2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS) | 978-1-6654-3929-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/DCOSS52077.2021.00014

Leveraging Cross-Technology Broadcast Communication to build Gateway-Free Smart Homes

Hannah Brunner†, Rainer Hofmann†, Markus Schuß†, Jakob Link‡, Matthias Hollick‡,
Carlo Alberto Boano†, and Kay Römer†
†*Institute of Technical Informatics, Graz University of Technology, Austria*
‡*Secure Mobile Networking Lab, Technical University of Darmstadt, Germany*
{hannah.brunner, rainer.hofmann, markus.schuss, cboano, roemer}@tugraz.at – {jlink, mhollick}@seemoo.tu-darmstadt.de

***Abstract*—Despite the growing interest in cross-technology communication, its application to real-world systems is still limited, as existing schemes are mostly unidirectional and technology-specific. The lack of generic solutions as well as the complexity of their integration reduces the applicability in a broader scope. In this paper, we propose a solution to augment Wi-Fi, BLE, and ZigBee devices with the ability to transmit and receive cross-technology broadcast frames alongside their existing functionality. After experimentally evaluating the performance of our solution on a variety of hardware platforms, we leverage it to build a gateway-free smart home, where a smartphone can simultaneously control heterogeneous smart objects. The smart objects, which include an off-the-shelf ZigBee light bulb and a BLE-enabled door lock from different vendors, perform cross-technology communication while retaining their original functionality and can maintain duty-cycled operations.**

***Index Terms*—Cross-technology communication, IoT, Nexmon, Smart door lock, Smart light bulb, BLE, IEEE 802.15.4, Wi-Fi, ZigBee, X-Burst, Broadcom `bcm43`, Raspberry Pi, Nexus 6P.**

I. Introduction

In recent years, a considerable number of wireless communication technologies have emerged to satisfy the diverse requirements of various Internet of Things (IoT) applications. Although many of these wireless technologies operate on the same radio frequencies, their incompatible physical layers (PHYs) often prevent a direct data exchange between devices.

Because of this, the use of multi-radio *gateways* is necessary to allow data collection or dissemination across heterogeneous appliances and networks, e.g., in the context of smart homes as well as industrial and health-care IoT systems [1], [2]. Gateway-based communication, however, introduces additional costs as well as translation overhead, reduces scalability, and further increases the network traffic in already crowded unlicensed industrial, scientific, and medical (ISM) bands.

To address this issue, recent efforts have focused on the development of *cross-technology communication* (CTC) schemes giving heterogeneous wireless devices the ability to directly exchange information. Due to the popularity of the license-free 2.4 GHz ISM band, the vast majority of works on CTC have targeted communication between technologies operating in these frequencies (e.g., Bluetooth Low Energy (BLE), ZigBee[1], and Wi-Fi). CTC schemes are typically based either on energy sensing and the adoption of packet-level properties [3]–[6] or on PHY emulation techniques enabling communication at a high throughput [7]–[10]. Their use is promising not only to avoid the use of multi-radio gateways, but also to enable on-the-fly reconfiguration of sensors [11] as well as the development of coexistence mechanisms mitigating cross-technology interference and improving spectral efficiency [12].

Limited real-world use of CTC. Despite the growing interest in the topic and the large number of schemes being proposed, CTC remains so far confined to academia and its application to real-world IoT systems is still rather limited. For example, to date, there is still no study showing how to *concretely* leverage one of the key benefits of CTC – the ability to perform gateway-free communication across heterogeneous devices – to revolutionize a given IoT application domain. A few works have argued that directly sharing information without the assistance from multi-radio gateways would be a game changer in the development of smart homes [4], [9], [13], a sector plagued by severe interoperability issues leading to frustration among end-users [14], [15]. However, the concrete use of CTC in the smart home context or in other IoT applications that would benefit from gateway-free communication has not been explored yet, and the corresponding challenges hence remain unsolved. We identify two main reasons that led to the status quo.

Lack of generality and missing broadcast support. Existing CTC solutions, especially recent PHY emulation approaches, are often not generic, i.e., they exploit properties in the modulation process to enable a unidirectional data exchange between two specific technologies (e.g., Wi-Fi $\rightarrow$ ZigBee [9], BLE $\rightarrow$ ZigBee [7], or BLE $\rightarrow$ Wi-Fi [16]). This limits the applicability of CTC, as the use of point-to-point unidirectional communication significantly increases the complexity in coordinating multiple heterogeneous devices. The design of CTC solutions should rather be steered towards a universal scheme leveraging technology-independent primitives, so to transparently transmit data to any device. This would enable the transmission of cross-technology *broadcast* frames to several heterogeneous devices simultaneously, which not only reduces complexity and traffic, but also facilitates common network tasks such as neighbour discovery and service advertisement [17].

[1] Although we explicitly refer to ZigBee throughout this paper, we implicitly refer to the body of technologies built on top of the IEEE 802.15.4 PHY.

978-1-6654-3929-9/21/$31.00 ©2021 IEEE
DOI 10.1109/DCOSS52077.2021.00014

Complex integration in constrained IoT devices. Existing CTC schemes often require hardware changes [18], [19] or are implemented on platforms with plentiful resources, such as laptops [4], [12] and software-defined radios [19]–[21]. Little effort has been put on simplifying the integration of CTC functionality on off-the-shelf IoT devices, especially those with highly-constrained resources (i.e., with limited processing power, energy budget, and memory capacity). This hinders the applicability of CTC in real-world systems encompassing low-power wireless sensors and actuators, which are rather pervasive in many IoT application domains. Furthermore, CTC has often been treated as a standalone piece of functionality and was rarely integrated *alongside* the native communication stack of a device. This makes it difficult to upgrade existing (legacy) devices with CTC-related features without impairing or even replacing existing functionality. In prior work [5], we went the first steps towards the development of a generic framework, named X-Burst, supporting CTC among constrained IoT platforms. However, we only produced a preliminary proof-of-concept on ZigBee and BLE devices running Contiki [22] and did not fully explore how to integrate CTC beside the existing operations of each device.

Contributions. In this paper, we address the aforementioned problems and enrich off-the-shelf Wi-Fi, BLE, and ZigBee devices with the ability to transmit and receive cross-technology unicast and broadcast frames alongside their existing communication stacks. We achieve this by extending the X-Burst framework [5] with a scheduler orchestrating the transmission and reception of cross-technology frames in parallel to the operations of native communication stacks, as well as with support for Wi-Fi devices. The latter is challenging, given that common Wi-Fi devices do not expose support for sampling the received signal strength (RSS) – a necessary feature to decode cross-technology frames. An experimental evaluation on a variety of commercial Wi-Fi, BLE, and ZigBee devices shows that we can exchange cross-technology broadcast frames at data rates above 1 kbps; and further quantifies the impact of CTC activities on the existing communication stack of a device as well as on its energy efficiency.

We leverage our findings to build a *smart home solution* in which a smartphone can use its Wi-Fi interface to simultaneously control smart objects operating in the 2.4 GHz band *without the need of any multi-radio gateway*. The off-the-shelf smart objects, which include a ZigBee light bulb and a BLE-enabled door lock from different vendors, are *augmented* with the ability to interact with surrounding devices using CTC, i.e., they still retain their original functionality and can maintain duty-cycled operations. To the best of our knowledge, this work showcases the first concrete use of CTC that involves multiple heterogeneous devices performing gateway-free communication in a real-world IoT context.

The paper proceeds as follows:

- We first highlight how the use of gateways is often troublesome in the context of smart homes, and discuss how CTC can be leveraged to ease the problem (Sec. II).
- We enable support for Wi-Fi devices in X-Burst and enable the creation of a cross-technology broadcast primitive among the three most ubiquitous wireless technologies in the 2.4 GHz ISM band (Sec. III).
- We enrich X-Burst with a scheduler orchestrating CTC alongside the native communication stack of a device without affecting existing functionality (Sec. IV).
- We evaluate experimentally the performance of our cross-technology broadcast primitive, as well as the impact of CTC activities on the existing communication stacks and on the energy efficiency of a device (Sec. V).
- We develop a gateway-free smart home solution allowing a direct communication among smartphones and commercial devices employing Wi-Fi, BLE, or ZigBee (Sec. VI).
- After discussing our work's limitations and the open challenges (Sec. VII), we describe related work (Sec. VIII) and conclude with a summary of our contributions (Sec. IX).

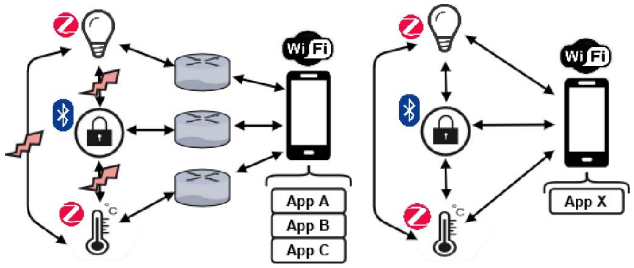
II. Case Study: Gateway-Free Smart Homes

The smart home market is one of the main drivers for the recent growth of the IoT [23]. In the past few years, an increasing number of vendors have developed a large variety of devices to extend the smart home ecosystem: examples are smart light bulbs, door locks, blinds, as well as sensors of various kinds (e.g., smoke, motion, and air quality).

Smart home appliances have largely different requirements and can significantly differ in terms of size, price, energy budget, memory, and computational capabilities, as well as employed wireless technology. They range from mains-powered devices with plentiful resources to constrained, battery-driven platforms with limited energy budget.

The latter typically cannot afford the use of Wi-Fi and instead employ low-power wireless technologies such as ZigBee or BLE. Although these technologies can all operate in the 2.4 GHz band, they are unable to directly exchange data with each other and with Wi-Fi due to PHY incompatibilities. Moreover, except for BLE, these low-power wireless technologies are rarely embedded in consumer electronic devices such as tablets and smartphones, which are often used to orchestrate the operations of a smart home. As a result, a smartphone that does not embed a ZigBee radio cannot directly control any ZigBee-based smart home device without a gateway.

Gateway nightmare and user frustration. To cope with this problem, smart home appliances often come with dedicated gateways: these enable interaction with nearby devices based on another technology and can act as a bridge to smartphones and tablets. Unfortunately, due to interoperability issues [24], it can happen that each vendor requires the use of a specific gateway, which may cause the deployment of several gateways within a single home. This introduces additional hardware, complex installation procedures, and ultimately results in costly, inefficient and non-scalable setups. Fig. 1(a) exemplifies the problem: several gateways need to be installed to let a smartphone interact with heterogeneous smart home devices from different vendors.



a) Conventional smart home setup b) Smart home setup with CTC

Fig. 1. The use of CTC in the context of smart homes allows to eliminate multi-radio gateways, thereby reducing costs and simplifying installation. With CTC, heterogeneous devices can interact directly and one can allow a smartphone to control multiple appliances at once with the same app.

Recently, smart home hubs such as the Apple HomeKit or Samsung SmartThings try to address this issue by providing support for a variety of appliances from different manufacturers with a single device [25]. However, they introduce a single point of failure and can only cover a fraction of the devices available on the market. Making sure that all appliances are compatible with each other is hence left to the end users: a quite heavy burden that can transform "the dream home into a nightmare" [26]. Furthermore, using several different devices introduces the "app problem", which refers to the obligatory installation of individual apps for each component [27]. This state of affairs often leaves customers frustrated and slows down the further adoption of smart home devices [28].

Easing the problem using CTC. The ideal scenario for end users is actually the one shown in Fig. 1(b), which depicts heterogeneous smart home devices that are *simultaneously* controlled from a smartphone using a *single* app, and that are able to *directly* interact with each other. Such a scenario does not only avoid the installation of gateways and maximize the user's flexibility when orchestrating all home devices, but also enables an autonomous control system architecture (i.e., smart home devices can independently exchange sensor data and adjust corresponding actuators without a central entity).

Such a *gateway-free* scenario can be achieved by means of CTC [4], [9], which allows a direct information exchange between devices making use of incompatible wireless technologies. However, as highlighted in Sect. I, state-of-the-art CTC solutions are still insufficient to make such a vision become reality. On the one hand, existing schemes lack generality and either focus on specific technologies/platforms, or are not fully bidirectional. Indeed, a technology-independent cross-technology broadcast primitive allowing a seamless interaction between both powerful (e.g., smartphones) and constrained (e.g., smart sensors and actuators) devices supporting BLE, ZigBee, or Wi-Fi, has not been developed yet. On the other hand, current CTC solutions do not focus on enabling an easy integration into existing resource-constrained devices. To be applicable in the smart home context, CTC should be added alongside existing operations, i.e., it should have a minimal footprint (so to still fit the remaining device memory) and should coexist with the native communication stacks. We show next how we fill these gaps and ultimately develop a prototypic solution for the scenario shown in Fig. 1(b).

III. Enabling a Cross-Technology Broadcast between Wi-Fi, BLE, and ZigBee Devices

In previous work [5], we went the first steps towards a generic framework supporting technology-independent CTC primitives on constrained IoT devices. We have tailored this framework, called X-Burst, to a few IoT platforms running the Contiki operating system and employing a ZigBee or BLE radio (i.e., the TI CC2650, Zolertia Firefly, and TelosB node).

To make X-Burst applicable to the scenario depicted in Fig. 1(b), we must also enable a bidirectional communication with off-the-shelf Wi-Fi devices and support more powerful appliances such as smartphones. This is a non-trivial task, as many Wi-Fi devices do not expose support for essential features such as frame injection or RSS sampling, and as operating systems for more powerful devices (e.g., Android, Linux) offer much less flexibility in exchanging data between user space and radio firmware compared to Contiki.

In this section, we tackle these problems and transform X-Burst into an OS-independent CTC framework supporting off-the-shelf Wi-Fi platforms and smartphones, ultimately enabling a cross-technology broadcast primitive among ZigBee, BLE, and Wi-Fi devices. We start our discussion by providing some background information on X-Burst's working principle.

A. The X-Burst Framework

X-Burst is a portable CTC framework that allows to convey information between heterogeneous devices by sending and receiving precisely-timed energy bursts [5]. These bursts can be transmitted by adjusting the length of legitimate packets, and received by observing the energy level on the RF channel, i.e., by performing a high-frequency RSS sampling. As most wireless devices have the ability to transmit payloads of arbitrary length and to perform energy detection (for clear channel assessment), X-Burst is highly generic and potentially allows *any* device to broadcast cross-technology frames. One just needs to agree on a common RF channel where to exchange cross-technology frames and on a shared *alphabet* to encode and decode information. The alphabet specifies how symbols are mapped to a predefined set of burst lengths, and depends on the characteristics of the communicating devices (e.g., the RSS sampling rate and the radio response time).

X-Burst is designed in a modular way to enable a high portability across multiple platforms and technologies. The original core modules described in [5] are highlighted in light grey in Fig. 2, whereas the modules in dark grey refer to the extensions we will describe in the remainder of this paper. A key role in making the framework generic plays the *hardware abstraction layer* (HAL), which allows the separation of CTC-related functionality from hardware-specific details. The main CTC logic (e.g., the encoding/decoding process, the (dis)assembly of cross-technology frames, and the mapping of symbols to bursts using a given alphabet) thus remains hardware-agnostic

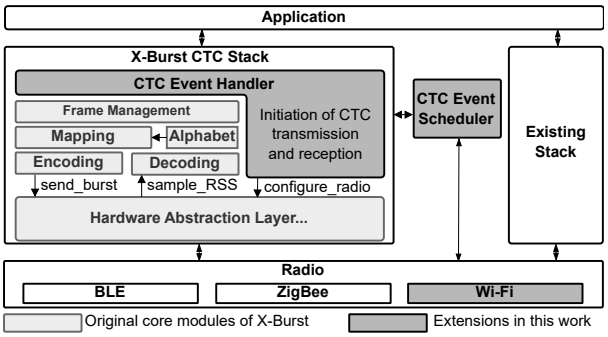


Fig. 2. Overview of our solution: X-Burst's original core modules are highlighted in light grey; our enhancements are highlighted in dark grey.

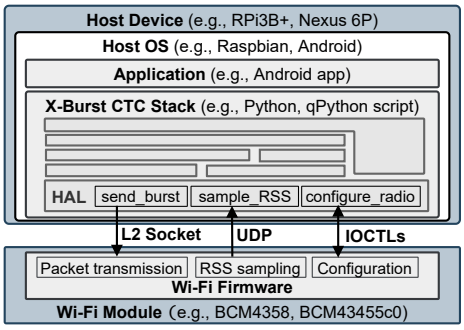


Fig. 3. Architecture used to support CTC on Wi-Fi platforms following the X-Burst core principles. The Wi-Fi radio firmware is extended to expose the necessary radio features to the CTC implementation running in user space.

and portable, while only radio-related primitives have to be implemented for each individual device. The HAL contains functions required to transmit energy bursts of a given length, to perform RSS sampling, and to configure the radio channel.

B. Enabling CTC on Off-the-Shelf Wi-Fi Platforms

In Contiki, an application has typically full access to the radio driver and all low-level features: thus, the implementation of X-Burst, including its HAL, is rather straightforward. On powerful Wi-Fi platforms (e.g., on a Raspberry Pi) and off-the-shelf smartphones, however, the radio cannot directly be controlled by applications, but is shielded by the operating system. Furthermore, several Wi-Fi modules run closed-source firmware that does not expose low-level functions (e.g., RSS sampling and monitor mode) by default. Examples of such closed-source modules are those produced by Broadcom/Cypress, which are embedded in popular platforms such as the Raspberry Pi 3/4, as well as several smartphones.

To overcome these limitations, we resort to the architecture shown in Fig. 3, which extends the firmware of the Wi-Fi radio and exposes the necessary features to the CTC implementation running in user space. The illustration is based on our implementation for the Raspberry Pi 3B+ and the Nexus 6P smartphone, both embedding a Broadcom/Cypress chip, but the general architecture is also applicable on other platforms (e.g., using Atheros or Intel Wi-Fi modules). As a Broadcom/Cypress chip runs closed-source firmware, we make use of the Ghidra reverse-engineering tool [29] and of the Nexmon C-based patching framework [30] to extend its functionality. Ghidra allows to explore the radio's capabilities that can then be made available to user-space applications by patching the corresponding addresses in the firmware. Nexmon allows to create such custom firmware patches and to enable monitor mode on a variety of Broadcom/Cypress chips.

We focus on the `bcm43` module, embedded in the Raspberry Pi 3B+ and Nexus 6P smartphone and use Nexmon to extend its firmware to provide X-Burst's HAL with access to the necessary radio features. To expose the *RSS sampling* capability, we implement a periodic timer triggering an existing energy detection function and tunnel the obtained RSS information to user-space applications through the kernel's network stack. Specifically, we transfer data to user-space by encapsulating it in *UDP frames*. When sent to the broadcast IPv4 address (255.255.255.255), the frames are automatically accepted by the kernel and passed on to user-space applications. This way, the HAL can retrieve the frames by listening to a datagram `socket`. Using this approach, we can achieve a RSS sampling rate of 10.3 and 5.9 kHz on the RPi and the Nexus 6P, respectively. These values are rather slow compared to BLE and ZigBee radios (e.g., the TI CC2650 BLE radio offers a RSS sampling rate of 43.5 kHz) and have to be accounted for in the alphabet computation. We further leverage `IOCTL` *system calls* to configure the Wi-Fi channel (i.e., to overlap with the operating channels of the other devices[2]) and to enable monitor mode. The latter is required to transmit Wi-Fi frames without prior connection to an access point. To *transmit energy bursts*, one can exploit the frame injection mechanism proposed in JamLab-NG [31]: by leveraging *L2 sockets*, the HAL can transfer raw Wi-Fi frames with a certain payload size (i.e., corresponding to a specific burst length) to the firmware, which are then transmitted using Nexmon's `sendframe()` function.

Exploiting all the aforementioned features, we are hence able to fully support CTC on a Raspberry Pi 3B+ and a Nexus 6P smartphone, providing them with the ability to seamlessly interact with both ZigBee and BLE devices.

Generality of the solution. The ability to access radio-related features from user-space allows to implement the CTC functionality in portable user-space scripts following X-Burst's original modular structure shown in Fig. 2. Our prototypic implementation is based on a *python* script, which is applicable to Linux-based operating systems like Raspbian or Android (using a *qPython* interpreter). As the interaction with the Wi-Fi firmware is solely carried out in the HAL, this script is easily portable to different platforms and not limited to Wi-Fi devices. Furthermore, our firmware patches can also be reused on other Broadcom/Cypress platforms supported by Nexmon, as they are written in portable C code.

[2]Wi-Fi radios use a much wider bandwidth (20 MHz), compared to BLE and ZigBee devices (2 MHz each). Despite these asymmetries, the energy bursts and their corresponding length can still be detected.

IV. Integrating CTC alongside Existing Communication Stacks

In the original X-Burst paper [5], we have shown how CTC could be integrated alongside Contiki's network stack, using ContikiMAC as a running example. Whilst this effort has shown the importance of integrating CTC next to the existing functionality of a device, it did not provide a generic solution. In commercial smart home applications such as the one shown in Fig. 1, indeed, the devices typically rely on full-grown ZigBee and BLE stacks, which are not available in Contiki. Hence, how to seamlessly integrate CTC alongside the functionality of a commercial smart home device (e.g., a smart light bulb such as the Ikea Trådfri) still needs to be investigated. In this section, we show how to tackle this problem by extending X-Burst with a CTC event handler and scheduler. We further describe how to implement these modules on BLE and ZigBee platforms available on the market.

A. CTC Event Handler and Scheduler

To coordinate radio access with existing communication stacks in a seamless way, we introduce a *CTC event scheduler* and a corresponding *CTC event handler* on top of X-Burst, as shown in Fig. 2. The *CTC event scheduler* is responsible for coordinating CTC activities alongside the native communication stack of a device and fires cross-technology transmission (CTC TX) and reception (CTC RX) events accordingly. The strategy with which CTC events are triggered is up to the developer: one can fire them periodically, or exploit the time in which the radio is idle. Such idle times can be inferred by the *CTC event scheduler* based on knowledge of the MAC protocol employed by the coexisting network stack [5], [17], or autonomously learnt by detecting periodicity in time series capturing the radio's idle time [32], [33]. The *CTC event handler* is in charge of configuring the radio for CTC operations, of initiating the cross-technology receptions or transmissions according to the scheduler's instructions, and of restoring the radio settings used by the coexisting communication stack.

Fig. 4 provides a deeper insight into the operations of the *CTC event handler and scheduler*. For simplicity, we assume that CTC RX events are periodic following a fixed interval $t_{Interval}$ [17] that is shared across all communicating devices. After each CTC RX event, the *CTC event handler* carries out RSS sampling for a maximum duration t_{Sense} to check for ongoing CTC transmissions. If a valid energy burst is detected within t_{Sense}, the device tries to determine the beginning of a CTC message by scanning for a preamble and eventually receives the cross-technology frame. Otherwise, the CTC operation is terminated immediately and the radio settings used by the coexisting communication stack are restored: this also keeps the radio on-time to a bare minimum. After a CTC TX event, which can be scheduled as soon as the radio is available (see Sec. IV-B), the *CTC event handler* triggers the consecutive transmission of a cross-technology frame until either an acknowledgement (ACK) is received, or a maximum time is reached. To ensure correctness of the CTC operation, one needs to ensure that:

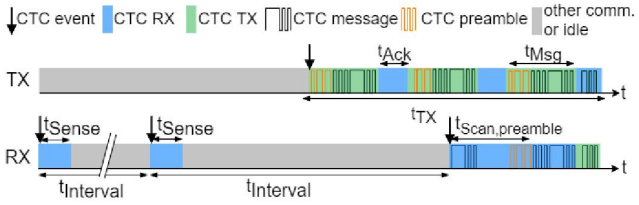


Fig. 4. Inner working of the CTC event handler and scheduler.

- $t_{Sense} > t_{Ack}$: the time a device scans for energy bursts must be greater than the longest ACK duration.
- $t_{Scan,preamble} > t_{Msg} + t_{Ack}$: after a device has detected an energy burst, it should scan for a preamble for a duration that is proportional to the longest message (t_{Msg}) and ACK (t_{Ack}), as the detected energy burst may refer to the beginning of a CTC transmission.

B. Implementation Remarks

While the *CTC event handler* is hardware-independent, the implementation of the *CTC event scheduler* depends on the software features of the employed communication stack. We exemplify our discussion by analyzing the integration of CTC next to the Silicon Labs EmberZnet ZigBee stack and the Nordic Semiconductor BLE stack, respectively.

The Nordic Semiconductor BLE stack offers primitives to request radio-access (for CTC RX and TX operations) during defined time intervals, while ensuring the proper execution of BLE-related tasks. This feature allows the *CTC event scheduler* to initiate CTC operations such that a reliable data exchange using BLE is guaranteed and implicitly results in a prioritization of BLE communication over CTC.

On the contrary, the Silicon Labs EmberZnet ZigBee stack does not provide a seamless scheduling of radio-related activity, as ZigBee communication is not bound to fixed time intervals. In case of non-preemptive access to the radio, the *CTC event scheduler* has hence to coordinate the execution of ZigBee and CTC tasks manually. The only information available to the scheduler is whether the radio is currently in use (i.e., if ZigBee communication is ongoing): whenever this is the case, pending CTC events should be omitted to avoid disrupting ZigBee activities. However, once a CTC event has been triggered, ZigBee communication is halted. These implementation differences, along with their implications, are analyzed more in detail and evaluated in Sec. V-B.

V. Evaluation

We perform next an experimental evaluation that quantitatively answers the following questions:

- What is the performance of our cross-technology broadcast primitive with respect to throughput, robustness, and communication range? (Sec. V-A)
- Can we transmit cross-technology broadcast frames without impairing existing communication stacks and while maintaining energy-efficient operations? (Sec. V-B)

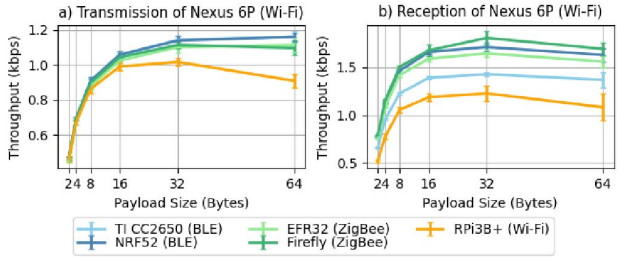


Fig. 5. CTC throughput measured by the Nexus 6P smartphone during both transmission (a) and reception (b) of cross-technology frames with different payload size to/from different IoT devices. In most of the cases, the achieved throughput is above 1 kbps.

A. Cross-Technology Broadcast Primitive

Building upon the concepts presented in Sec. III, we demonstrate and evaluate the performance of our cross-technology broadcast primitive across Wi-Fi, BLE, and ZigBee devices. In the following experiments, we focus on the CTC implementation only, i.e., other communication mechanisms are disabled.

Setup. We use a variety of off-the-shelf IoT platforms: the Nexus 6P smartphone and the Raspberry Pi (RPi) 3B+ (Wi-Fi), the Nordic Semiconductor nRF52840 DK and the TI CC2650 LaunchPad (BLE), as well as the Silicon Labs EFR32 Thunderboard Sense and the Zolertia Firefly (ZigBee). Unless stated otherwise, the measurements are performed in an office environment on (almost) interference-free channels (Wi-Fi ch. 9, ZigBee ch. 20, and BLE ch. 22). All devices are placed 1 m apart and share a common alphabet consisting of 2-bit encoding with four burst durations (224, 576, 928, 1280 μs).

Throughput. In a first experiment, we let the Nexus 6P smartphone broadcast 100 CTC messages back-to-back to all other devices and repeat the experiment 10 times. Fig. 5(a) shows the throughput of each device for different payload sizes. Higher payload sizes enhance the throughput thanks to a lower message overhead (e.g., preamble, header and checksum). At a certain point, however, the probability of decoding errors increases, leading to a lower throughput. The variance strongly depends on the RSS sampling capabilities of the receiving platform, and is particularly higher on the RPi, as it offers the slowest RSS sampling rate. The throughput in the other direction is shown in Fig. 5(b) and is more platform-dependent, as it is a function of the radio response time, i.e., of the minimum time required between the transmission of two energy bursts. Overall, Fig. 5 shows that CTC across all three technologies is possible with a data rate above 1 kbps, which is more than sufficient for the exchange of control messages or sensor data on constrained IoT devices. In principle, the throughput can be significantly improved when using a faster alphabet that is supported by all devices involved in the communication. Fig. 6, for example, compares the throughput achieved by the nRF52840 DK (BLE) and the Thunderboard Sense (ZigBee) when using the same alphabet used previously and a faster one consisting of 4-bit encoding with sixteen burst durations (224, 316, 408, ..., 1664 μs).

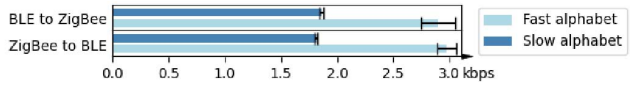


Fig. 6. CTC throughput of BLE and ZigBee devices with different alphabets.

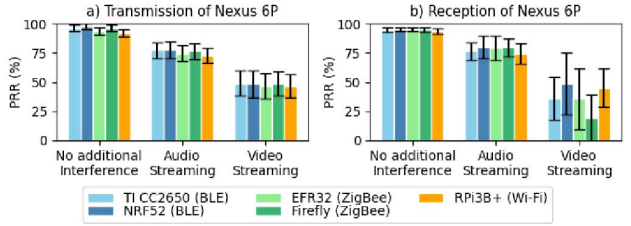


Fig. 7. Packet reception rate (PRR) under different interference scenarios.

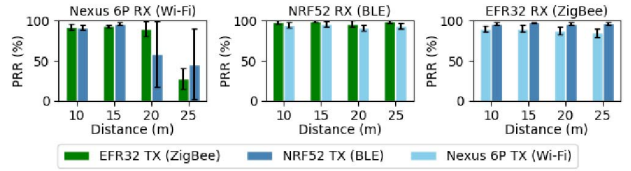


Fig. 8. PRR at increasing communication distance.

Robustness to RF noise. We repeat the same experiments with a fixed payload size of 16 byte and evaluate the packet reception rate (PRR) in the presence of RF noise. To this end, we record Wi-Fi interference patterns using a TP-Link USB Wi-Fi adapter and replay them repeatedly on the channel used for CTC using `tcpreplay`. The Wi-Fi interference is generated at a distance of 1 m from the receiving device with a transmission power of 13 dBm. Fig. 7 shows that, as expected, the PRR decreases as the amount of RF interference increases. While the PRR approaches 100% in absence of noise, it decreases to 72–80% in the presence of audio streaming traffic, and drops below 50% in the presence of video streaming traffic. As the cross-technology frames are broadcasted, the PRR drops in a consistent way for all receiving devices.

Communication range. We evaluate next the communication range by observing the PRR as a function of the distance between sender and receiver. Fig. 8 shows that BLE and ZigBee devices can receive cross-technology frames without a significant decrease of the PRR until 25 m. In contrast, on the Wi-Fi based smartphone, the PRR starts to visibly drop at a distance of 20 m. The reason lies in the wider bandwidth of Wi-Fi compared to BLE and ZigBee: as the smartphone measures the energy level on the entire 20 MHz channel, it is more difficult to detect the 2 MHz-wide energy bursts generated by BLE and ZigBee.

B. Integration into Existing Devices

Building upon the concepts presented in Sec. IV, we evaluate next the integration of CTC alongside the EmberZnet ZigBee stack running on a Thunderboard Sense and the Nordic Semiconductor BLE stack running on the nRF52840 DK.

We are interested in quantifying the impact of CTC operations on the native communication stack running on the device as well as the additional energy overhead introduced by CTC.

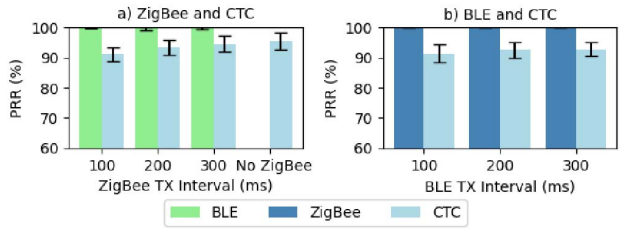


Fig. 9. Packet reception rate (PRR) measured while simultaneously using CTC in parallel to either ZigBee (a) or BLE (b).

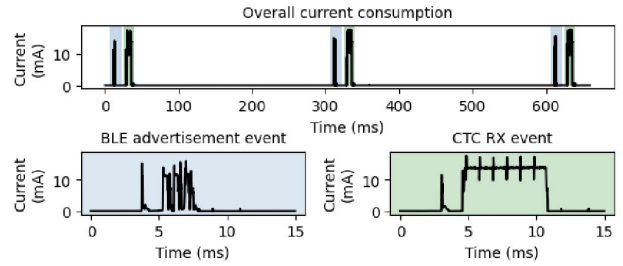


Fig. 10. Measured current consumption during BLE and CTC activities.

TABLE I
MEMORY FOOTPRINT OF THE ENHANCED IKEA TRÅDFRI LIGHT BULB

	Existing functionality	CTC stack	Free / Unused
ROM (kB)	240.69 (94%)	2.72 (1%)	12.58 (5%)
RAM (kB)	18.40 (57.5 %)	0.61 (2%)	12.99 (40.5%)

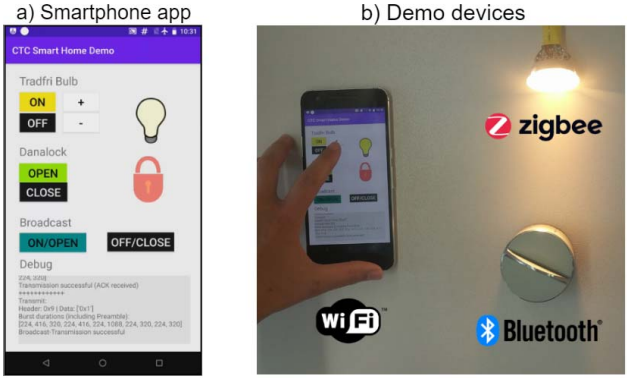


Fig. 11. Devices used in our gateway-free smart home prototype.

Impact on native communication stack. We concurrently transmit CTC and BLE frames to the nRF52840 DK (or ZigBee frames to the Thunderboard Sense) and observe the PRR experienced by both network stacks. Specifically, we transmit 100 CTC messages back-to-back, while BLE (or ZigBee) frames are transmitted periodically using different transmission intervals. Fig. 9 shows the results: the PRR of CTC activities is always above 90%, in accordance with the previous experiments. While the BLE communication is unaffected (PRR=100%), some of the ZigBee frames are lost, resulting in a PRR≈99%. Although minimal, this difference highlights the observations made in Sec. IV: depending on the primitives offered by the platform (e.g., the nRF52840 DK allows to request radio access so that BLE traffic is prioritized), one may or may not be able to guarantee that CTC has no impact on the native communication stack.

Energy overhead. We measure the current consumption of the nRF52840 DK using a Nordic Semiconductor Power Profiler Kit while the BLE and CTC stacks operate in idle mode. Fig. 10 shows the results when using a BLE connection interval and $t_{Interval}$ of 300 ms, as well as a t_{Sense} of 6 ms. Each idle CTC event consumes 257.8 μJ of additional energy. In comparison, a BLE advertisement event costs 54.9 μJ, i.e., in this configuration, the additional CTC activities increase the duty cycle from 0.87% to 2.97%. While the exact energy expenditure depends on the traffic load, hardware characteristics, and timing configurations (e.g., $t_{Interval}$), these results show that duty-cycled operations can be maintained, which is important for constrained devices with limited energy budget.

VI. BUILDING A GATEWAY-FREE SMART HOME

We leverage our findings to build a gateway-free smart home application, where a smartphone directly controls heterogeneous smart home devices available on the market using cross-technology broadcast communication, as shown in Fig. 1(b).

Equipping off-the-shelf smart objects with CTC. We make use of two off-the-shelf smart home devices to emphasize the real-world applicability of our solution: an Ikea Trådfri light bulb using a Silicon Labs EFR32 ZigBee SoC, and a Danalock V3, a BLE-enabled door lock employing a Nordic Semiconductor nRF52382 SoC. We replicate the existing functionality of both devices, including a full-grown ZigBee and BLE stack, respectively, and integrate our CTC solution next to it while ensuring a seamless coexistence with the native communication stack, as described in Sec. IV. This ensures that both devices can transmit and receive CTC frames while still performing the original operations, i.e., the two devices can be controlled with a smartphone as envisioned in Fig. 1(b), but also with their original remote controllers.

Minimal memory footprint. The integration of CTC on legacy devices is challenging due to the often limited storage capacity. For example, the Ikea Trådfri light bulb offers only 256 kB of ROM, out of which 94% is already occupied with existing functionality. Our solution requires less than 3 kB of ROM, thus still fitting into the highly constrained memory. Table I shows the memory footprint of our solution on the Ikea Trådfri: similar values apply to other platforms.

Demonstration video. Building on top of the cross-technology broadcast primitive presented in Sec. III, we let a Nexus 6P smartphone use its Wi-Fi interface to *directly* and *simultaneously* control the ZigBee-based Ikea Trådfri as well as the BLE-based Danalock V3 by means of CTC. We assign each device a fixed 1-byte address and define several commands to control the appliances accordingly. Our solution requires only a single smartphone app, shown in Fig. 11(a), and does not require any gateway. A demonstration video using the devices shown in Fig. 11(b) is available on YouTube[3].

[3]https://youtu.be/whD_H-UynJY

VII. Discussion and Future Work

Our demonstration setup can be extended to other smart home platforms operating in the 2.4 GHz band. This is possible as our CTC implementation is non-invasive (i.e., no hardware modifications are required) and thus, provided that low-level radio-features are accessible, existing smart home devices can support CTC with a simple firmware update. Note that the applicability of our solution is not limited to smart homes: it can be used, for example, in the context of industrial applications consisting of heterogeneous networks.

The smart home presented in Sec. VI relies on predefined, mutually available commands and device addresses. A common application layer that is well-understood across all CTC-enabled devices as well as a neighbour discovery service are required to build generic and scalable solutions. To this end, one can exploit recent efforts towards new connectivity standards increasing compatibility among smart home products [34] and towards the development of cross-technology neighbour discovery schemes [17]. We will investigate how to incorporate these efforts in our solution in future work.

Finally, a real-world application requires security mechanisms to provide authentication and encrypted communication. Although encryption algorithms are available on the targeted hardware platforms (e.g., as part of the ZigBee or BLE stack), secure cross-technology key management and data exchange is an open challenge. Along with efficient authentication schemes tailored to the CTC context, e.g., [35], this is also an interesting direction for future work.

VIII. Related Work

CTC has recently attracted a lot of attention in the research community, as a direct data exchange between devices with incompatible PHYs allows the development of many attractive services, including channel coordination [12], coordinated data aggregation [36], sensor reconfiguration [11], as well as clock synchronization across heterogeneous devices [37].

CTC is also promising to avoid the use of multi-radio gateways, which may be especially beneficial in smart home applications. Such a use case has been discussed theoretically [4], [9] and a few works have showcased a unidirectional communication from smartphones to individual light bulbs [7], [13]. In this work, instead, we let smartphones broadcast data to multiple commercial smart home appliances simultaneously, and allow a seamless control using a single app.

Existing work on CTC can be broadly classified into two categories: *packet-level modulation* and *PHY emulation*. In the latter, the payload of one technology is adjusted to embed a legitimate frame of another. This approach allows to transmit at high data rates from a high-end transmitter to a low-end receiver (e.g., Wi-Fi $\rightarrow$ ZigBee [8] or BLE $\rightarrow$ ZigBee [7]), but is asymmetric (i.e., it only works in one direction) and highly technology-specific. Recent studies have focused on enabling a data exchange also in the reverse direction: for example, [13], [20] allow the transmission of ACK frames. XBee [9] and LEGO-Fi [19] enable high-throughput communication from low-end to high-end devices using cross-decoding and demapping, but require hardware modifications.

Earlier CTC approaches are based on packet-level modulation (i.e., on the manipulation of packet-level properties such as transmission power [38], packet length [3], [6], or timing intervals [4], [21]) and are less technology-specific than PHY emulation. Moreover, they are also more suitable for bidirectional communication: for example, DCTC [21] and FreeBee [4] support ZigBee $\leftrightarrow$ Wi-Fi data exchange, whereas [5] showcased ZigBee $\leftrightarrow$ BLE communication. In contrast to PHY emulation, however, the achievable throughput is lower due to the limited granularity of packet-level properties and RSS sampling rates, with achievable data rates ranging from a few bps [4] to several kbps [5].

Although several CTC solutions exploiting packet-level modulation have been proposed, none of them allows a fully-bidirectional exchange across BLE, ZigBee *and* Wi-Fi devices. In this work, we create a platform-independent primitive enabling an all-to-all cross-technology broadcast communication on off-the-shelf BLE, ZigBee, and Wi-Fi devices. We have described a preliminary prototype of this primitive in a demo abstract [39]: in this paper, we fully describe its design and apply it in a real-world IoT scenario.

IX. Conclusions

In this paper, we have showcased a prototypic gateway-free smart home in which heterogeneous smart devices can directly interact with each other and be simultaneously controlled from a smartphone using a single app. To this end, we have first enriched the X-Burst framework with the ability to perform cross-technology broadcast transmissions across off-the-shelf BLE, ZigBee, and Wi-Fi devices. We have then designed and implemented concepts to extend existing smart home devices with full-grown ZigBee and BLE stacks with CTC functionality without affecting their normal operations. An experimental evaluation shows the performance of our solution, including its robustness, energy overhead, and achievable data rate.

Acknowledgments

This work was performed within the TU Graz LEAD project "Dependable Internet of Things in Adverse Environments". This work was also supported by the Austria Wirtschaftsservice Prototypenförderung (P1905510-WTG01) and by the SCOTT project. SCOTT (http://www.scott-project.eu) has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement no. 737422. This joint undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium, Norway. SCOTT is also funded by the Austrian Federal Ministry of Transport, Innovation and Technology under the program "ICT of the Future" (https://iktderzukunft.at/en/). This work was also partially supported by the LOEWE initiative (Hesse, Germany) within the emergenCITY centre and by the German Research Foundation (DFG) as part of the CRC 1053 (MAKI).

REFERENCES

[1] D. Yacchirema Vargas *et al.*, "Smart IoT Gateway For Heterogeneous Devices Interoperability," *IEEE Latin Am. Trans.*, vol. 14, no. 8, 2016.

[2] T. Zachariah *et al.*, "The Internet of Things Has a Gateway Problem," in *Proc. of the 16th HotMobile Worksh.*, 2015.

[3] K. Chebrolu *et al.*, "Esense: Energy Sensing-Based Cross-Technology Communication," *IEEE Trans. on Mobile Comp.*, vol. 12, no. 11, 2013.

[4] S. M. Kim and T. He, "FreeBee: Cross-Technology Communication via Free Side-Channel," in *Proc. of the 21st MobiCom Conf.*, 2015.

[5] R. Hofmann *et al.*, "X-Burst: Enabling Multi-Platform Cross-Technology Communication between Constrained IoT Devices," in *Proc. of the 16th SECON Conf.*, 2019.

[6] Y. Zhang *et al.*, "HoWiES: A Holistic Approach to ZigBee Assisted WiFi Energy Savings in Mobile Devices," in *Proc. of the InfoCom Conf.*, 2013.

[7] W. Jiang *et al.*, "BlueBee: a 10,000x Faster Cross-Technology Communication via PHY Emulation," in *Proc. of the 15th SenSys Conf.*, 2017.

[8] Z. Li and T. He, "WEBee: Physical-Layer Cross-Technology Communication via Emulation," in *Proc. of the 23rd MobiCom Conf.*, 2017.

[9] W. Jiang *et al.*, "Achieving Receiver-Side Cross-Technology Communication with Cross-Decoding," in *Proc. of the MobiCom Conf.*, 2018.

[10] Y. Chen *et al.*, "Reliable PHY Cross-Technology Communication with Emulation Error Correction," *IEEE/ACM Trans. Netw.*, vol. 28, 2020.

[11] I. Rüb *et al.*, "Ad Hoc 802.11-802.15.4 Crosstalk-Based Communication in Practice," in *Proc. of the 3rd MadCom Worksh.*, 2018.

[12] Z. Yin *et al.*, "Explicit Channel Coordination via Cross-technology Communication," in *Proc. of the 16th ACM MobiSys Conf.*, 2018.

[13] S. Wang *et al.*, "Networking Support For Physical-Layer Cross-Technology Communication," in *Proc. of the 26th ICNP Conf.*, 2018.

[14] Hestia Magazine, "Smart Home Compatibility: Towards an Open Eco System?" [Online] https://bit.ly/35rYXl3 – Last access: 2021-01-25.

[15] J. Kastrenakes, "It's Really Complicated to Connect the Home of the Future," 2018, [Online] https://bit.ly/2K3bion – Last access: 2021-01-25.

[16] Z. Chi *et al.*, "Concurrent Cross-Technology Communication Among Heterogeneous IoT Devices," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, 2019.

[17] R. Hofmann *et al.*, "SERVOUS: Cross-Technology Neighbour Discovery and Rendezvous for Low-Power Wireless Devices," in *Proc. of the 18th EWSN Conf.*, 2021.

[18] S. Wang *et al.*, "Symbol-level Cross-technology Communication via Payload Encoding," in *Proc. of the 38th ICDCS Conf.*, 2018.

[19] X. Guo *et al.*, "LEGO-Fi: Transmitter-Transparent CTC with Cross-Demapping," in *Proc. of the InfoCom Conf.*, 2019.

[20] H. He *et al.*, "Reliable Cross-Technology Communication With Physical-Layer Acknowledgement," *IEEE Trans. on Comm.*, vol. 68, no. 8, 2020.

[21] W. Jiang *et al.*, "Transparent Cross-Technology Communication over Data Traffic," in *Proc. of the 36th InfoCom Conf.*, 2017.

[22] A. Dunkels *et al.*, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proc. of the 29th LCN Conf.* IEEE Computer Society, 2004.

[23] K. L. Lueth, "IoT 2019 in Review: The 10 Most Relevant IoT Developments of the Year," 2020, [Online] https://bit.ly/3s6euAC – Last access: 2021-01-25.

[24] Plasmatic Techn., "Smart Home Interoperability: A Fragmented Landscape," 2019, [Online] https://bit.ly/3oablgd – Last access: 2021-01-25.

[25] L. Phan *et al.*, "Breaking Down the Compatibility Problem in Smart Homes: A Dynamically Updatable Gateway Platform," *Sensors*, 2020.

[26] M. Brown, "Apple, Google and Amazon's Plan could finally end the Smart Home Nightmare," 2019, [Online] https://bit.ly/3hYX0S2 – Last access: 2021-01-25.

[27] WIRED Brand Lab, "Why the Typical Smart Home Has Some Growing Up To Do," [Online] https://bit.ly/2XsFCf7 – Last access: 2021-01-25.

[28] Strata-Gee, "The Greatest Barrier to 'Smart Home' Adoption is Complexity," [Online] https://bit.ly/3oy81MK – Last access: 2021-01-25.

[29] "Ghidra. A software reverse engineering (SRE) suite," [Online] https://ghidra-sre.org/ – Last access: 2021-01-25.

[30] M. Schulz *et al.*, "Nexmon: Build Your Own Wi-Fi Testbeds with Low-Level MAC & PHY-Access using Firmware Patches on Off-the-Shelf Mobile Devices," in *Proc. of the 11th WinTECH Worksh.*, 2017.

[31] M. Schuß *et al.*, "JamLab-NG: Benchmarking Low-Power Wireless Protocols under Controllable and Repeatable Wi-Fi Interference," in *Proc. of the* 16^{th} *EWSN Conf.*, 2019.

[32] T. Puech *et al.*, "A Fully Automated Periodicity Detection in Time Series," in *Proc. of the 4th AALTD Workshop*, 2019.

[33] M. G. Elfeky *et al.*, "Using Convolution to Mine Obscure Periodic Patterns in One Pass," in *Proc. of the 9th EDBT Conf.*, 2004.

[34] "Project Connected Home over IP," 2018, [Online] https://www.connectedhomeip.com/ – Last access: 2021-01-25.

[35] S. Yu *et al.*, "Secure Authentication in Cross-Technology Communication for Heterogeneous IoT," in *Proc. of the DySPAN Symposium*, 2019.

[36] Y. Pan *et al.*, "CDD: Coordinating Data Dissemination in Heterogeneous IoT Networks," *IEEE Comm. Magazine*, vol. 58, no. 6, 2020.

[37] Z. Yu *et al.*, "Crocs: Cross-Technology Clock Synchronization for WiFi and ZigBee," in *Proc. of the 15th EWSN Conf.*, 2018.

[38] X. Guo *et al.*, "WiZig: Cross-Technology Energy Communication Over a Noisy Channel," *IEEE/ACM Trans. Netw.*, vol. 28, 2020.

[39] H. Brunner *et al.*, "Demo: Cross-Technology Broadcast Communication between Off-The-Shelf Wi-Fi, BLE, and IEEE 802.15.4 Devices," in *Proc. of the 17th EWSN Conf., demo session*, 2020.