Boosting the Bitrate of Cross-Technology Communication on Commodity IoT Devices

Wenchao Jiang, Zhimeng Yin, Ruofeng Liu, Zhijun Li, *Member, IEEE*, Song Min Kim, and Tian He, *Fellow, IEEE, ACM*

***Abstract*— The cross-technology communication (CTC) is a promising technique proposed recently to bridge heterogeneous wireless technologies in the ISM bands. Existing solutions use only the coarse-grained packet-level information for CTC modulation, suffering from a low throughput (e.g., 10 b/s). Our approach, called BlueBee, explores the dense PHY-layer information for CTC by emulating legitimate ZigBee frames with the Bluetooth radio. Uniquely, BlueBee achieves dual-standard compliance and transparency for its only modifying the payload of Bluetooth frames, requiring neither hardware nor firmware changes at either the Bluetooth sender or the ZigBee receiver. Our implementation on both USRP and commodity devices shows that BlueBee can achieve standard ZigBee bit rate of 250 kb/s at more than 99% accuracy, which is over 10 000x faster than the state-of-the-art packet-level CTC technologies.**

***Index Terms*— Cross-technology communication, signal emulation, bluetooth low energy, ZigBee, Internet of Things.**

I. Introduction

THE body of wireless devices has undergone an explosive increase in the last decade, which, under the emerging Internet of Things (IoT) era, is anticipated to grow as large as 20 billion by 2020 [7]. The dense deployment of wireless devices introduces highly-coexisting wireless environment which has long been perceived as a harsh habitat with severe interference. However, recent studies reveal that coexistence offers unique opportunities of collaboration – by taking advantages of specialized features among heterogeneous wireless technologies– that enable them to reach beyond independent operation. For example, in ZiFi [38] energy expenditure of power-hungry WiFi interfaces are significantly cut down with the assistance from low-power ZigBee radio, where it turns on the WiFi only when WiFi APs are found in the vicinity.

The traditional way of communicating among heterogeneous devices is to deploy multi-radio gateways, which suffers from several drawbacks including additional hardware cost, complicated network structure, and increased traffic overhead due to traffic flowing into and out from the gateway. To address these issues, the latest literature introduces the cross-technology communication (CTC) techniques which achieve direct communication among heterogeneous wireless devices with incompatible PHY layers. Existing solutions commonly use packet-level modulations, where the combinations of timing [18] and durations [5] of packets convey the data. Despite their effectiveness, the bit rates are inherently limited as they adopt coarse-grained 'packets' as the basis for modulation (analogous to 'pulse' in typical digital communication). For instance, the bit rate of BLE to ZigBee communication in the state of the art is limited to 17bps [18], four orders of magnitude slower compared with the 250kbps and 1Mbps data rate for legacy ZigBee and Bluetooth. The limited data rate greatly restricts real-time CTC applications, such as network coordination.

This paper introduces BlueBee, which boosts the bit rate of CTC via physical-layer emulation. More specifically, by smartly selecting the payload bits in a Bluetooth packet, BlueBee effectively encapsulates a ZigBee packet within the payload of a Bluetooth packet. The Bluetooth packet follows normal Bluetooth standard while the encapsulated ZigBee packet is compliant with the ZigBee standard and reaches the ZigBee bitrate cap of 250kbps at 99% accuracy. During the process, BlueBee does not require any hardware or firmware changes at either the Bluetooth transmitter or the ZigBee receiver, but only application-level payload embedding at the Bluetooth transmitter. In fact, the emulated ZigBee packet from Bluetooth is indistinguishable by the ZigBee receivers from normal ZigBee packets. These features make BlueBee ready to be deployed on billions of existing commodity IoT devices, smartphones, PCs, and peripherals.

Signal emulation is challenging, especially when the bandwidth of Bluetooth (1MHz) is only half of that of ZigBee (2MHz). The BlueBee design stems from two key technical insights: (i) the bridge between the (de)modulation techniques of Bluetooth and ZigBee and (ii) error tolerance of ZigBee demodulation (OQPSK/DSSS). Specifically, the phase differences between samples, referred to as *phase shifts*, is the bridge between the phase shift keying in ZigBee and the frequency shift keying in Bluetooth, which makes emulation possible. In addition, although the ZigBee signal cannot be perfectly emulated due to a narrower bandwidth of Bluetooth, BlueBee is optimally designed such that the inevitable error is minimized and kept under the tolerance of (i.e., the error is successfully corrected by) ZigBee's DSSS demodulator. As a result, BlueBee effortlessly runs on commodity Bluetooth devices by simply putting specific bit patterns in the Bluetooth packet payload. Also, BlueBee effectively utilizes the frequency hopping feature of Bluetooth to support concurrent

Manuscript received April 26, 2018; revised November 18, 2018; accepted March 27, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor T. Hou. Date of publication June 6, 2019; date of current version June 14, 2019. This work was supported in part by the NSF under Grant CNS-1525235, Grant CNS-1718456, Grant CNS-1717059, and NSF China under Grant 61672196. A conference paper [15] containing preliminary results of this paper appeared in ACM Sensys 2017. *(Corresponding authors: Tian He; Zhijun Li.)*

W. Jiang, Z. Yin, R. Liu, and T. He are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: jiang832@umn.edu; yinxx283@umn.edu; liux4189@umn.edu; tianhe@umn.edu).

Z. Li was with the University of Minnesota, Minneapolis, MN 55455 USA. He is now with the Harbin Institute of Technology, Harbin 150006, China (e-mail: lizhijun.hit@gmail.com).

S. M. Kim is with the School of Electrical Engineering, KAIST, Daejeon 34141, South Korea (e-mail: songmin@kaist.ac.kr).

Digital Object Identifier 10.1109/TNET.2019.2913980

communication across devices operating on different channels. Lastly, BlueBee offers reliable communication under dynamic wireless channel conditions. The contribution of this work is three-fold.

- We design BlueBee, the first CTC technique that emulates a legitimate ZigBee frame within the payload of a legitimate Bluetooth packet. No modification to the hardware or the firmware, for either the transmitter (Bluetooth) or the receiver (ZigBee), enabling full compatibility to billions of commodity devices.
- We address several unique challenges of signal emulation, including (i) optimized ZigBee phase shifts emulation using Bluetooth signal, (ii) the support for concurrent communication and low duty cycle operation under the frequency hopping of Bluetooth, and (iii) link layer reliability under dynamic channel conditions. These solutions offer general insights for signal emulation between other heterogeneous devices.
- We design and implement BlueBee on both the USRP platform and commodity devices. Our extensive experiments demonstrate that BlueBee establishes a high throughput and reliable communication under various environments and settings. Compared with a 17bps rate achieved by the state-of-the-art CTC from Bluetooth to ZigBee [18], BlueBee's reliable throughput of 250kbps, reaching that of the ZigBee standard, indicates performance gain of more than 10,000 times!

II. Motivation

With the rapid development of wireless technologies, such as WiFi, Bluetooth, and ZigBee, the ISM band suffers from the cross-technology interference (CTI) and channel inefficiency [12], [21], [36]. That is because the wireless technologies coexisting in the ISM band have heterogeneous PHY layer and cannot communicate directly with each other, thus not able to effectively coordinate channel use. The traditional approach to tackle the issue is to use a multi-channel gateway. But recently, researchers propose cross-technology communication (CTC) techniques as a promising alternative. However, both the traditional gateway approach and the existing CTC technologies have some intrinsic limitations.

- **Limitation of Gateway**. Multi-radio gateway is a usual and straightforward solution to bridge multi-technology communication [8], [16], [22], [24]. However, a gateway introduces not only additional hardware cost but also the labor intensive deployment cost, which would be prohibitive for the mobile and ad hoc environment. In addition, a dual-radio gateway increases the traffic overhead by doubling traffic volume in the ISM band, which further intensifies the cross-technology interference.
- **Limitation of Packet-Level CTC**. The recent cross-technology communication aims at building explicit channel coordination with the direct communication among heterogeneous wireless technologies. For examples, heterogeneous devices can allocate the channel in a way similar to the RTS/CTS in the 802.11 protocol [1], thus leading to a better channel efficiency. Unfortunately, to the best of our knowledge, existing CTC designs [5], [18], [37] rely on sparse packet level information such as the beacon timing [18] and multi-packet sequence patterns [32], introducing a delay of at least hundreds of milliseconds. Such a delay is way too for effective channel coordination in real-time.

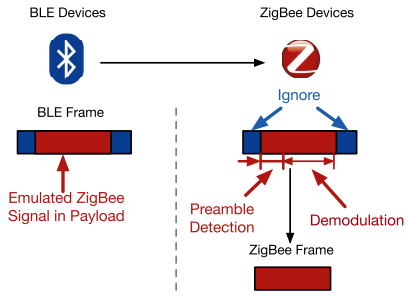


Fig. 1. The system architecture of the BlueBee.

In contrast to the limitations of gateway approach and existing CTC approaches, BlueBee is able to transmit a ZigBee packet directly from a Bluetooth device as fast as a normal ZigBee device, which is an enabler for real-time applications, such as the channel coordination between heterogeneous commodity devices in the dynamic environment. In the paper, although our description will be based on one specific Bluetooth standard, Bluetooth Low Energy (BLE), to ZigBee, the idea can be generalized to other Bluetooth standards, such as Bluetooth Classic (discussed in Section VII-A), as well as other phase shift keying and frequency shift keying wireless technologies.

III. BlueBee in a Nutshell

A. Overview

BlueBee is a high data-rate CTC technique from BLE to ZigBee compatible with both ZigBee and BLE standards. The basic idea of BlueBee is illustrated in Fig. 1 – BlueBee encapsulates a legitimate ZigBee frame within the payload of a legitimate BLE frame, by carefully choosing the payload bytes. At the PHY layer, the selected payload resembles (i.e., emulates) the signal of a legitimate ZigBee frame. When the BlueBee packet reaches a ZigBee receiver, the emulated ZigBee frame in the payload part is detected (via the compatible ZigBee preamble) and demodulated, just like any other ZigBee frame originated from a ZigBee sender. We note that the header and trailer of the BLE packet are incompatible with ZigBee and are naturally disregarded, or equivalently, treated as noise. In fact, such a design makes BlueBee dual-standard compliant. At the sender, a BlueBee packet is no more than a normal BLE packet with a carefully chosen payload. At the receiver side, the ZigBee device cannot tell whether the frame is from a ZigBee device or is emulated by a BLE device, due to the indistinguishable PHY layer waveform.

B. Unique Features

In Table I, we illustrate the technical advantages of BlueBee, as the first PHY-layer CTC, compared to the gateway approach and the state-of-the-art packet-level CTC approaches: 1) Compared with the gateway approach, BlueBee provides direct communication between heterogeneous devices, which saves the gateway facility and deployment cost and incurs less interference to the wireless network, i.e., a gateway will double the traffic in the air by the traffic going into and outside the gateway; 2) Compared with existing CTC solutions [5], [18], [32], BlueBee boosts the CTC data rate, making it comparable with normal wireless technology, i.e., ZigBee, so that real-time applications, such as channel coordination through CTC, is feasible; 3) BlueBee enables multi-channel concurrent CTC by the inherent frequency hopping in the BLE communication to best serve ZigBee devices on a wider ISM band.

TABLE I
COMPARISON OF BLUEBEE AND EXISTING CTC SOLUTIONS

	Cost	Spectrum Efficiency	Bit Rate	Multi-channel CTC
Gateway	Medium	Medium	High	Not Support
ESense [5]	Low	Low	Low	Not Support
FreeBee [19]	Low	Medium	Low	Not Support
B^2W^2 [6]	Low	Medium	Low	Support
BlueBee	**Low**	**High**	**High**	**Support**

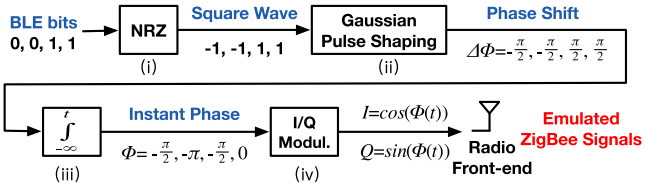


Fig. 2. BLE as the transmitter with GFSK modulation.

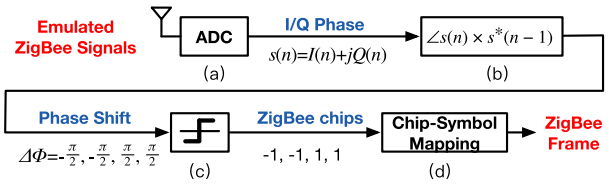


Fig. 3. ZigBee as the receiver with OQPSK demodulation.

IV. BLUEBEE DESIGN

This section briefly introduces some backgrounds about ZigBee and BLE followed by the BlueBee design in detail.

A. Background

We first study how a BLE transmitter and a ZigBee receiver work in relation to BlueBee.

B. BLE Transmitter

BLE uses Gaussian Frequency Shift Keying (GFSK) modulation, where the frequency shift will introduce phase changes over time .[1] Fig. 2 illustrates the entire procedure from payload bits to corresponding BLE waveform from steps (i) to (iv). In (i) BLE bits first go through a non-return-to-zero (NRZ) module that modulates a series of BLE bits to a square wave of either -1 or 1, where each square pulse lasts $1\mu s$ long. (ii) This wave passes through the Gaussian low pass filter, which shapes the wave into a band-limited signal. The bit change between -1 and 1 indicates negative and positive frequency shift, which further results in the phase change or phase shift of $\pm\pi/2$ during one-bit time. (iii) By integrating the phase shifts over time, we yield the instant phase with respect to time. (iv) The In-phase and Quadrature (I/Q) signal are calculated through the cosine and sine of the instant phase, respectively, which are multiplied to the carrier and pushed into the air through the BLE RF front-end.

The goal of BlueBee is to construct time-domain waveforms that can be demodulated by a commodity ZigBee receiver. In other words, emulate ZigBee signal at BLE. To do so, we imagine ZigBee signal containing the data of our choice is emitted from the BLE RF front-end, and reverse engineer steps (iv) to (i) accordingly. In step (iv), ZigBee signal in the air is sampled at BLE sampling rate (1Msps). From the sampled I/Q signals, the corresponding instant phases are obtained. Reversing step (iii) yields the phase shifts between consecutive BLE samples, where the corresponding series of square waves are found by reversing step (ii). Finally, these waves are mapped to data bits at the BLE packet payload which can be freely set, indicating that the targeted ZigBee signal is emulated simply by setting the BLE packet payload with the correct bits.

Such an approach enables the emulated waveform to be seamlessly demodulated by commodity ZigBee radios as legitimate ZigBee packets, without any change incurred to BLE's GFSK modulator. However, such emulation is not trivial due to various constraints, such as the narrower bandwidth of BLE (1MHz) compared to ZigBee (2MHz), which will be discussed in the later part of the section.

[1]Note that frequency is the derivative of phase. A frequency shift keying $s(t) = Acos(2\pi(f \pm \Delta f)t)$ is equivalent to a phase shift keying of $s(t) = Acos(2\pi ft \pm \Phi(t))$, where $\Phi(t) = 2\pi\Delta ft$.

TABLE II
SYMBOL-TO-CHIP MAPPING IN ZIGBEE (802.15.4)

Symbol (4 bits)	Chip Sequence (32 bits)
0 0 0 0	11011001110000110101001000101110
0 0 0 1	11101101100111000011010100100010
...	...
1 1 1 1	11001001011000000111011110111000

C. ZigBee Receiver

As Fig. 3 depicts, BlueBee enables BLE to transmit emulated ZigBee packets which can be demodulated by any commodity ZigBee device through the standard Offset Quadrature Phase Shift Keying (OQPSK) demodulation procedure. This is initiated by step (a), where ZigBee captures the BLE signal on the overlapping 2.4GHz ISM through the analog-to-digital converter (ADC), to obtain I/Q samples. A pair of I/Q samples are often referred to as a complex sample $s(n) = I(n)+jQ(n)$. In step (b), the phase shifts between consecutive complex samples are computed from $arctan(s(n)\times s^*(n-1))$, where $s^*(n-1)$ is the conjugate of $s(n-1)$. In step (c) positive and negative phase shifts are quantized to be 1 and -1, corresponding to ZigBee chips 1 and 0.

Finally, in (d), 32 ZigBee chips are mapped to a ZigBee symbol, by looking up a symbol-to-chip mapping table (Table II) predefined in DSSS. There are 16 different symbols

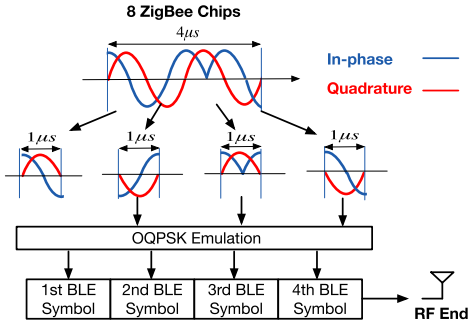


Fig. 4. Emulating ZigBee with BLE.

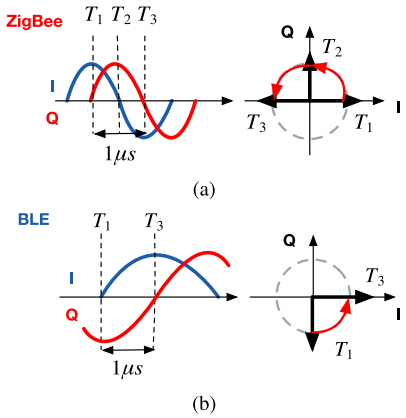


Fig. 5. (a) ZigBee signal indicating two chips, '11', as phase shifts from T_1 to T_2, and from T_2 to T_3 are both positive ($\pi/2$). (b) is the emulated signal of (a), by BLE (which is in fact BLE symbol '1'). When fed into ZigBee receiver this signal is sampled at T_1, T_2, and T_3 to give two consecutive positive ($\pi/4$) phase shifts. This yields ZigBee chips of '11', indicating successful emulation.

where each represents $log_2 16 = 4$ bits. We note that in the face of noise/interference the phases may suffer from errors ($+ \leftrightarrow -$), which induce reversed chips ($1 \leftrightarrow 0$). In such a case, the closest symbol with the smallest Hamming distance is selected.

D. Opportunities and Challenges of Emulation

Conceptually, emulation of ZigBee signal via BLE is possible due to two key technical insights. First, the phase shift is the bridge between ZigBee's OQPSK and BLE's GFSK (de)modulation. Phase shifts are directly interpreted as bits in BLE, while at the same time related to ZigBee chips. So in BlueBee, we carefully choose BLE bits so that their output phase shifts mimic consecutive ZigBee chips to achieve signal emulation, as shown in Sec IV-E.

Second, ZigBee only considers *sign* ($+$ or $-$) of the phase instead of the absolute phase value, which offers great flexibility in emulation. It is really important in the emulation because the bandwidth of BLE (1MHz) is only half of that of ZigBee (2MHz), so that phase shifts in BLE are not sufficient to express all ZigBee chips, leading to inevitable errors in emulation. More specifically, the DSSS modulation in ZigBee maps 32bit chip sequences to 4bit symbols, so that a ZigBee symbol can be correctly decoded if the Hamming distance between the received and ideal chip sequence is within a threshold of 12 (may be adjusted up to 20 [21]). We will see in Sec IV-F this margin is enough for ZigBee to correctly demodulate the emulated signal.

E. OQPSK Emulation

Here we demonstrate emulating ZigBee's OQPSK modulation with BLE, which is a nontrivial problem due to the narrower bandwidth of BLE compared to ZigBee (1MHz vs 2MHz). Fig. 4 illustrates the emulation process with an example of 8 ZigBee chips, where it starts by cutting the sequence into two-chips pieces (one In-phase chip and one Quadrature chip) with durations of $1\mu s$. Since one BLE symbol also lasts $1\mu s$, each of the pieces is then emulated by a BLE symbol.

Let us now look into how the emulation is performed on a two-chip piece. Recall that OQPSK (i.e., ZigBee) observes phase shifts between consecutive samples, whose signs are translated to chips of -1 and 1 (steps (a) and (b) in Fig. 3). The left in Fig. 5 depicts ZigBee signal (not emulated) containing two chips of '11', where $T_1 - T_3$ are the timings of three consecutive samples every $0.5\mu s$, the ZigBee sampling rate. On the right, the constellation of the samples at the corresponding timings are plotted with arrows. The phase shift between the arrows of T_1 and T_2 is $\pi/2$. Since a positive value, this is translated to a chip of '1'. The next chip is computed similarly between samples T_2 and T_3, which also yields a chip of '1'.

Now we show that the above mentioned ZigBee signal can be successfully emulated by BLE, which is demonstrated in the left in Fig. 5b. Although the signal appears to be distinct from ZigBee signal (left in Fig. 5a), it still delivers the same chips of '11' to ZigBee receiver. The key point here is that only the *sign* of phase shift is considered (not the amount). To understand this, we first notice that the left in Fig. 5b reflects the bandwidth of BLE being only half of ZigBee – i.e., the sinusoidal curves indicating I/Q signals have half the frequency, or equivalently, double the period. When this signal is fed into the ZigBee receiver and sampled at $T_1 - T_3$, the resulting constellation is as the right in Fig. 5b. From the plot, the phase shift between T_1 and T_2 is $\pi/4$ (i.e., positive), which yields a chip of '1'. The same applies to the phase shift between T_2 and T_3. This indicates that the BLE signal in the left in Fig. 5b) indeed yields the same chip sequence of '11' at the ZigBee receiver, as the ZigBee signal in the left in Fig. 5a. In other words, the ZigBee signal is successfully *emulated* by the BLE.

In fact, from the BLE's perspective, the signal at the left of Fig. 5b is simply a BLE signal representing a phase shift of $\pi/2$. This is because the sampling rate of BLE is half of the ZigBee, due to the bandwidth difference and the corresponding Nyquist sampling rate. Specifically, BLE samples T_1 and T_3 whose phase difference is $\pi/2$. Conversely speaking, by letting BLE transmit bits corresponding to phase shift of $\pi/2$, the BLE devices are able to deliver chip sequences of '11' to a ZigBee receiver. This is the key enabler to BlueBee, where ZigBee packet is encapsulated within a BLE packet simply through payload bit patterns.

From the example in Fig. 5b, we have found that a single phase shift in BLE is interpreted as two phase shifts in ZigBee, as per bandwidth difference. That is, BLE has lower degree of freedom, where it can change phase shifts ($- \leftrightarrow +$) every $1\mu s$ whereas it is $0.5\mu s$ for ZigBee. Due to this, while ZigBee chip sequences are of '11' or '00' ('consistent phase' hereafter, since phase shifts are kept consistent at $+$ or $-$) can be

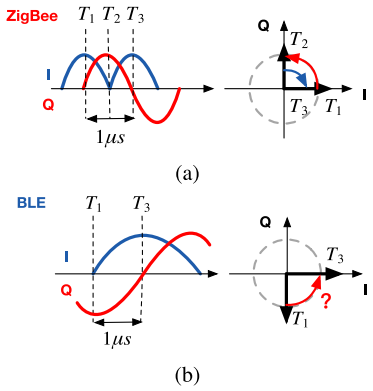


Fig. 6. Impact of inconsistent ZigBee phase shifts. (a) Inconsistent phase shifts at ZigBee. (b) Imperfect signal emulation at BLE.

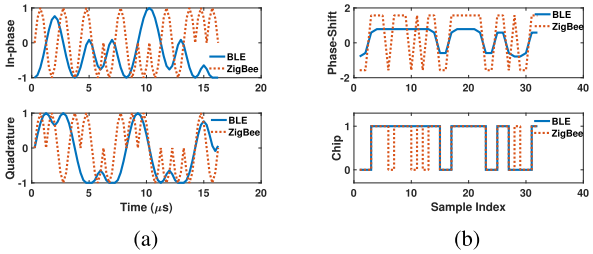


Fig. 7. Comparison between BLE emulated signal and the desired ZigBee signal. (a) Time domain emulated signal. (b) Constellation.

perfectly emulated, this is not the case for sequences '01' or '10'. Fig. 6a demonstrates ZigBee chip sequence of '10'. As shown in Fig. 6b BLE emulates this to be '11' (in the figure) or '00', incurring 1 chip error in either case. While such a chip error is inevitable due to the nature of BLE's narrower bandwidth, interestingly, its impact on decoded *bits* can be significantly reduced depending on the BLE phase shift. That is, by smartly emulating chip sequence '01' to either '11' or '00' (same to '10'), we are able to maximize the probability of DSSS to map the received chip sequences to the correct symbol, and to output correct bits. We discuss this in detail in the following section.

As a proof of concept example, we emulate a 32-chip ZigBee symbol '0' (i.e., '0000' in Table II) from BLE. In Fig. 7a, the time domain I/Q signals for both ZigBee and BLE are compared, which are quite different due to the disparate pulse shapes, i.e., Gaussian for BLE and half sine for ZigBee. As discussed earlier, phase shifts depicted in the upper part of Fig. 7b demonstrates that the shift per $0.5\mu s$ is $\pm\pi/4$ for BLE, where it is $\pm\pi/2$ for ZigBee. Moreover, some errors are observed where the phase shifts are inconsistent at ZigBee. This is also reflected in the chips (lower in Fig. 7b), which we consider in emulating DSSS so as to minimize the error in the decoded bits. This is explained in detail in the following section.

F. Optimal DSSS Emulation

In this section, we discuss how BlueBee minimizes the impact of the inevitable chip error introduced in OQPSK emulation, via DSSS. To start, let us first go through a simplified walk-through example: Fig 8 illustrates an emulation in the 4-bit hamming space (simplified from 32 bits in ZigBee DSSS).

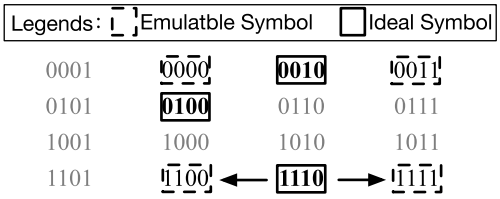


Fig. 8. An example of optimized DSSS emulation.

In this hamming space, there are three ZigBee symbols, called 'ideal symbols', which need to be emulated using the method introduced in Section IV-E. Due to the limited capability of BLE, BlueBee can only generate a limited number of emulation symbols, which are marked with dashed rectangles in this figure. Other symbols in this hamming space cannot be represented by BlueBee. Let S_i denote the i^{th} ideal symbol - the legitimate ZigBee symbol to be emulated, and E_i to denote the i^{th} emulated symbol which could be generated by BlueBee's emulation. Then, we define two symbol (Hamming) distances as follows:

Definition 1: $Dist(E_i, S_i)$ is intra hamming distance from the emulation symbol E_i to the ideal symbol S_i.

Definition 2: $Dist(E_i, S_j)$ is inter hamming distance from the emulation symbol E_i to the ideal symbol S_j, where $j \neq i$.

Take Fig. 8 for example. To emulate the ideal symbol '1110', BlueBee can generate two possible emulation symbols '1100' and '1111', which have the same intra-symbol distances of 1 to the target ideal symbol '1110'. After this, BlueBee considers the inter-symbol distance from these emulation symbol to the other ideal symbols different from the target symbol '1110'. For emulation symbol '1100', it has the inter-symbol distance of 1 and 3 to the ideal symbol '0100' and '0010' respectively. Similarly, for emulation symbol '1111', it has the inter-symbol distance of 3 and 3 respectively. As a result, BlueBee chooses the '1111' as the emulation choice, since it has the maximum value of the minimum inter-symbol distance (i.e., maximum margin), thus reducing the symbol error possibility in the emulation.

The previous example illustrates the idea of optimizing DSSS emulation in the 4-bit hamming space. Now we will talk about how BlueBee optimizes the DSSS emulation in the standard ZigBee symbol space, following the same principles.

G. Intra-Symbol Distance

Each 4-bit ZigBee symbol is mapped to 32 chips. Dividing the 32 chips into 16 consecutive pairs of chips and counting '01' or '10' yields the number of chip errors in the ZigBee emulation by BLE, or equivalently, $Dist(E_i, S_i)$ (i.e., intra-symbol Hamming distance). This value is constant for a given symbol, since emulation of '01' or '10' always induces one chip error, regardless of being emulated to '00' or '11'. For example, in Fig. 9, we have plotted the intra hamming distances for all possible ZigBee symbols. We find the maximum intra-symbol hamming distance is 8, such as the intra-symbol hamming distance of ZigBee symbol '0000'. Note that the intra-symbol hamming distance cannot be optimized, because there will always be one chip error at whatever bits BLE choose to emulate inconsistent ZigBee phase shifts.

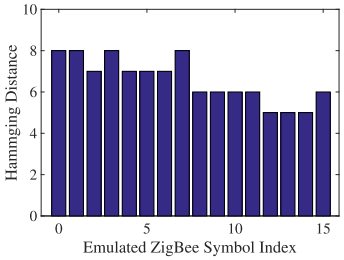


Fig. 9. Intra-symbol Hamming distance between emulated and ideal ZigBee symbols.

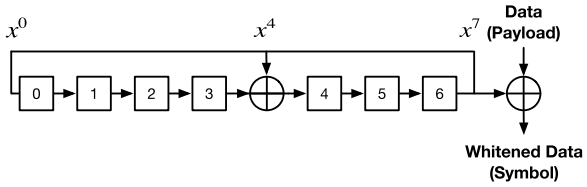


Fig. 10. BLE data whitening through LFSR.

H. Inter-Symbol Distance

Although the intra-symbol distance of each symbol is fixed, BlueBee tries to increase the inter-symbol distance for improving the reliability. this is because the inter-symbol Hamming distance $Dist(E_i, S_j)$, $i \neq j$, depends on how '01' or '10' are emulated. For example, '01' can be emulated via either '00' or '11'. Therefore, a ZigBee symbol can be emulated in $2^{Dist(E_i,S_i)}$ different sequences, where BlueBee chooses the emulation symbol with the maximum minimum inter-symbol hamming distance. This optimization can be described in the following equation:

$$\underset{E_i}{\text{argmax}} \quad min\{Dist(E_i, S_j), i \neq j\} \tag{1}$$

We note that the computation is lightweight with the limited search space of $0 \leq i, j \leq 15$. Furthermore, this only needs to be computed once, and thus can be precomputed and loaded on the device prior to running BlueBee.

I. Dealing With the BLE Data Whitening

Due to security concerns, the symbol transmitted by BLE is not the plain message of payload. Instead, a scramble technique called data whitening is adopted on BLE payload to randomize the matching between the payload bytes and the bytes transmitted in the air. Therefore, it is crucial to overcome the data whitening on BLE to control transmitted signal through BLE payload.

In fact, recent literature has shown that BLE's LFSR circuit is reversible [13], [30]; Technically, BLE uses the 7-bit linear feedback shift register (LFSR) circuit with the polynomial x^7+x^4+1 as shown in Fig. 10. The circuit is used to generate a sequence of bits to whiten the incoming data by XOR operation. The initial state of the LFSR circuit is the current channel number (i.e., from 0 to 39) in binary representation defined in the BLE specification. BlueBee reverse engineers the whitening process to generate the BLE payload according to the carefully chosen bytes for emulation. This makes BlueBee fully compatible with commodity BLE devices, validated with extensive testbed implementations and evaluations on commodity devices in Sec. VIII.

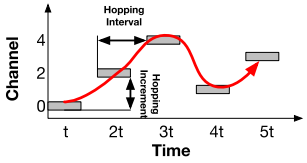


Fig. 11. BLE normal frequency hopping.

V. Concurrent Communication

One specific feature of BLE is the frequency hopping, which helps BLE devices to avoid busy channels occupied by other ISM band radios. In BlueBee, this feature allows one BLE device to hop among the $2.4GHz$ band and communicate with multiple ZigBee devices at different channels. For the convenience of discussion, we assume a static address configuration where the ZigBee channels and IDs are known in advance while leaving sophisticated addressing techniques for future studies. Furthermore, we can control the BLE frequency hopping sequence, while still following BLE frequency hopping protocol. In this section, we will first briefly introduce BLE frequency hopping protocol, followed by our design of two BlueBee channel scheduling solutions.

A. BLE Frequency Hopping

BLE has 40 $2MHz$ wide channels, labeled as channel 0 to channel 39. Among them, channel 37, 38, and 39 are advertising channels and the others are data channels. Once a connection is established on a advertising channel, two paired devices will hop among the data channels to communicate.

In BLE, a simple yet effective frequency hopping protocol is used to determine the next channel to hop. The first channel is always '0', and after a time duration of the *hopping interval*, the BLE device will hop to the next channel with an increment of *hopping increment*. In formula

$$C_{next} = C_{current} + hoppingInc \ (mod 37), \tag{2}$$

where C_{next} and $C_{current}$ indicate next and current channel respectively, 37 is the total number of BLE data channels, and hoppingInc is the hopping increment. In Fig. 11 we illustrate a frequency hopping sequence on 5 channels (i.e., channel '0' to channel '4') with a hopping increment of 2 and hopping interval of t.

To avoid collision with other wireless radios on the same ISM band, BLE adopts adaptive frequency hopping (AFH) when packet accept ratio is low on certain channels. In BLE AFH, a 37-bit channel map is used to maintain the channel link quality where '0' indicates a bad channel and '1' indicates a good channel. Let us use S_{good} and S_{bad} to indicate the good and bad channel sets respectively. Whenever the next channel will be a bad channel, it will be replaced by another channel in the S_{good}. More specifically, a *remapIndex* will be calculated through

$$remapIndex = C_{next} \ mod \ |S_{good}|, \tag{3}$$

and C_{next} will be replaced by $S_{good}(remapIndex)$. For example, in Fig. 12, $S_{good} = \{0, 3, 4\}$, while $S_{bad} = \{1, 2\}$. So whenever a BLE device hops to a bad channel, such as channel 1, its remapIndex will be calculated as $1 \ mod \ |S_{good}| = 1$, which indicates channel $S_{good}(1) = 3$ following the Equ. 3.

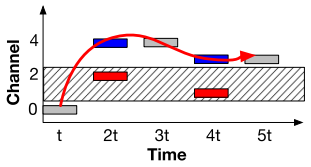


Fig. 12. BLE adaptive frequency hopping.

B. BlueBee Channel Scheduling

AFH will change the frequency a BLE device visits the channels. In Fig. 12, we illustrate the BLE channels visited before and after AFH. In the example, the BLE device starts from channel 0 and the hopping increment is 2, so that the hopping sequence is $\{0 \rightarrow 2 \rightarrow 4 \rightarrow 1 \rightarrow 3\}$. If the channel 1 and 2 are marked as 'bad channels', shown as the striped box, all the visits to these two channels will be remapped by AFH to channel 4 and 3 according to Equ. 3. The hopping sequence after AFH is $\{0 \rightarrow 4 \rightarrow 4 \rightarrow 3 \rightarrow 3\}$. AFH may affects two critical QoS criteria to the users:

i. Throughput. AFH changes the frequency of BLE to visit each channel. In CTC, we want BLE to visit the BLE-ZigBee overlapping channel, i.e., 2410, 2420,... 2470MHz more times to maximize CTC throughput.

ii. Fairness. The visits to BLE channels may differ with the AFH. We want the visits to the overlapping channels to be even so that ZigBee devices on different channels will have a chance to communicate with the BLE device.

To control BLE channel hopping in a non-disruptive manner, we can take advantage of the 37-bit channel bit map in BLE. Recall that the channel bit map records idle BLE channels and decides the hopping sequence. We propose to change the BLE hopping sequence by updating the channel bit map, which is a function available through a BLE host-level (i.e., user-level) command such as the $HCI\_set\_AFH\_Channel\_Classification$ [29].

1) Maximum-Throughput Solution: By updating the channel bit map, we can control the set of channels a BLE device can hop on. To maximize the throughput of concurrent BlueBee, we can leave the ZigBee-BLE overlapping channels in the channel bit map if they are marked as idle in the original channel bit map (i.e., S_{good}) while blacklisting the non-overlapping channels. So that the throughput on overlapping channels is maximized.

2) Load Balanced Solution: We can also achieve the fair channel hopping by updating the channel bit map. Thanks to the AFH, the output hopping sequence is almost uniformly distributed, although there are some channels visited one or two more times than the others in a run, i.e., 37 hops. We propose a heuristic algorithm that helps change the visits to a specific channel locally with minimum impact to other parts of the hopping sequence. The idea is to **remap the visits to a candidate channel to the target channel** by deliberately marking the candidate channel as 'bad'. So that the visits to the target channel increase. However, to reduce the impact on the whole hopping sequence, we maintain the size of S_{good}, which control the remap of other BLE devices. Specifically, for each under-visited channel c (i.e., visited less than other overlapping channels), we find another channel c' in S_{good} whose remapped channel will be c. Then we mark c' as a bad channel in the channel bit map, so that whenever BLE devices hop to c', the channel will be remapped to c. Since we do not want to disturb other remap index, we guarantee

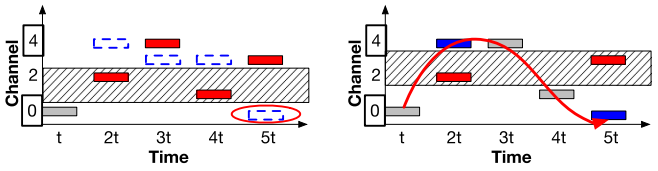


Fig. 13. The steps of BlueBee channel scheduling. (a) Choose a channel in S_{good} whose remapped channel will be the target channel. (b) Add a channel in S_{bad} to S_{good}.

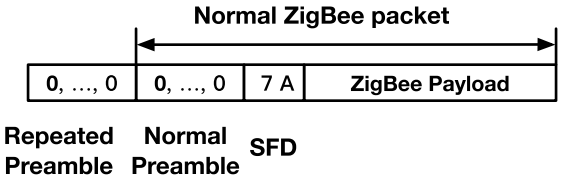


Fig. 14. Reliable CTC with repeated preamble.

the $|S_{good}|$ unchanged by marking one bad channel to be good in the channel bit map, so that $|S_{good}|$ still keeps the same.

In Fig. 13a and Fig. 13b we demonstrate our scheduling algorithm. In the example, we try to rebalance channel 0 and channel 4. We find channel 0 are visited less than channel 4, so we want to redirect frequency hopping to channel 0. We first assume all the channels need remapping (marked as red), except channel 0. Then we find the channel whose remapped channel will be channel 0, which is channel 3, as shown in Fig. 13a. We add channel 3 to S_{bad} to replace one channel in S_{bad}, i.e., channel 1, so that $|S_{good}|$ doesn't change as shown in Fig. 13b. Finally we have rebalanced channel 0 and channel 4. Admittedly, it is a best effort scheduling method, because sometimes it is unable to balance all the overlapped channels due to too many bad channels. In that case, we won't disrupt a lot of good channels to achieve the rebalance goal.

VI. Link Layer Protection

In the wireless communication, some predefined signal is transmitted before the payload for receivers to detect and synchronize to the on-air packet. For example, in the Zigbee a sequence of ZigBee symbol '0', known as the preamble and a ZigBee symbol 'a7', known as the start frame delimiter (SFD) are transmitted before the payload. In the ZigBee receiver, the ZigBee frames are detected via preamble detection procedure, i.e., searching for the preamble sequence of ZigBee symbol '0's.

Preamble is critical for the success of BlueBee reception because the failure in detecting preamble leads to the loss of the complete emulated ZigBee frame. However, we notice that due to the limitation of signal emulation in BlueBee, the preamble symbols are also imperfectly emulated. Thus, BlueBee further improves the reliability in the link layer by protecting the preamble detection procedure. In specific, we propose to repeat the emulated ZigBee preambles, i.e., the sequence of symbol '0', to improve the ZigBee preamble detection. As shown in Fig. 14, BlueBee emulates multiple repeated preambles which increase the probability of frame detection and the quality of synchronization. The evaluation demonstrates the repeated preamble improves frame reception ratio effectively.

VII. Discussion

A. Generic Emulation

This section discusses the possible extensions of emulation under other scenarios, e.g., Bluetooth classic and DBPSK (differential binary phase shift keying).

Similar to BLE, Bluetooth classic is also a popular Bluetooth standard, which is defined in Bluetooth core specification 1.0. These two standards have several differences in modulation and channels. First, although both standards adopt GFSK, Bluetooth Classic's modulation index is 0.35 while BLE's modulation index is between 0.45 and 0.55. The difference in modulation index affects the shape of the final signal. However, since the phase shift error brought by pulse shape can be well mediated through phase shift quantization at ZigBee receiver, BlueBee can still be used in Bluetooth Classic. Second, Bluetooth Classic has 79 channels distributed from $2402MHz$ to $2480MHz$ spaced $1MHz$ apart. So it can cover all ZigBee channels. Third, on the frequency hopping, Bluetooth Classic will hop among all 79 channels following a frequency hopping pattern calculated through the master device's MAC address and clock. Its hopping interval will always be $625\mu s$. The hopping interval is long enough to transmit a Bluetooth emulated ZigBee packets. Although the channel scheduling methods will be different, the same heuristic method can be used to find a channel scheduling solution.

In addition, BlueBee is able to emulate the DBPSK signal. By setting its payload in the transmitted BLE packets, BlueBee controls the phase shift values in the wireless waveform. When sampled at the receiver with DBPSK demodulation, the phase shifts will be quantized to positive/negative phase changes, and then demodulated to binary bits. As a result, BlueBee also suits for the DBPSK emulation with its emulation technique.

B. Feasibility of Reverse Communication

Although in this paper, we focus on the communication from BLE to ZigBee, in a real communication system, the reverse direction (e.g., CTC from ZigBee to BLE) is also needed to form bidirectional communication to support most upper layer protocol designs. The relationship between ZigBee chips and BLE bits in phase shift can be utilized in building the reverse communication, i.e., from ZigBee to BLE. The main idea is that a BLE receiver can listen to the ZigBee symbols in the air. From the demodulated BLE bits, which indicate the phase shift of ZigBee symbols in a lower sampling rate, the BLE receiver can recover the ZigBee symbols in the air. The reliability of the reverse communication, its compatibility to the commodity devices, and upper layer protocol designs, e.g., MAC layer coordination, are left as our future work.

C. Non-Disruptive to ZigBee and BLE Network

Due to the transparency of signal emulation technique to the receiver side, a BlueBee transmitter is non-disruptive to the ZigBee network. The normal ZigBee operations won't be disrupted by BlueBee. In Sec. VIII-E, we can see a ZigBee receiver can effortlessly receive packets from both the normal ZigBee transmitter and the BlueBee transmitter at the same time, showcasing that a BlueBee transmitter is regarded no more than another ZigBee device from the receiver's point of view.

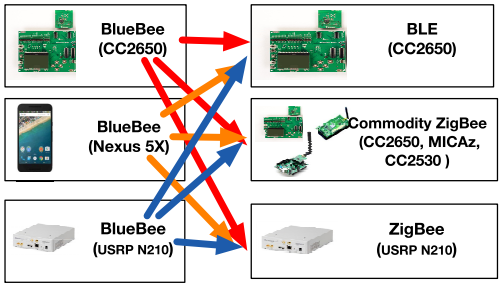


Fig. 15. Experiment Setting for BlueBee.

At the BlueBee transmitter, basically we are able to deliberately send BlueBee packets for CTC communication to achieve the best performance, such as embedding BlueBee packets in the BLE advertising packets. But we can further take advantages of two opportunities in BLE to embed BlueBee packets in a fully non-disruptive way: 1) The BLE data traffic is designed to be quite sparse for energy saving purpose. However, in idle time, the paired devices still need to exchange zero-length empty packets to keep synchronized and maintain the connection alive [11]. It provides the chance for us to send BlueBee packets when the BLE channels are idle. We can mark these packets to be CTC packets in the header so that the receiver can ignore them. 2) The 'zero-length' empty packets will be ignored by BLE devices without any impact on the higher layer data [11]. So that we might be able to deliberately mark the length of a BlueBee packet to be zero to be non-disruptive to the BLE network, although some hacking is needed at the sender.

VIII. Evaluation

In this section, we evaluate the performance of BlueBee across various domains, such as CTC performance comparison, communication reliability, support in mobility and low-duty cycle, and two demo applications of coexistence between ZigBee and BLE and smart light bulb control.

A. Platform Setting and Implementation

Fig. 15 demonstrates the evaluation platform of BlueBee, including (i) Software defined radio, i.e., the USRP-N210 platform with GNU radio BLE implementation [2]; (ii) Commodity ZigBee and BLE development boards, i.e, TI CC2650 (support both BLE and ZigBee), TI CC2530(ZigBee) and MICAz (ZigBee); (iii) Commodity smartphone, i.e., Nexus 5X. Note that the USRP is only used for its convenience in adjusting parameters, and BlueBee can be easily deployed on the aforementioned commodity devices. In addition, our platforms cover the latest BLE 4.2 devices, i.e., CC2650 and Nexus 5X, which support new features such as longer BLE frame size up to 257 bytes.

On BLE development boards, we modify only the payloads of the BLE packets to embed BlueBee bytes through the TI SmartRF software, which opens only user-level interfaces and guarantees the transmitted packets are fully BLE-compliant. On the smartphone, we use the Android *BluetoothLeAdvertiser* class to embed BlueBee bytes in the BLE advertising packets. In all of our implementation, only user-level operations are needed, so that BlueBee can be easily deployed on billions of existing BLE devices.

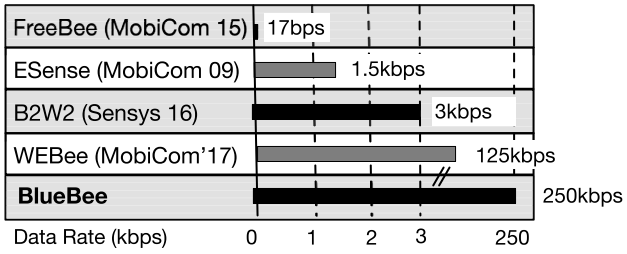


Fig. 16. Comparison with the state of the arts.

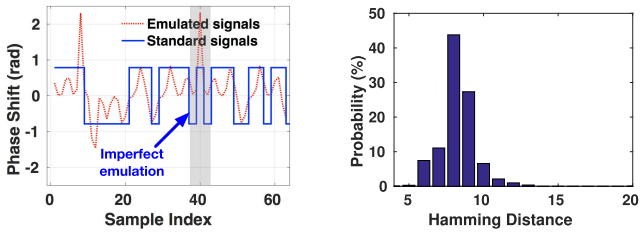


Fig. 17. Performance of phase shift emulation. (a) Phase shift of emulated and standard symbols. (b) Hamming distance of all emulated symbols.

B. CTC Data Rate

To evaluate the CTC data rate of BlueBee, we compare its throughput with the state-of-the-art packet level CTC methods.

1) Compare With FreeBee: The only state-of-the-art CTC work on BLE to ZigBee communication is FreeBee [18]. FreeBee's throughput is $17bps$ with a single CTC transmitter, while the data rate of BlueBee is $250kbps$, $14,000\times$ the data rate of FreeBee. Admittedly, FreeBee has its unique advantage of a free channel design, which differentiates it from those CTC designs that saturate the channel for high throughput. BlueBee can also beat existing packet-level CTC technologies that can saturate the channel for high data rate.

2) Compare With Other Packet-Level CTC: Here we compare BlueBee with other state-of-the-art packet-level CTC technologies, including Esense [5] (WiFi $\rightarrow$ ZigBee), and B^2W^2 [6] (BLE $\rightarrow$ WiFi) in data rate. Note that, these CTC techniques have a high-bandwidth radio (i.e., $20MHz$ WiFi radio) either at the sender or at the receiver. From Fig. 16, we can see that BlueBee can surpass the state-of-the-art packet-level CTC by $70\times - 100\times$. It indicates the intrinsic advantage of PHY-layer CTC over packet-level CTC.

3) Compare With PHY-Layer CTC: WEBee [19] is the latest PHY-layer CTC emulating ZigBee signal with a WiFi radio. Although both WEBee and BlueBee adopt the concept of signal emulation, there is a clear gap in each's design philosophy. With the power WiFi, WEBee simply manipulates the WiFi signal to mimic the ZigBee waveform. However, such emulation is not feasible for BLE due to its low data rate and simpler waveform. In contrast, BlueBee proposes to emulate signal that can be *demodulated correctly*, instead of similarity in the waveform, which is especially suitable for low-power devices in IoT scenario. The data rate of BlueBee is comparable with WEBee by about $2x$ that of WEBee on a single BLE channel, as shown in Fig. 16. However, it needs to be clarified that the high data rate does not come from better emulation, but rather from BLE's simple waveform without the uncontrollable cyclic prefix as in WiFi.

C. Emulation Reliability

Here we evaluate the emulation reliability of BlueBee, including PHY-layer reliability (i.e., phase shift and hamming distance) and link-layer reliability (i.e., frame reception ratio). To provide the details, we test these experiments under various situations, including different transmission power, distances, scenarios, and different packet duration.

1) Performance of Emulated Signals: Since BlueBee's BLE sender emulates the phase shifts in legitimate ZigBee frames, we first examine the performance of signal emulation.

Recall that in Section IV, ZigBee's OQPSK demodulation is based on the phase shifts, whose positive and negative signs will be further decoded as BLE symbol '1' and '0'. To verify the emulation performance, we use a USRP to dump the received chips and phase shifts of received ZigBee symbols. In Fig. 17a, we plot the phase shift of a received ZigBee symbol and an ideal ZigBee symbol. We find that BLE can emulate consistent phase shifts (i.e., slowly changing phase shifts) while failing to emulate inconsistent phase shifts (i.e., fast-changing phase shifts) due to its limited bandwidth. Note that the 64 samples for a ZigBee symbol are due to the oversampling of commodity ZigBee devices. The 64 samples will then be decimated to 32 chips for decoding. In Fig. 17b, the distribution of the Hamming distances of decoded ZigBee symbols is plotted. Due to the white Gaussian channel noise, the Hamming distances of emulated ZigBee symbols distributes in the range of $[6, 10]$ especially in $[8, 9]$, but mostly within the tolerance of the ZigBee demodulator (< 12), which guarantees the effectiveness of BlueBee.

While the emulation error in the OQPSK emulation, i.e., the intra-symbol distance, is fixed, the emulation error in the DSSS emulation, i.e., the inter-symbol distance, can be optimized. In other words, we choose the emulation E_i so that it is farthest from other ZigBee symbols S_j, where $j \neq i$ so that the emulation will not likely to be demodulated as other ZigBee symbols, as denoted in Equ. 1. In Fig. 18, we plot the minimum inter-symbol distance of each ZigBee symbol emulation to other legitimate ZigBee symbols without the DSSS emulation, denoted as the basic design, and with the DSSS emulation, shown as the improvement. For example, without the DSSS emulation, i.e., choosing the ZigBee emulation according to the first coming phase shift, the inter-symbol distance of the emulated ZigBee symbol '0' is only 3. The value will increase to 7 by adopting the DSSS emulation. The increase in inter-symbol distance provides the emulated ZigBee symbol '0' larger margin to other legitimate ZigBee symbols so that it can tolerate more noises and interference. We find the DSSS emulation benefits all the emulated ZigBee symbols except the symbol '2', which already has a large enough inter-symbol distance of 6, showing the effectiveness of the DSSS emulation.

2) Compare BlueBee With Normal ZigBee: Due to the transparency of signal emulation to the receiver, a BlueBee device can be viewed as a virtual ZigBee device. In Fig. 19, we compare the performance of a BlueBee device with a commodity ZigBee device in the frame reception ratio (FRR) under various Rx power. We choose two CC2650 development boards, which can work on either the BLE mode or ZigBee mode, for both the transmitter and the receiver to implement BlueBee, i.e., one CC2650 board on BLE mode and another on ZigBee mode and normal ZigBee communication, i.e., both

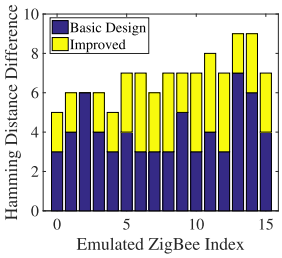


Fig. 18. H-distance improvement of DSSS emulation.

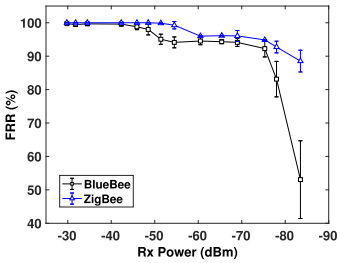


Fig. 19. Compare BlueBee with normal ZigBee.

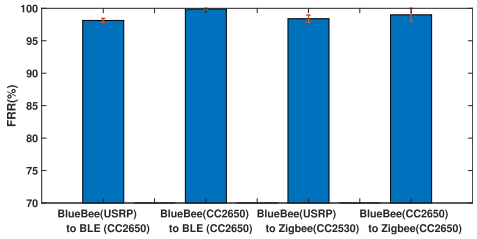


Fig. 20. BlueBee dual-standard compliance.

boards on ZigBee mode. We adjust the Tx power, between $-21dBm$ to $5dBm$, and the distance between the transmitter and receiver, between $30cm$ to $1m$, so that the Rx power is stable, i.e., within $\pm 0.5dBm$. In Fig. 19, we find the CC2650 chip can reach the FRR of normal ZigBee chip when the Rx power is above $-45dBm$, and slightly worse (within 5%) when the Rx power is above $-75dBm$. In addition, when the Rx power is above $-75dBm$, an FRR of above 90% is always guaranteed.

3) Dual-Standard Compliance: In BlueBee, a legitimate ZigBee packet is embedded in a legitimate BLE frame. To verify and evaluate such embedding, we have implemented BlueBee on various hardware devices, including 1) software defined radio, i.e., USRP N210 and 2) commodity BLE devices, i.e., CC2650 development kit. At the receiver side, we use both commodity BLE receiver (i.e., CC2650) and commodity ZigBee receiver (i.e., CC2530). As shown in Fig. 20, BlueBee, either the USRP implementation or commodity device implementation, can achieve over 99% frame reception ratio (FRR) at commodity BLE receiver, showing that it is a BLE compliant design. In addition, BlueBee's USRP and commodity device implementations can achieve an over 90% FRR and an over 99% FRR at the commodity ZigBee receiver, showing that it is also a ZigBee compliant design. Note that the FRR on commodity devices may vary on different hardware devices. The best result of 99% FRR comes from two CC2650 platforms with the sender running in BLE mode and the receiver running in ZigBee mode,

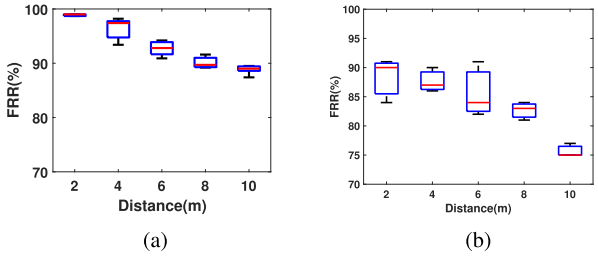


Fig. 21. FRR with distance. (a) FRR with distance on the USRP platform (lab). (b) FRR with distance on commodity devices (lab).

showing the effectiveness of the BlueBee design. The FRR on other ZigBee platforms, such as CC2530 and MICAz, is lower which we believe is due to vendor specific hardware designs, such as the matched filter, which will not affect homogeneous IoT network but only the heterogeneous network. To make the evaluation results more representative, later on we choose CC2530 ZigBee chip as the receiver without further explanation except the low duty cycle and mobility experiments, for MICAz motes provide the low duty cycle programming interface and a larger memory to store data in mobile applications.

The characteristic of dual-standard compliance indicates BlueBee can achieve *cross-technology broadcast*. That means we can construct a dual-standard frame where part of it is a ZigBee frame and part of it is a BLE frame. Each technology can identify their parts by detecting legitimate preamble and header while regarding the rest as noise.

4) Impact of Distance: We also evaluate the frame reception ratio (FRR) where the BLE sender sends out emulated ZigBee frames on both USRP and commodity CC2530 development kit. Fig. 21a depicts the FRR when USRP N210 emulates the ZigBee frames with the transmission power of 0dBm, the maximum energy level allowed in BLE standard [29]. In all the experiments, the average FRR is within 92% to 86%, demonstrating the reliability of BlueBee, at a transmission distance of 10m (the usual communication range between two BLE devices) Note that the FRR slightly decreases with the increase of distance, due to the lower SNR. Even so, in all the experiments, the FRRs are all above 85%. The experiments on commodity CC2530 development kit have a similar trend. During these experiments, the FRR is above 73% for all the different transmission distances.

5) Impact of Frame Duration: In BLE specification 4.2 [29], the maximum payload for BLE has been extended from $39bytes$ to $257bytes$, which means the frame duration will grow from around $0.3ms$ to over $2ms$. So we here study the impact of frame duration on BlueBee's performance. In Fig. 22, we study the FRR with frame duration ranging from $0.3ms$ to $1.2ms$, following the latest standard. We find that the increase in frame duration will slightly decrease FRR by about 2%. That is because a longer frame is usually more vulnerable to environmental noise and interference [28]. Even so, over 90% FRR shows BlueBee's resistance to the impact of the frame length.

6) Impact of Tx Power and Tx Distance: In Fig. 23 we study the frame reception ratio (FRR) of BlueBee with the impact of various Tx power and distance from a USRP to a commodity CC2530 ZigBee device for its convenience to control transmission power. We find that when Tx power

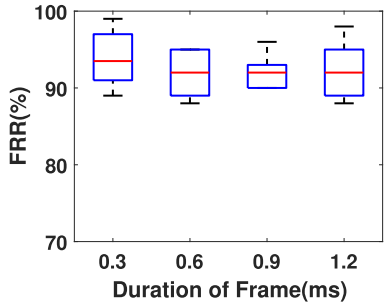


Fig. 22. FRR with frame duration.

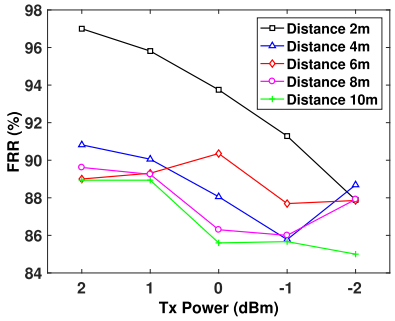


Fig. 23. FRR with Tx power.

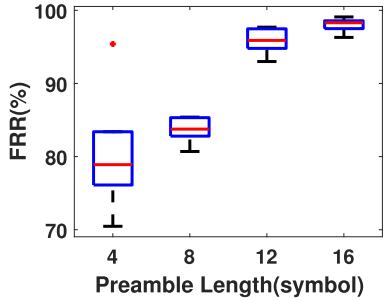


Fig. 24. FRR with preamble length.

increases from $-2dBm$ to $2dBm$, most FRR also increases from 85% to 90%. Then we fix the Tx power and study the FRR of BlueBee with different distances. We find that when the distance is as far as $10m$, the FRR is still over 80%. Note that the transmission power of a typical BLE device is $0dBm$ and the typical transmission range is $10m$. That means BlueBee can work well with typical BLE setting.

7) Protection in the Link Layer With Multiple Headers: In wireless communication, the preamble of a ZigBee packet plays critical roles in helping the receiver identify and synchronize to the packet in the air for further demodulation. In Fig. 24 we study the impact of the length of ZigBee preambles on the FRR and the proposed link layer preamble protection by repeating the preamble. Typical preamble length in ZigBee is 8 ZigBee symbols '0'.The number of '0's can be changed with at least four '0's. We change the length of ZigBee preamble from 4 symbols to 16 symbols which double the length of the preamble. We can see from the figure, with a typical preamble length of 8 symbols, FRR is about 84%. When we increase the preamble length to 12 symbols, the FRR jumps to about 95%, a 13% improvement. The experiments prove the effectiveness of our multiple preamble technique.

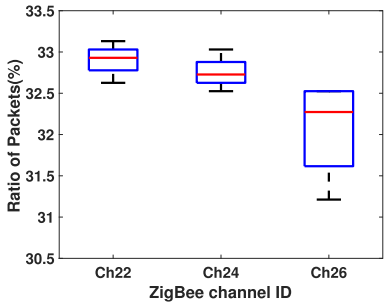


Fig. 25. Concurrent BlueBee on three ZigBee channels.

Even when we reduce the preamble length to 4 symbols, we find that the average FRR is still about 78%, which shows the robustness of BlueBee.

D. BlueBee Channel Scheduling

Here we evaluate the impact of BLE frequency hopping on the frame reception ratio. We implement the BlueBee scheduling algorithm on USRP N210-platform aiming at evenly distributing the BlueBee emulation frames on multiple channels. In the experiment, we set three ZigBee nodes, i.e., TelosB motes, working on ZigBee channel 22, 24, and 26, whose central frequencies are the same as BLE channel 27, 32, and 39. The USRP BLE transmitter runs BlueBee scheduling algorithm and sends 999 emulated ZigBee frames evenly on three channels and measures how many frames will be received at each ZigBee receivers.

In Fig. 25 we depict the number of successful receptions at each ZigBee channel. It is obvious that these ZigBee nodes working at different ZigBee channels are able to receive the similar number of frames (only 1% difference), demonstrating the BLE channel hopping will not affect BlueBee's FRR on the overlapping channels. Of course, different channels do show different wireless conditions. For example, the ratio of the received packet is slightly lower on ZigBee channel 26 because it overlaps with BLE advertising channel 39, which is crowded with BLE advertising packets.

E. Non-Disruptive to ZigBee Network

BlueBee is non-disruptive to the ZigBee network. To verify that, we compare the data rate of two scenarios: 1) a pure ZigBee network consisting of $[1, 4]$ ZigBee nodes and 2) a hybrid network consisting one BlueBee node and $[0, 3]$ ZigBee nodes, as shown in Fig. 26. In the experiment, both the ZigBee nodes and the BlueBee node will send $8bytes$ ZigBee frames every $100ms$. These nodes are turned on one after another and the accumulated data rates are plotted. We find the data rate of a single ZigBee/BlueBee device is about $640bps$. When the number of nodes increases, the data rate of each node will slightly decrease due to the share of the channel, but keep around $640bps$. In addition, the data rate of the BlueBee device shows no difference from and no much impact on other ZigBee devices, indicating that BlueBee can seamlessly work with other ZigBee devices.

F. Low Duty Cycle Support

In practical application scenarios, ZigBee devices may not be always on. Instead, they usually work in low duty cycle modes to save energy. Here we evaluate the performance

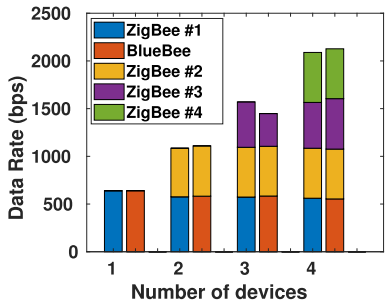


Fig. 26. BlueBee data rate w/ multiple ZigBee nodes.

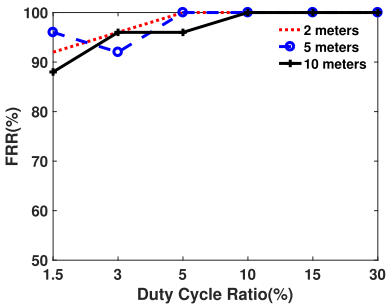


Fig. 27. Low duty cycle support.

of BlueBee with a ZigBee receiver working in low duty cycle mode. In the experiment, we transmit BLE frames from a USRP to some MICAz ZigBee receivers. The BLE transmitter's hopping interval is set to be $10ms$, within the range of available hopping interval in the standard. From Sec. V-A we know that BLE will always return to the start channel after 37 hops, which means the transmission interval of BLE to a ZigBee node at a specific channel will be $370ms$. To make successful CTC from BLE to ZigBee in low duty cycle mode, BLE will retransmit each frame 20 times, i.e., in 20 hops. As shown in Fig. 27, FRR increases when ZigBee's duty cycle becomes larger. When the duty cycle is larger than 10%, 100% FRR is reached. However, even when BLE's duty cycle is only 1.5%, an FRR of at least 88% still can be reached. This experiment indicates that BlueBee has the potential to be used in a low duty cycle as a long-lasting coordinator.

G. Mobility Support

BLE radios are widely used in the mobile scenario such as in smart wristbands. In this part, we study the impact of mobility on the performance of BlueBee. In the experiment, a USRP with BlueBee sender is put on a table broadcasting emulated ZigBee frames on a fixed channel. A person carrying a commodity ZigBee node, i.e., a MICAz node, is moving at different speeds, from about $0.5m/s$ to $8m/s$, at about $10m$ away. As shown in Fig. 28, there is only a slight decrease in FRR when the speed increases. Even when the person is running at speed $8m/s$, we can still achieve about 90% FRR. Both indoor and outdoor environment show similar results.

H. Application

1) Application I: Channel Coordination: One use case of BlueBee is the channel coordination between one BLE device and one ZigBee device. It is well known that BLE has a scheduled hopping sequence and won't avoid the ongoing ZigBee traffic, which makes the coordination between these

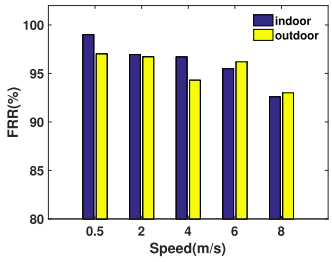


Fig. 28. BluBee's support for mobile scenario.

Fig. 29. BlueBee smart light bulb control.

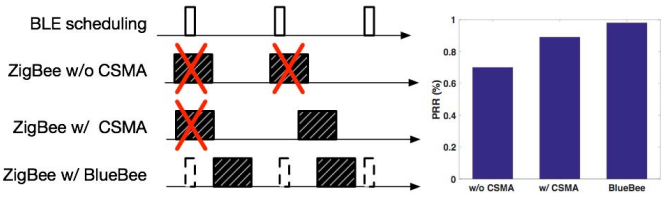


Fig. 30. Channel coordination between ZigBee and BLE.

two technologies quite challenging. The only way to avoid collisions based on existing MAC protocols is that ZigBee devices detect BLE traffic in advance and clear the channel for BLE through the CSMA protocol. Fortunately, with BlueBee, we can achieve a new coordination paradigm, where a BLE device can notify ZigBee of its channel hopping schedule to avoid potential collisions. In the left part of Fig. 30, we compare three ZigBee collision avoidance strategies with ongoing BLE traffic: (i) No collision avoidance, i.e., ZigBee with CSMA disabled; (ii) Normal collision avoidance, i.e., ZigBee with CSMA; and (iii) ZigBee is notified the slot length by BlueBee packets and schedule packet transmission before the next BLE packet.

The right part of Fig. 30 shows the experimental results. Compared with CSMA disabled, and normal CSMA coordination, BlueBee successfully improves the frame reception ratio to 98%, demonstrating that ZigBee can effectively avoid collision with BLE with the knowledge of BLE hopping schedule provided by BlueBee. It is a new paradigm in wireless coordination that the receiver can have the knowledge of the transmitter's schedule through CTC and paves the way for various cross-technology MAC designs in the future.

2) Application II: Smart Light Control: BlueBee can be easily deployed on commodity smartphones with BLE support, e.g., Nexus 5X smartphone, and benefit the smart home devices in real life. In Fig. 29, we implement BlueBee on a Nexus 5X smartphone to control ZigBee light bulbs at one of the overlapped channels, i.e., 2.48GHz. Available commands including the on/off status, the color, the intensity, and which light bulb to control. BlueBee achieves direct control of

ZigBee devices from a BLE radio without a ZigBee-BLE gateway [10] and any hardware modification at either side, and can be easily generalized to other IoT scenarios. It is a key enabler for other IoT cross-technology control design under commodity ZigBee and BLE devices. The related video can be found at [14].

IX. Related Work

The boom of diversified wireless technologies, such as WiFi, ZigBee, Bluetooth, and LTE-U in the last decade satisfies different needs for wireless connections. However, it makes the harmony coexistence of heterogeneous wireless technologies on the common ISM band an even more challenging and urgent problem. The efforts in the academia trying to relieve and solve this problem can be categorized as follows: (i) wireless communication under cross-technology interference, (ii) packet-level cross-technology communication, and (iii) physical layer signal manipulation between heterogeneous wireless devices. In contrast, BlueBee achieves efficient explicit communication between heterogeneous wireless devices with no hardware or firmware change, which creates a new paradigm for cross-technology communication and coordination.

A. Communication Under Cross-Technology Interference

Popular wireless technologies on the 2.4G ISM band, such as WiFi, ZigBee, Bluetooth, and LTE-U are competing for the spectrum. However, their heterogeneous PHY and asymmetric link blind them from detecting the existence of each other, which bring significant cross-technology interference(CTI) [3], [9], [16], [25]. To alleviate this issue, there have been numerous research works on alleviating the CTI by detecting and avoiding the interference, or recovering the corrupted signal from the interference [26]–[28], [31], [33]–[36], [38].

Although these solutions are effective, they generally require modification of the PHY layer mechanisms, or suffer from the dynamic interference patterns. In contrast, by enabling direct CTC among heterogeneities, BlueBee offers the capability of explicit channel coordination between BLE and ZigBee, thus alleviating the severe CTI problem, which has been demonstrated in Section VII-H.

B. Packet-Level Cross-Technology Communication

In recent years, researchers propose cross-technology communication (CTC) which directly builds the communication between heterogeneous devices [5], [6], [18], [34], [37]. The core idea of these CTC methods is that the sender creates special energy patterns by sending out legacy packets, while the receivers detect these patterns by either the received signal strength (RSS) sampling or the channel state information (CSI), which are supported by the existing hardware. For example, Esense [5], and HoWiES [37] build the CTC from WiFi to ZigBee by sending out multiple dedicated WiFi packets with specific packet durations to distinguish the CTC packets from background noises. FreeBee [18] relies on the mandatory WiFi beacons, and embed CTC message in a free channel by changing the transmission timings of existing beacons. GSense [34] attach customized preambles before heterogeneous wireless packets and exchange CTC information through the gaps between the customized preambles. B^2W^2 [6] introduces the CTC from Bluetooth to WiFi by modulating the energy level of Bluetooth packets and then demodulate through WiFi CSI at the receiver side. C-Morse [32] constructs a series of Morse code like long and short WiFi packets, which can be demodulated at the ZigBee receiver. However, the packet level CTC technologies commonly use coarse packet-level information, thus intrinsically suffer from the low throughput and long transmission delay. In contrast, BlueBee introduces the PHY-layer emulation for boosting the CTC throughput, thus offering the possibility of explicit channel coordination via CTC.

C. Physical Layer Signal Manipulation

The core idea of signal emulation in the BlueBee is inspired by several recent works studying the signal manipulation [4], [13], [17], [20], [23]. In [17], researchers build the legacy WiFi packets via customized tags by utilizing the backscatter effect. In addition, in the LTE system, Ultron [4]

emulates the WiFi packets via a LTE transmitter to coordinate between LTE and WiFi. However it requires the modification of existing LTE standard, and the sent frames are no longer LTE-compliant MAC frames. Different from these approaches, BlueBee does not require any modification to existing hardware, and is fully compatible with existing commodity Bluetooth and ZigBee hardware.

X. Conclusion

In this work we present BlueBee, a new PHY layer cross-technology communication technique proposing a direction of emulating legitimate ZigBee frames using BLE radio. BlueBee paves the road to practical CTC by offering over $10,000\times$ the throughput compared to the state-of-the-art CTC designs that rely on coarse-grained packet-level information. The emulation is achieved simply by selecting the payload bytes of BLE frames to provide unique dual-standard compliance and transparency where neither hardware nor firmware changes are required at the BLE senders and ZigBee receivers. BlueBee includes advanced features such as multi-channel concurrent CTC via adaptive frequency hopping. Comprehensive testbed evaluation on both USRP and commodity ZigBee/BLE devices show that BlueBee achieves 99% accuracy, while providing reliability under mobile and duty cycled scenarios.

References

[1] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, Standard IEEE 802.11, 1999.

[2] (2016). *Scapy Radio*. [Online]. Available: https://github.com/BastilleResearch/scapy-radio

[3] F. Adib, S. Kumar, O. Aryan, S. Gollakota, and D. Katabi, "Interference alignment by motion," in *Proc. MobiCom*, 2013, pp. 279–290.

[4] E. Chai, K. Sundaresan, M. A. Khojastepour, and S. Rangarajan, "LTE in unlicensed spectrum: Are we there yet?" in *Proc. MobiCom*, 2016, pp. 135–148.

[5] K. Chebrolu and A. Dhekne, "Esense: Communication through energy sensing," in *Proc. 15th Annu. Int. Conf. Mobile Comput. Netw.*, 2009, pp. 85–96.

[6] Z. Chi *et al.*, "B2w2: N-way concurrent communication for IoT devices," in *Proc. 14th ACM Conf. Embedded Netw. Sensor Syst. CD-ROM*, 2016, pp. 245–258.

[7] Gartner. (2016). *Gartner Report*. [Online]. Available: http://cloudtimes.org/2013/12/20/gartner-theinternet-of-things-willgrow-30-times-to-26-billion-by-2020/.

[8] M. Ha, K. Kwon, D. Kim, and P.-Y. Kong, "Dynamic and distributed load balancing scheme in multi-gateway based 6LoWPAN," in *Proc. IEEE Int. Conf. Internet of Things (iThings), IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput.*, Taipei, Taiwan, Sep. 2014, pp. 87–94.

[9] T. Hao, R. Zhou, G. Xing, M. W. Mutka, and J. Chen, "WizSync: Exploiting Wi-Fi infrastructure for clock synchronization in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1379–1392, Jun. 2014.

[10] M. Hawelikar and S. Tamhankar, "A design of Linux based ZigBee and bluetooth low energy wireless gateway for remote parameter monitoring," in *Proc. Int. Conf. Circuits, Power Comput. Technol.*, Mar. 2015, pp. 1–4.

[11] R. Heydon, *Bluetooth Low Energy: The Developer's Handbook*, vol. 1. Upper Saddle River, NJ, USA: Prentice-Hall, 2013.

[12] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting WiFi white space for ZigBee performance assurance," in *Proc. 18th IEEE Int. Conf. Netw. Protocols*, Oct. 2010, pp. 305–314.

[13] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith, "Inter-technology backscatter: Towards Internet connectivity for implanted devices," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 356–369.

[14] W. Jiang, R. Liu, L. Liu, and T. He. (2017). *BlueBee for Smart Light Control*, [Online]. Available: https://youtu.be/ZAtHBLRmEUs

[15] W. Jiang *et al.*, "BlueBee: A 10,000x faster cross-technology communication via PHY emulation," in *Proc. 15th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, New York, NY, USA, 2017, p. 3. doi: 10.1145/3131672.3131678.

[16] T. Jin, G. Noubir, and B. Sheng, "WiZi-Cloud: Application-transparent dual ZigBee-WiFi radios for low power Internet access," in *Proc. INFOCOM*, Apr. 2011, pp. 1593–1601.

[17] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "PASSIVE Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Proc. NSDI*, 2016, pp. 151–164.

[18] S. M. Kim and T. He, "FreeBee: Cross-technology communication via free side-channel," in *Proc. MOBICOM*, 2015, pp. 317–330.

[19] Z. Li and T. He, "Webee: Physical-layer cross-technology communication via emulation," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, 2017, pp. 2–14.

[20] Z. Li, Y. Xie, M. Li, and K. Jamieson, "Recitation: Rehearsing wireless packet reception in software," in *Proc. MobiCom*, 2015, pp. 291–303.

[21] C.-J. M. Liang, N. B. Priyantha, J. Liu, and A. Terzis, "Surviving Wi-Fi interference in low power ZigBee networks," in *Proc. 8th ACM Conf. Embedded Netw. Sensor Syst. (SenSys)*, 2010, pp. 309–322.

[22] S. Nastic, H.-L. Truong, and S. Dustdar, "SDG-Pro: A programming framework for software-defined IoT cloud gateways," *J. Internet Services Appl.*, vol. 6, no. 1, pp. 21-1–21-17, 2015.

[23] J. Ou, Y. Zheng, and M. Li, "MISC: Merging incorrect symbols using constellation diversity for 802.11 retransmission," in *Proc. INFOCOM*, Apr./May 2014, pp. 2472–2480.

[24] S. Qanbari, N. Behinaein, R. Rahimzadeh, and S. Dustdar, "Gatica: Linked sensed data enrichment and analytics middleware for IoT gateways," in *Proc. 3rd Int. Conf. Future Internet Things Cloud*, Rome, Italy, Aug. 2015, pp. 38–43.

[25] S. Rathinakumar, B. Radunovic, and M. K. Marina, "CPRecycle: Recycling cyclic prefix for versatile interference mitigation in OFDM based wireless systems," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.*, 2016, pp. 67–81.

[26] A. Saifullah *et al.*, "SNOW: Sensor network over white spaces," in *Proc. 14th ACM Conf. Embedded Netw. Sensor Syst. CD-ROM*, 2016, pp. 272–285.

[27] S. Sen, R. R. Choudhury, and S. Nelakuditi, "No time to countdown: Migrating backoff to the frequency domain," in *Proc. MobiCom*, 2011, pp. 241–252.

[28] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi, "Successive interference cancellation: Carving out MAC layer opportunities," *IEEE Trans. Mobile Comput.*, vol. 12, no. 2, pp. 346–357, Feb. 2013.

[29] B. Specification. (2011). *Bluetooth Technology Website*. [Online]. Available: http://www.bluetooth.com/

[30] D. Spill and A. Bittau, "BlueSniff: Eve meets alice and Bluetooth," in *Proc. WOOT*, vol. 7, 2007, pp. 1–10.

[31] S. singh *et al.*, "TRINITY: A practical transmitter cooperation framework to handle heterogeneous user profiles in wireless networks," in *Proc. MobiHoc*, 2015, pp. 297–306.

[32] Z. Yin, W. Jiang, S. M. Kim, and T. He, "C-Morse: Cross-technology communication with transparent Morse coding," in *Proc. INFOCOM*, May 2017, pp. 1–9.

[33] S. Yun and L. Qiu, "Supporting WiFi and LTE co-existence," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 810–818.

[34] X. Zhang and K. G. Shin, "Gap Sense: Lightweight coordination of heterogeneous wireless devices," in *Proc. INFOCOM*, Apr. 2013, pp. 3094–3101.

[35] X. Zhang and K. G. Shin, "Enabling coexistence of heterogeneous wireless systems: Case for ZigBee and WiFi," in *Proc. MobiHoc*, 2011, p. 6.

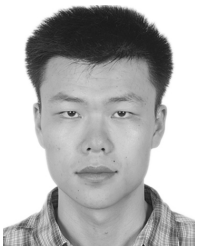
[36] X. Zhang and K. G. Shin, "Cooperative carrier signaling: Harmonizing coexisting WPAN and WLAN devices," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 426–439, Apr. 2013.

[37] S. Yun and L. Qiu, "Howies: A holistic approach to ZigBee assisted WiFi energy savings in mobile devices," in *Proc. INFOCOM*, Apr. 2013, pp. 1366–1374.

[38] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma, "ZiFi: Wireless LAN discovery via ZigBee interference signatures," in *Proc. 16th Annu. Int. Conf. Mobile Comput. Netw.*, 2010, pp. 49–60.

Wenchao Jiang received the B.S. and M.S. degrees from Shanghai Jiao Tong University, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Minnesota Twin Cities. His research interests include the Internet of Things, wireless networks, sensor networks, and mobile systems.

Zhimeng Yin received the B.S. and M.S. degrees from the Huazhong University of Science and Technology, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Minnesota Twin Cities. His research interests include the Internet of Things, mobile systems, and wireless networks.



Ruofeng Liu received the B.S. and M.S. degrees from Northwestern Polytechnical University, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Minnesota Twin Cities. His current research interests include the Internet of Things, wireless network, backscatter systems, and mobile systems.

Zhijun Li (M'18) received the M.S. and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology in 2001 and 2006, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, Harbin Institute of Technology. His research focuses on the wireless networks, the Internet of Things, and ubiquitous computing. He was a recipient of the Mobicom17 Best Paper Award.

Song Min Kim received the B.E. and M.E. degrees from the Department of Electrical and Computer Engineering, Korea University, in 2007 and 2009, respectively, and the Ph.D. degree from the Department of Computer Science and Engineering, University of Minnesota, in 2016. He was with the Department of Computer Science, George Mason University, USA. He is currently an Assistant Professor with the School of Electrical Engineering, KAIST, South Korea. His research interests include wireless and low-power embedded networks, mobile computing, the Internet of Things, and cyber–physical Systems. His research is supported by the National Science Foundation, the Commonwealth of Virginia, and the National Research Foundation in Korea. He received the Best Paper Award at the IEEE ICDCS 2018.

Tian He (F'18) is currently a Professor with the Department of Computer Science and Engineering, University of Minnesota Twin Cities. He is the author or coauthor of over 280 papers in premier network journals and conferences with over 22 000 citations (H-Index 65). His research interests include wireless networks, networked sensing systems, cyber–physical systems, real-time embedded systems, and distributed systems. He was the recipient of the NSF CAREER Award in 2009, the McKnight Land-Grant Chaired Professorship in 2011, the China NSF Outstanding Overseas Young Researcher I and II in 2012 and 2016, the George W. Taylor Distinguished Research Award in 2015, and seven best paper awards in international conferences, including MobiCom, SenSys, and ICDCS. He has served a few general/program chair positions in international conferences and on many program committees and also has been an editorial board member for six international journals. He is an ACM Fellow.