Adv* provides a higher data rate (8×), which reduces transmission time and thus energy consumption.

***6BLEMesh* vs. *IP-BLE-Adv*.** *6BLEMesh* outperforms *IP-BLE-Adv* in terms of reliability (§ 5.1) and energy consumption (§ 5.4) but leads to higher packet latency. The time sliced channel hopping in *6BLEMesh* allows for extremely efficient radio usage and reliable data transfer. The management of allocating channels, however, adds radio and CPU overhead as well as latency. We argue that *IP-BLE-Adv* is a deployment option for IP over BLE applications that need to be optimized for low latency and effortless neighbor discovery while being able to tolerate certain overhead in terms of energy consumption and packet loss. These include, *e.g.,* device provisioning and high density assembly line scenarios (see § 1).

When configuring *IP-BLE-Adv* networks, we identified the advertising interval and the retransmission count as major parameters influencing the network performance (§ 5.2). The advertising interval is directly correlated to the latency of retransmitted packets while the retransmission count impacts reliability. For the latter we found that a retransmission count of 2 events provides the best trade-off between increase in network load and reliability gains, where values above 2 did only marginally increase the reliability. In the current state of the *IP-BLE-Adv* design, where nodes use a scanning duty cycle of 100% (radio in RX per default), the impacts of different parameters on the energy consumption are negligible.

Because both approaches depend on the identical base system of radio hardware and Bluetooth stack, they could be run simultaneously on the same node. As of now this is not supported by our implementation but can be added in the future. This would allow for more sophisticated use cases where e.g. a node could use *IP-BLE-Adv* to acquire provisioning information on how to join a local *6BLEMesh* network.

Implementation and debugging pitfalls. The BLE features utilized by *IP-BLE-Adv* are common to any BLE 5.0 stack (and newer versions). In contrast to the implementation of *6BLEMesh*, which uses L2CAP connection-oriented channels, we did not experience any stability issues with NimBLE. Reaching a stable configuration of NimBLE to handle large extended advertising data (*i.e.,* IP data) was, however, challenging. A number of parameters for the controller, HCI, and host levels needed to be synchronized to enable stable transfer of advertising data payloads (see § 4.1).

The Bluetooth standard does not require that the lengths of the individual advertising data (AD) segments and the fragmentation of these BLE payloads into link layer packets by the controller are aligned. This means that the AD headers do not have to be aligned with the payloads of the transmitted radio packets. This leads to a pitfall when debugging IP communication using external packet sniffers. The individually captured packets are then marked broken by tools such as Wireshark for not containing valid AD segments. To allow for seamless debugging we aligned the maximum size of AD segments with the controllers maximum link layer packet size.

Further system support. In this paper, we have focused on connected, constrained microcontrollers, which challenge system implementations but will become more popular in the future. Currently, mobile consumer devices play a major role when supporting BLE. Our proposal can also be implemented there. To the best of our knowledge, iOS and Android, the two dominant mobile platforms, support both all BLE features and APIs that are needed to run *IP-BLE-Adv* in user space. Porting *IP-BLE-Adv* allows for direct communication between mobile and embedded devices opening new use cases, *e.g.,* using mobiles as border routers.

7 Related Work

Transferring data over BLE advertisements in multi-hop networks has been standardized by the Bluetooth SIG in Bluetooth Mesh [17] and been extended by third-parties to support extended advertisements [33]. Bluetooth Mesh uses proprietary network and data protocols and does not support transfer of IP data. To the best of our knowledge, there is no prior work on transferring IP over BLE advertisements.

In earlier work [34], we showed that static packet retransmissions combined with flood-based routing applied by Bluetooth Mesh significantly multiplies the number of advertising packets sent. This behavior does not apply in *IP-BLE-Adv* networks due to IP-based routing instead of flooding. The performance of CoAP in common large-scale IEEE 802.15.4 networks was analyzed by Gündoğan *et al.* [21]. *IP-BLE-Adv* shows a similar network performance.

Nikodem *et al.* [31] analyzed the reliability of BLE advertisements in a star topology consisting of 200 nodes. They show that increasing advertising traffic does significantly reduce packet delivery rates. Our results are in line with this observation. More recent work by Zachariah *et al.* [43] shows, however, that data reception rates for BLE advertisements in such dense networks can be significantly improved by dynamically adjusting packet redundancy. Our work is compatbile with the proposed framework. Applying it to improve the reliability of *IP-BLE-Adv* networks will be part of our future work.

Potential optimizations of our initial design may rely on prior work. In our current *IP-BLE-Adv* proposal, all nodes use a BLE scanner duty cycle of 100% to keep the design and implementation lean and simple. To improve energy consumption, we could adapt parts that focused on optimizing BLE advertising and scan parameters given measures such as reliability or energy [12, 26, 37, 38, 41]. Further optimization schemes to reduce radio interference [11, 22, 42] and network load [32] in BLE advertising networks could also be applied to optimize *IP-BLE-Adv* networks.

8 Conclusion and Outlook

In this work, we presented *IP-BLE-Adv*, multi-hop IPv6 over BLE extended advertisements. In contrast to other common connection-less, low-power IoT technologies such as IEEE 802.15.4 and Bluetooth Mesh, *IP-BLE-Adv* features higher data rates and less energy consumption as well as higher throughput and the flexible implementation of Internet services and applications. Our experiments based on real IoT hardware deployed in a mid-sized testbed show that *IP-BLE-Adv* can complement IP over connection-based BLE (*6BLEMesh*). *IP-BLE-Adv* networks offer low latency and reliability that is sufficient in many multi-hop deployments as long as the network load is relaxed. *6BLEMesh*, on the other hand, is robust even in stressed networks and tailored to very low-power scenarios. Therefore, *IP-BLE-Adv* closes the gap towards a standard-compliant BLE stack that provides both connection-less and connection-based communication in Internet scenarios. We plan to analyze possible interferences due to concurrent operation and cross-dependencies in future research.

Acknowledgments. This work was supported in parts by the German Federal Ministry of Education and Research (BMBF) within the project *PIVOT*.

9 References

[1] Apache NimBLE. `https://github.com/apache/mynewt-nimble`.

[2] RIOT OS. `https://www.riot-os.org/`.

[3] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. FIT IoT-LAB: A large scale open experimental IoT testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464, Piscataway, NJ, USA, Dec 2015. IEEE Press.

[4] N. Ahmed, R. A. Michelin, W. Xue, S. Ruj, R. Malaney, S. S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, and S. K. Jha. A survey of covid-19 contact tracing apps. *IEEE Access*, 8:134577–134601, 2020.

[5] M. Aly, F. Khomh, Y. Guéhéneuc, H. Washizaki, and S. Yacout. Is Fragmentation a Threat to the Success of the Internet of Things? *IEEE Internet of Things Journal*, 14(8), Aug 2018.

[6] E. Baccelli, C. Gündogan, O. Hahm, P. Kietzmann, M. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wählisch. RIOT: an Open Source Operating System for Low-end Embedded Devices in the IoT. *IEEE Internet of Things Journal*, 5(6):4428–4440, December 2018.

[7] C. Bormann. 6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). RFC 7400, IETF, November 2014.

[8] C. Bormann, M. Ersue, and A. Keranen. Terminology for Constrained-Node Networks. RFC 7228, IETF, May 2014.

[9] S. M. Darroudi and C. Gomez. Experimental Evaluation of 6BLEMesh: IPv6-Based BLE Mesh Networks. *Sensors*, 20(16), 2020.

[10] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200, IETF, July 2017.

[11] R. G. Garroppo, L. Gazzarrini, S. Giordano, and L. Tavanti. Experimental assessment of the coexistence of Wi-Fi, ZigBee, and Bluetooth devices. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–9.

[12] K. Geissdoerfer and M. Zimmerling. Bootstrapping battery-free wireless networks: Efficient neighbor discovery and synchronization in the face of intermittency. In *NSDI*, pages 439–455, 2021.

[13] C. Gomez, S. Darroudi, T. Savolainen, and M. Spoerk. IPv6 Mesh over BLUETOOTH(R) Low Energy Using the Internet Protocol Support Profile (IPSP). RFC 9159, IETF, December 2021.

[14] C. Gomez, S. Darroudi, T. Savolainen, and M. Spörk. IPv6 Mesh over Bluetooth(R) Low Energy using IPSP. Internet-Draft – work in progress 10, IETF, April 2021.

[15] B. S. I. Group. Internet Protocol Support Profile. Bluetooth Specification 1.0.0, Bluetooth SIG, December 2014.

[16] B. S. I. Group. Bluetooth Core Specification. Bluetooth Specification 5.0, Bluetooth SIG, December 2016.

[17] B. S. I. Group. Bluetooth Mesh Profile. Mesh Profile 1.0.1, Bluetooth SIG, January 2019.

[18] B. S. I. Group. Bluetooth Market Update 2020. Technical report, Bluetooth SIG, 2020.

[19] B. S. I. Group. Bluetooth Core Specification Supplement. Core Specification Supplement 9, Bluetooth SIG, March 2021.

[20] B. S. I. Group. Bluetooth Market Update 2021. Technical report, Bluetooth SIG, 2021.

[21] C. Gündogan, P. Kietzmann, M. Lenders, H. Petersen, T. C. Schmidt, and M. Wählisch. NDN, CoAP, and MQTT: A Comparative Measurement Study in the IoT. In *Proc. of 5th ACM Conference on Information-Centric Networking (ICN)*, pages 159–171, New York, NY, USA, September 2018. ACM.

[22] D. Han, L. Xu, R. Cao, H. Gao, and Y. Lu. Anti-Collision Voting Based on Bluetooth Low Energy Improvement for the Ultra-Dense Edge. 9:73271–73285.

[23] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282, IETF, September 2011.

[24] IndustryARC. Bluetooth Low Energy Market Forecast (2021-2026). Technical report, 2021.

[25] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng. Ble beacons for internet of things applications: Survey, challenges, and opportunities. *IEEE Internet of Things Journal*, 5(2):811–828, 2018.

[26] C. Julien, C. Liu, A. L. Murphy, and G. P. Picco. Blend: Practical continuous neighbor discovery for bluetooth low energy. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 105–116, 2017.

[27] T. Lee, M. Lee, H. Kim, and S. Bahk. A Synergistic Architecture for RPL over BLE. In *Proc. of 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2016.

[28] M. Lenders, P. Kietzmann, O. Hahm, H. Petersen, C. Gündogan, E. Baccelli, K. Schleiser, T. C. Schmidt, and M. Wählisch. Connecting the World of Embedded Mobiles: The RIOT Approach to Ubiquitous Networking for the Internet of Things. Technical Report arXiv:1801.02833, Open Archive: arXiv.org, January 2018.

[29] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, IETF, September 2007.

[30] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668, IETF, October 2015.

[31] M. Nikodem and M. Bawiec. Experimental Evaluation of Advertisement-Based Bluetooth Low Energy Communication. 20(1):107.

[32] M. Nikodem, M. Slabicki, and M. Bawiec. Efficient Communication Scheme for Bluetooth Low Energy in Large Scale Applications. 20(21):6371.

[33] D. Pérez-Díaz-De-Cerio, A. Hernández-Solana, M. García-Lozano, A. V. Bardají, and J.-L. Valenzuela. Speeding Up Bluetooth Mesh. 9:93267–93284.

[34] H. Petersen, P. Kietzmann, C. Gündogan, T. C. Schmidt, and M. Wählisch. Bluetooth Mesh under the Microscope: How much ICN is Inside? In *Proc. of 6th ACM Conference on Information-Centric Networking (ICN)*, pages 134–140, New York, 2019. ACM.

[35] H. Petersen, T. C. Schmidt, and M. Wählisch. Mind the Gap: Multi-hop IPv6 over BLE in the IoT. In *Proc. of 17th International*

Conference on emerging Networking EXperiments and Technologies (CoNEXT)), pages 382–396, New York, December 2021. ACM.

[36] S. Raza, P. Misra, Z. He, and T. Voigt. Building the internet of things with bluetooth smart. *Ad Hoc Networks*, 57:19–31, 2017.

[37] J. Seo and K. Han. A survey of enhanced device discovery schemes in bluetooth low energy networks. *IETE Technical Review*, 38(3):365–374, 2021.

[38] G. Shan and B.-h. Roh. Performance model for advanced neighbor discovery process in bluetooth low energy 5.0-enabled internet of things networks. *IEEE Transactions on Industrial Electronics*, 67(12):10965–10974, 2020.

[39] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252, IETF, June 2014.

[40] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer. BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices. In *Proc. of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, New York, NY, USA, 2017. ACM.

[41] J. Yang, C. Poellabauer, P. Mitra, and C. Neubecker. Beyond beaconing: Emerging applications and challenges of ble. *Ad Hoc Networks*, 97:102015, 2020.

[42] T.-T. Yang and H.-W. Tseng. A service-aware channel partition and selection for advertising in bluetooth low energy networks. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, pages 2144–2150. Association for Computing Machinery.

[43] T. Zachariah, N. Jackson, B. Ghena, and P. Dutta. Reliable: Towards reliable communication via bluetooth low energy advertisement networks. In *International Conference on Embedded Wireless Systems and Networks*, 2022.

A Artifacts

All artifacts used to create the results in this paper are publicly available. These artifacts are composed of the actual implementation of *IP-BLE-Adv*, tooling for conducting experiments in the FIT IoTlab testbed and analyzing their results, as well as all the raw output of all experiments. Using these artifacts anyone should be able to reproduce our results, not only based on the provided raw data but also by re-running our experiments in an automated fashion.

Our *IP-BLE-Adv* experiments are based on the implementation described in § 3.2. To support full reproducibility, the exact configuration parameters used in the conducted experiments are listed in § 4.

A.1 Hosting

All artifacts produced in this work are available through the following sources:

`https://github.com/haukepetersen/RIOT/tree/ipbleadv`

Contains the source code to create a RIOT image that supports *IP-BLE-Adv*.

`https://github.com/haukepetersen/mynewt-nimble/tree/ipbleadv`

Contains the NimBLE branch used in our experiments.

`https://github.com/ilabrg/artifacts-ccr-ipbleadv`

Contains the detailed experiment descriptions and the tooling needed to run and analyze them.

`https://zenodo.org/record/8112830/files/ipbleadv_logs_raw.zip`

Contains the raw data gathered during our experiments. The data can be used to replicate the results presented in this work.

A.2 Software Platform

The *IP-BLE-Adv* module is located under `pkg/nimble/jelling` as part of the linked RIOT branch. This implementation includes a number of software hooks for collecting and printing trace data used for analyzing network packet flows.

The NimBLE branch also contains a number of custom print functions used to trace packets through the BLE stack.

A.3 Experimentation Framework

All experiments were controlled using a custom experimentation framework that takes care of allocating and controlling the nodes used in the testbed as well as collecting the experiment output. In this framework, each experiment is fully described in a dedicated YAML configuration file. Based on these configuration files it is possible to re-run any experiment.

The directory `tools/` contains the tooling used to analyze the experiment results and to create the figures presented in this paper.

The `README.md` contains more detailed descriptions of the framework and step-by-step instructions on its usage.