BlueBee: a 10,000x Faster Cross-Technology Communication via PHY Emulation

Wenchao Jiang
Department of Computer Science and Engineering, University of Minnesota
jiang832@umn.edu

Zhimeng Yin
Department of Computer Science and Engineering, University of Minnesota
yinxx283@umn.edu

Ruofeng Liu
Department of Computer Science and Engineering, University of Minnesota
liu4189@umn.edu

Zhijun Li*†
Department of Computer Science and Engineering, University of Minnesota
lixx5427@umn.edu

Song Min Kim
Department of Computer Science, George Mason University
song@gmu.edu

Tian He†
Department of Computer Science and Engineering, University of Minnesota
tianhe@umn.edu

ABSTRACT

Cross-Technology Communication is a promising solution proposed recently to the coexistence problem of heterogeneous wireless technologies in the ISM bands. The existing works use only the coarse-grained packet-level information for cross-technology modulation, suffering from a low throughput (e.g., 10bps). Our approach, called BlueBee, proposes a new direction by emulating legitimate ZigBee frames using a Bluetooth radio. Uniquely, BlueBee achieves dual-standard compliance and transparency by selecting only the payload of Bluetooth frames, requiring neither hardware nor firmware changes at the Bluetooth senders and ZigBee receivers. Our implementation on both USRP and commodity devices shows that BlueBee can achieve a more than 99% accuracy and a throughput 10,000x faster than the state-of-the-art CTC reported so far.

CCS CONCEPTS

• **Networks** → *Wireless personal area networks*;

KEYWORDS

Cross Technology Communication; Signal Emulation; Bluetooth Low Energy, ZigBee, Internet of Things

ACM Reference Format:
Wenchao Jiang, Zhimeng Yin, Ruofeng Liu, Zhijun Li, Song Min Kim, and Tian He. 2017. BlueBee: a 10,000x Faster Cross-Technology Communication via PHY Emulation. In *Proceedings of 15th ACM Conference on Embedded Networked Sensor Systems (SenSys'17).* ACM, New York, NY, USA, 13 pages. https://doi.org/http://dx.doi.org/10.1145/3131672.3131678

*Zhijun Li is a visiting professor at the University of Minnesota and officially affiliated with Harbin Institute of Technology, China.
†Zhijun Li and Tian He are the corresponding authors of this work.

SenSys'17, November 6–8, 2017, Delft, The Netherlands

ACM ISBN 978-1-4503-5459-2/17/11…$15.00
https://doi.org/http://dx.doi.org/10.1145/3131672.3131678

1 INTRODUCTION

The body of wireless devices has undergone 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 [12]. The dense deployment induces 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 – by taking advantages of specialized features among heterogeneous wireless technologies, collaboration enables them to reach beyond independent operation. For example, in ZiFi [44] 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 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, latest literature introduces Cross-technology communication (CTC) techniques which achieve direct communication among heterogeneous wireless devices with incompatible PHY layer, using legacy devices. Such techniques commonly use packet-level modulations, where the combinations of timing [22] and durations [7]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 18bps [22]. This not only constraints the usage, but also indicates spectrum inefficiency compared to 250kbps and 1Mbps if used for legacy ZigBee and Bluetooth.

This paper introduces BlueBee, which paves the way to practical CTC via physical-layer emulation. By smartly selecting the payload bits in a Bluetooth packet, BlueBee effectively encapsulates a ZigBee packet within a Bluetooth packet payload. This is fully compatible with legacy ZigBee devices while reaching the ZigBee bitrate cap of 250kbps. In other words, BlueBee does not require any hardware or firmware change to either Bluetooth transmitter or ZigBee receiver, offering full compatibility (i.e., implementable as an application) to existing billions of commodity IoT devices, smartphones, PCs, and peripherals.

The emulated ZigBee packet transmitted from Bluetooth is, in fact, indistinguishable by the ZigBee receivers. This is surprising, 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) similarity of (de)modulation techniques of Bluetooth and ZigBee and (ii) error tolerance of ZigBee demodulation (OQPSK/DSSS). Specifically, both technologies use the phase differences between samples, referred to as *phase shifts*, to indicate symbols, which makes emulation possible. 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 OQPSK/DSSS demodulator. BlueBee effortlessly runs on commodity Bluetooth devices by simply putting specific bit patterns in the Bluetooth packet payload. It achieves 250kpbs at 90% frame reception ratio (FRR), 10,000 times faster than the state-of-the-art [22]. Also, BlueBee effectively utilizes the frequency hopping feature of Bluetooth to support concurrent 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. The design does not require any modification to the hardware or the firmware, for either the transmitter (Bluetooth) or the receiver (ZigBee), enabling full compatibility to billions of existing commodity devices.
- We address several unique challenges of signal emulation, including (i) optimized ZigBee phase shifts emulation using Bluetooth signal, (ii) supporting concurrent communication and low duty cycle operation under frequency hopping of Bluetooth, and (iii) link layer reliability under dynamic channel conditions. These solutions offer general insights for other signal emulation between 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 to a 18bps rate achieved by the state-of-the-art CTC from Bluetooth to ZigBee [22], BlueBee's reliable throughput of 225kbps indicates performance gain of more than 10,000 times!

2 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 [17, 24, 40]. That is because the wireless technologies coexisting in the ISM band have heterogeneous PHY layer and can not communicate directly, thus not able to effectively coordinate channel use. To achieve effective channel use, the traditional approach is to use a multi-channel gateway. And recently, researchers also propose cross-technology communication (CTC) techniques as a promising solution for channel coordination. However, neither the traditional gateway approach nor the existing CTC technologies work well for the channel coordination due to their intrinsic limitations.

• **Limitation of Gateway**. Multi-radio gateway is a usual and straightforward solution to bridge multi-technology communicatio [13, 14, 20, 26, 29]. 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. Also, 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 direct communication among heterogenous wireless technologies, thus make explicit channel coordination available. 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 our knowledge, existing CTC designs [7, 22, 43] rely on sparse packet level information such as the beacon timing [22] and multi-packet sequence patterns [37], introducing a delay of at least hundreds of milliseconds. Such a delay prevents the existing solutions from coordinating channels effectively in real-time.

In contrast to the limitations of gateway approach and existing CTC approaches, BlueBee is able to transmit a ZigBee packet directly from a Bluetooth radio within a few milliseconds, for the first time, making channel coordination feasible. In the paper, although our description will be based on one specific Bluetooth protocol, Bluetooth Low Energy (BLE), the idea can be generalized to other Bluetooth protocols, such as Bluetooth Classic (discussed in Section 7.1).

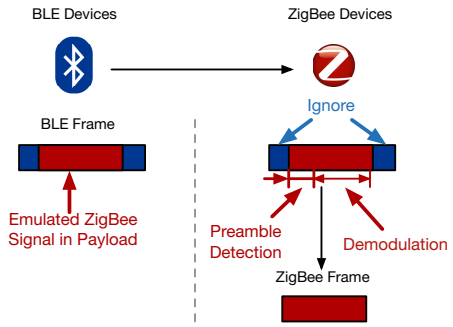


Figure 1: The system architecture of the BlueBee.

3 BLUEBEE IN A NUTSHELL

Overview. BlueBee is a high data-rate CTC communication from BLE to ZigBee, while being compatible to both ZigBee and BLE protocols. 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-emulated BLE packet reaches a ZigBee device, the payload part is detected (via preamble) and demodulated, just like any other ZigBee packet originated from a ZigBee sender. We note that the header and trailer of the BLE frame are incompatible to ZigBee and is naturally disregarded, or equivalently, treated as noise. In fact, such a design makes BlueBee transparent; At the sender, the BLE device can

not distinguish whether it is a normal BLE packet or it contains emulated ZigBee frame because it is merely a byte pattern in the payload. Conversely, at the receiver, the ZigBee device can not tell whether the frame is from a ZigBee device or is emulated by a BLE device, due to the indistinguishable PHY layer waveform.

	Cost	Spectrum Efficiency	Throuput	Multi-channel CTC
Gateway	Medium	Medium	High	Not Support
ESense [7]	Low	Low	Low	Not Support
FreeBee [22]	Low	Medium	Low	Not Support
B^2W^2 **[11]**	Low	Medium	Low	Support
BlueBee	**Low**	**High**	**High**	**Support**

Table 1: Comparison of BlueBee and existing CTC solutions

Unique Features. In Table 1, 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. BlueBee overcomes the shortages of existing gateway approach by providing direct communication between heterogeneous devices. As opposed to the gateway, BlueBee does not incur deployment cost or additional traffic. At the same time, it offers significantly higher communication throughput and lower transmission delay compared to the CTC presented until now. Also, BlueBee enables multi-channel concurrent CTC by the inherent frequency hopping in the BLE communication.

BlueBee also has a few innovative and unique features in compatibility: First, it is the first CTC design from BLE to ZigBee that requires neither hardware nor firmware change. Other designs require at least firmware changes [7, 22, 37] at the receivers. Second, BlueBee is "dual-standard compliance" in a sense that a frame can be received and demodulated by both ZigBee and BLE receivers.

4 BLUEBEE DESIGN

This section illustrates the BlueBee design in detail.

4.1 Background

We first give a brief introduction of how a BLE transmitter and a ZigBee receiver work in relation to BlueBee, followed by the feasibility of signal emulation.

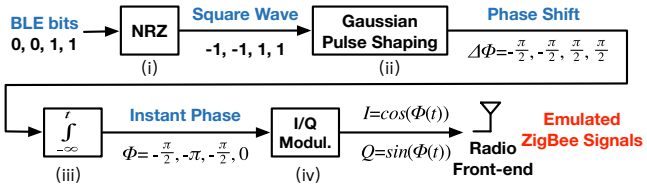


Figure 2: BLE as the transmitter with GFSK modulation.

BLE Transmitter. BLE uses Gaussian Frequency Shift Keying (GFSK) modulation, which is normally realized by phase shift over time [1]. Fig. 2 illustrates the entire procedure from payload bits to corresponding radio waves from steps (i) to (iv). In (i) BLE bits first go through a non-return-to-zero (NRZ) module that modulates series of BLE bits to series of squarewaves with amplitudes of either -1 or 1. Since each wave is $1\mu s$ long and carries a single bit, this leads to the 1Mbps bitrate of BLE. (ii) This wave passes through the Gaussian low pass filter, which shapes the waves into a band-limited signal. This baseband signal corresponds to phase shifts of $\pm\pi/2$ when multiplied to the carrier. (iii) Taking integral of the series of waves to t yields phase with respect to time (i.e., instant phase). This is essentially a time-domain representation of the accumulated phase shifts from the previous step. (iv) The In-phase and Quadrature (I/Q) signal is 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.

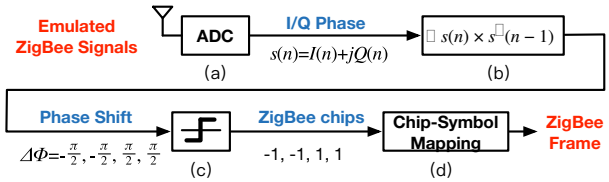


Figure 3: ZigBee as the receiver with OQPSK demodulation.

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 2) predefined in DSSS. There are 16 different symbols where each represents $log_2 16 = 4$ bits. We note that in the face of noise/interference the

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

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

Table 2: Symbol-to-chip mapping in ZigBee (802.15.4)

phases may suffer from errors (+ ↔ −), which induce reversed chips (1 ↔ 0). In such case, the closest symbol with smallest Hamming distance is selected.

4.2 Opportunities and Challenges of Emulation

Conceptually, emulation of ZigBee signal via BLE is possible due to two key technical insights. First is the similarity of (de)modulation techniques of BLE and ZigBee. That is, BLE's GFSK and ZigBee's OQPSK commonly utilize phase shifts between consecutive samples to indicate symbols (chips for ZigBee). Furthermore, ZigBee only considers *sign* (+ or -) of the phase instead of a particular phase value, which offers great flexibility in emulation. However, the challenge comes from the fact that the bandwidth of BLE (1MHz) is only half of that of ZigBee (2MHz). This fundamentally limits BLE's rate of phase shifts. In other words, phase shifts in BLE are not sufficiently fast to express all ZigBee chips, leading to inevitable errors in emulation. This shortage is covered by the second key component to BlueBee emulation – i.e., DSSS in ZigBee.

DSSS maps 32bit chip sequences to 4bit symbols (Table 2), leaving tolerance margin for robustness against noise and interference. Due to this margin, 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 [24]). This tolerance margin can be exploited to recover from the inevitable error caused by the bandwidth asymmetry. In the following sections, we provide a detailed illustration on the two insights, and how BlueBee is designed to effectively explore them to enable CTC.

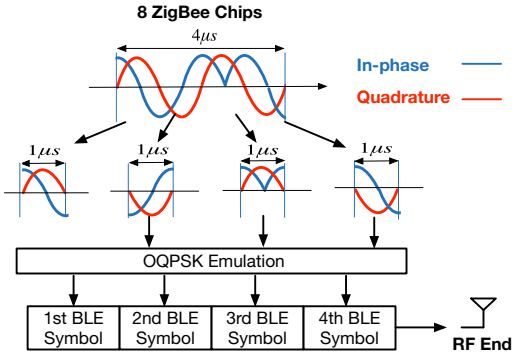
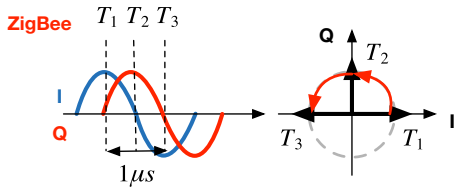


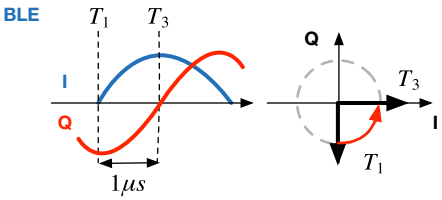
Figure 4: Emulating ZigBee with BLE

4.3 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 $1\mu s$. Each of the pieces is then emulated to be a BLE symbol which we will discuss in detail in the following section. We note that the technique introduced only involves setting bit pattern of BLE packet payload, and does not enforce any change to hardware or firmware.



(a) ZigBee signal with two chips, '11'.



(b) Emulation of (a) by BLE

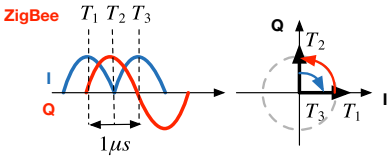
Figure 5: (a) ZigBee signal indicating two chips, '11', as phase shifts from T_1 to T_2, and from T_2 to T_3 are both positive ($\pi/2$). (b) is the emulated signal of (a), by BLE (which is in fact BLE symbol '1'). When fed into ZigBee receiver this signal is sampled at T_1, T_2, and T_3 to give two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

Let us now look into how the emulation is performed on a two-chip piece divided in Fig. 4. 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. 5a 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 chip of '1'. The next chip is computed similarly between samples T_2 and T_3, which also yields a chip of '1'.

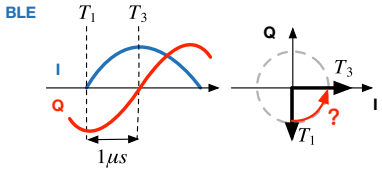
Now we show that the above mentioned ZigBee signal can be successfully emulated by BLE, which is demonstrated in the left in Fig. 5b. Although the signal appears to be distinct from ZigBee signal (left in Fig. 5a), it still delivers the same chips of '11' to ZigBee receiver. The key point here is that only *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, phase shift between T_1 and T_2 is $\pi/4$ (i.e., positive), which yields 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 the Fig. 5b is simply a BLE signal representing 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 to transmit bits corresponding to phase shift of $\pi/2$, the BLE devices is 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.



(a) Inconsistent phase shifts at ZigBee

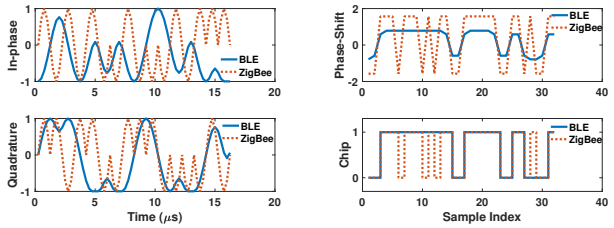


(b) Imperfect signal emulation at BLE

Figure 6: Impact of inconsistent ZigBee phase shifts

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 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 cases. 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 2) 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 the Fig. 7b demonstrate 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.



(a) Time domain emulated signal **(b) Phase shifts of emulated signal**

Figure 7: Comparison between BLE emulated signal and the desired ZigBee signal

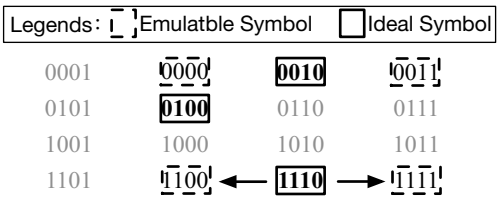


Figure 8: An example of optimized emulation

4.4 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 in ZigBee DSSS). In this hamming space, there are three ideal symbols, which need to be emulated using the method introduced in Section 4.3. Due to the limited capability of BLE, BlueBee can only generate 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, and E_i to denote the i^{th} emulated symbol. Then, we define two symbol (Hamming) distances as follows:

Definition 4.1. Intra symbol distance $Dist(E_i, S_i)$ is hamming distance from the emulation symbol E_i to the ideal symbol S_i.

Definition 4.2. Inter symbol distance $Dist(E_i, S_j)$ is 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 generates two emulatable symbols '1100' and '1111', which have the same intra symbol distances of 1. After this, BlueBee considers the inter symbol distance from these emulation symbol to the other two ideal symbols. 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).

The previous example illustrates the idea of 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.

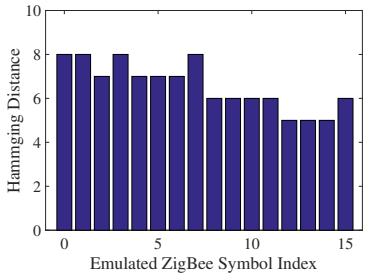


Figure 9: Intra symbol Hamming distance between emulated and ideal ZigBee symbols

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 induce 1 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 hamming distance is 8, such as the intra hamming distance of ZigBee symbol '0000' . Note that the intra symbol hamming distance can not be optimized, because there will always be one chip error at whatever bits BLE choose to emulate inconsistent ZigBee phase shifts.

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 light weight 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.

4.5 Dealing with the BLE Data Whitening

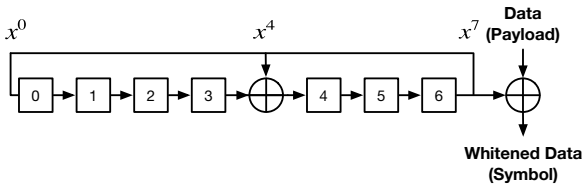


Figure 10: BLE data whitening through LFSR

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 have shown that BLE's LFSR circuit is reversible [19, 35]; 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 to commodity BLE devices, validated with extensive testbed implementations and evaluations on commodity devices in Sec. 8.

5 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. Furthermore, we can control BLE frequency hopping sequence, while still following BLE frequency hopping protocol. In this section, we will first introduce briefly BLE frequency hopping protocol, followed by our design of two BlueBee channel scheduling solutions.

5.1 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 connection is established on the advertising channels, two paired devices will hop among the data channels.

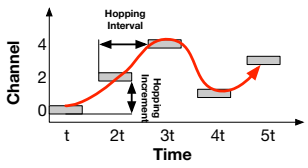


Figure 11: BLE normal frequency hopping

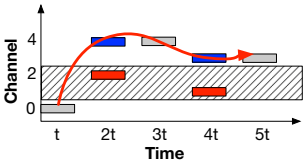


Figure 12: BLE adaptive frequency hopping

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 *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, the channel 1 and channel 2 are bad channels. So whenever BLE devices hop to these two channels, they will be remapped to channel 3 and channel 4 respectively following the Equ. 3.

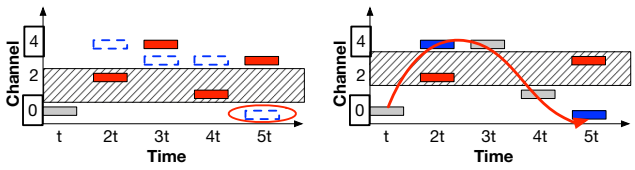
5.2 BlueBee Channel Scheduling

With AFH, the frequency to visit different channels becomes uneven. For example, in Fig. 12, channel 0 will only be visited once during one hopping period (i.e., 5 hops in the example) half the frequency of channel 3 and 4. In real network environment, AFH will cause unfair services to ZigBee nodes at BLE-ZigBee overlapped channels (i.e., 2410, 2420, ... 2480MHz). In other words, the QoS of BlueBee can not be guaranteed. To resolve this issue, we want to balance BLE's frequency of visiting overlapped channels in a non-disruptive way.

To achieve that, we can take advantage of the 37-bit channel bit map in BLE. As mentioned earlier, the channel bit map is used to calculate the next channel to hop if AFH is enabled. In addition, current BLE protocol supports the update of the channel bit map during normal transmission to adapt to the fast-changing network environment. So we can control the hopping behavior of BLE by only updating the channel bit map. For different optimization goals in application scenarios, we propose two concurrent BlueBee solutions.

Maximum-throughput solution. By updating the channel bit map, we can control the set of channels BLE device can hop on. To maximize the throughput of concurrent BlueBee, we can leave the ZigBee-BLE overlapped 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-overlapped channels. The channel bit map needs to be set only once in the connection initialization stage, so the network overhead is very low. Note that what we do is just choosing a subset channels from the idle channels, so we will not disrupt the original functionality of BLE channel hopping, which is to avoid channel collision. In addition, such change is supported by BLE standard through host (i.e., user) level commands such as the $HCI\_set\_AFH\_Channel\_Classification$ [34].

Load balanced solution. In some scenarios, the fairness of CTC is more important, such as the multi-channel synchronization problem. The maximum-throughput design may not guarantee a load balanced CTC on different channels. Here a simple yet effective heuristic method is proposed to balance the BLE's frequency hopping on overlapped channels while still being compliant to BLE's AFH protocol. More specifically, we can balance the BlueBee traffic by slightly modify S_{good} and S_{bad} in the channel bit map. The basic idea is that for each unbalanced channel c (i.e., visited less than other overlapped 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. Of course we need to guarantee $|S_{good}|$ unchanged so that the remapIndex is unchanged. To do that we choose to mark one bad channel to be good in the channel bit map, so that $|S_{good}|$ still keeps the same.



(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}

Figure 13: The steps of BlueBee channel scheduling

In Fig. 13a and Fig. 13b we illustrate 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 one. 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 channel to achieve the rebalance goal.

6 LINK LAYER PROTECTION

In this section, we introduce the link layer protection method of BluBee, i.e. multiple preambles, link layer coding, and the adaptive protection based on BLE link statistics.

6.1 Frame Retransmission

To improve the transmission reliability, BlueBee can transmit the same Bluetooth packet multiple times for emulating the ZigBee packets, in case some of the emulated ZigBee packet is dropped at the receiver side. The ZigBee receiver is able to receive the correct information if there is at least one copy of the same ZigBee packet is correctly received, i.e., the packet passes through the CRC checksum as specified in the 802.15.4 standard [18]. The frame retransmission technique is naturally compatible with the ZigBee protocol at the receiver side. That is because the ZigBee receiver will automatically ignore retransmitted ZigBee frames if it has already received one according to the 802.15.4 standard.

The number of frame retransmission is related to the frame reception ratio (FRR). Assuming that the reception of each emulated ZigBee frame is independent of the others, after transmitting k copies, we will successfully receive at least one ZigBee frame with probability $1 - (1 - FRR)^k$. As demonstrated in our experiment, the successful reception of the BlueBee packet varies with different SNR situations. Supposing we have a FRR of 70%, then after 6 retransmissions, the final successful reception rate is more than 99.9%, suggesting that BlueBee can achieve a very high FRR by simply retransmitting the emulated frames. Note that the retransmission

will not cause significant overhead to the channel efficiency, since CTC is usually used for the control purpose with little total traffic demand.

6.2 Repeated Preamble

In addition to the frame retransmission technique mentioned above, BlueBee also utilizes the repeated preamble technique to further improve the reliability. In the commodity ZigBee chips, the demodulation of possible ZigBee packets starts by searching for the specific preamble, which consists of eight '0' symbols, followed by the symbols 'a7', which is the start frame delimiter (SFD). Since this preamble detection is done before ZigBee can receive any frames, it cannot be protected using upper layer coding. To improve the packet reception rate, BlueBee sends out multiple repeated preambles, as shown in Fig. 14. If the first preamble is successfully received, the ZigBee receiver will then discard the remaining preambles in the upper layer decoding. Otherwise, ZigBee still has a second chance to detect the preamble.

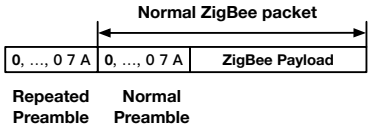


Figure 14: Reliable CTC with repeated preamble

7 DISCUSSION

7.1 Compatibility with Bluetooth Classic

BLE is defined in Bluetooth core specification 4.0 [34]. Another well known Bluetooth technique is Bluetooth Classic, defined in Bluetooth core specification 1.0. There are some connection and distinctive difference between these two techniques. First, in modulation, although both adopts 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. As mentioned earlier, the phase shift error brought by pulse shape can be mediated through phase shift quantization at ZigBee receiver, which means 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 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.

7.2 Feasibility of Reverse Communication

Although in this paper, we focus on the communication from BLE to ZigBee, the reverse communication (e.g., CTC from ZigBee to BLE) might be needed to provide the feedback (e.g., ACKs for BLE to ZigBee packets) from ZigBee. The reverse communication from ZigBee to BLE is also possible through the phase shift emulation. More specifically, due to the similarity in (de)modulation, a BLE receiver can get the information about the phase shifts of a ZigBee symbol in the air, but only in coarse grain (i.e., 1Mbps BLE data rate compared to the 2Mbps ZigBee chips) restricted to its limited bandwidth. However, a BLE receiver is still able to derive the corresponding ZigBee symbols from the detected phase shift information because ZigBee chips are redundant. We will make the communication from ZigBee to BLE and its compatibility with commodity devices our future work.

8 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 the example application of coexistence between ZigBee and BLE.

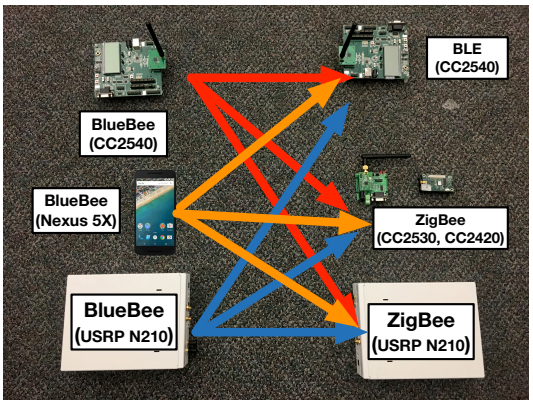


Figure 15: Experiment Setting for BlueBee

8.1 Platform Setting

Fig. 15 demonstrates the evaluation platform of BlueBee. We have implemented BlueBee as a sender on (i) a GNU radio BLE implementation called scapy radio [3] with a USRP-N210 platform, (ii) a commodity BLE CC2540 development kit [2], and (iii) a commodity smartphone Nexus 5X. Note that, we use USRP here only for its convenience to change parameters in the experiments. Our design is compatible with the widely used BLE 4.0 chips, such as CC2540, as well as smartphones with the latest BLE 4.2 protocol, such as Nexus 5X, which is back compatible to BLE 4.0 and supports the long BLE frame up to 257 bytes.

As for the receiver side, we have tested the BlueBee on the following platforms: 1) A commodity BLE receiver (i.e., CC2540 development kit); 2) Commodity ZigBee receivers including CC2530 and CC2420 (i.e., MICAz and TelosB); and 3) 802.15.4 implementation on USRP N210 to provide detailed examination of the PHY level emulation performance. The arrows from BlueBee to three receivers indicate that a broadcast frame from BlueBee (either USRP or commodity devices) can be decoded by both commodity ZigBee receivers and commodity BLE receivers simultaneously, indicating the emulated frames are both BLE-compliant and ZigBee-compliant.

8.2 CTC Throughput

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

8.2.1 Compare with FreeBee. The only state-of-the-art CTC work on BLE to ZigBee communication is FreeBee [22]. FreeBee's throughput is 17*bps* with a single CTC transmitter, while the throughput of BlueBee is 225*kbps*, 13,000× the throughput 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 throughput.

8.2.2 Compare with Other Packet-Level CTC. Here we compare BlueBee with other state-of-the-art packet-level CTC technologies, including Esense (WiFi → ZigBee), and B^2W^2 (BLE → WiFi) in throughput. Note that, these CTC techniques have a high-bandwidth radio (i.e., 20*MHz* 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 × −100×. It indicates the intrinsic advantage of PHY-layer CTC over packet-level CTC.

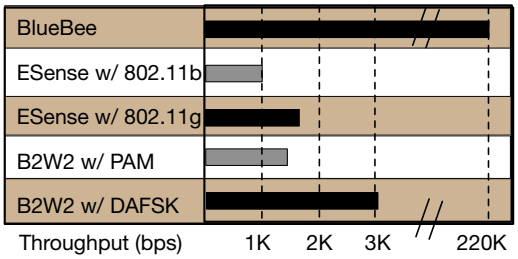
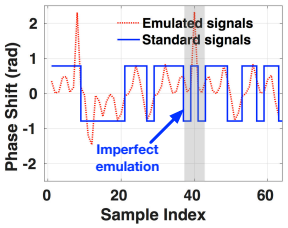


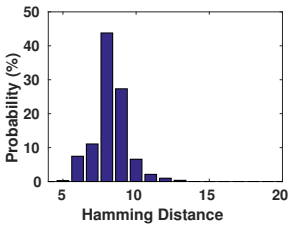
Figure 16: Comparison with the state of the arts

8.3 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.



(a) Phase shift of emulated and standard symbols



(b) Hamming distance of all emulated symbols

Figure 17: Performance of phase shift emulation

8.3.1 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 the Section IV, ZigBee's OQPSK demodulation is based on the phase shifts, whose positive and negative sign will be further decoded as BLE symbol '1' and '0'. In Fig. 17a, we plot the phase shift of 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 is due to the over-sampling 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. We find that most Hamming distances are in the range of [6, 10] especially in [8, 9], showing that the number of error chips caused by inconsistent phase shifts is small and within the tolerance of ZigBee.

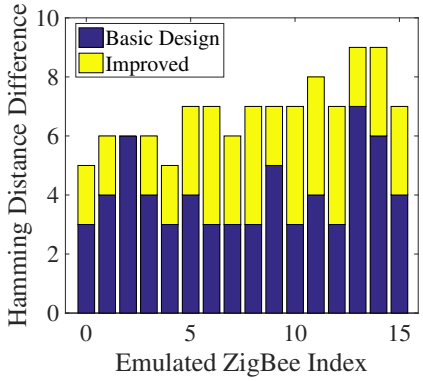


Figure 18: Hamming distance improvement of DSSS emulation

Since the ZigBee's OQPSK demodulation needs to consider the closest hamming distance, the inter-symbol hamming distance also affects the accuracy of emulation. In Fig. 18, we illustrate the performance gain when BlueBee considers the intra symbol hamming distance. For example, after the optimization, the hamming distance improvement is in Figure 18. In the basic design, the hamming distance difference ranges from 3 to 7, while the hamming distance difference of 3 suggests very little protection from the background noises. With the optimization of BlueBee, we manage to increase this hamming distance difference for the emulated ZigBee symbols, as shown in Figure 18. This means that BlueBee can tolerate more background noises than the basic design, leading to a better reliability.

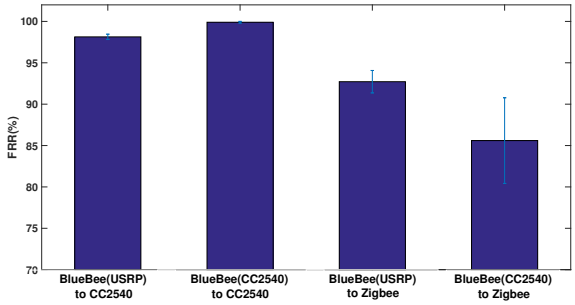


Figure 19: FRR comparison under BLE and ZigBee

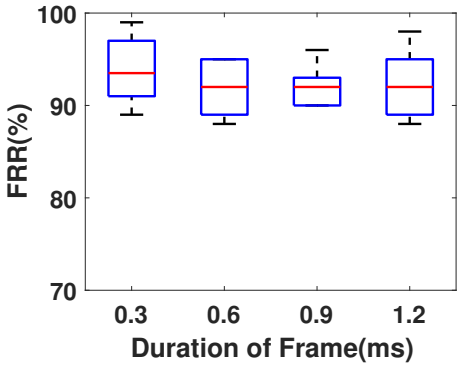
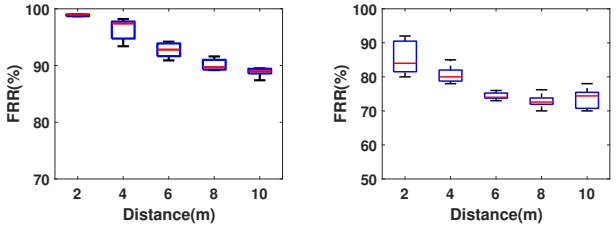


Figure 21: FRR with different frame duration

8.3.2 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 hardwares, including 1) software defined radio, i.e., USRP N210 and 2) commodity BLE devices, i.e., CC2540 development kit. At the receiver side, we use both commodity BLE receiver (i.e., CC2540) and commodity ZigBee receiver (i.e., MICAz). As shown in Fig. 19, 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 85% FRR at commodity a ZigBee receiver, showing that it is also a ZigBee compliant design.

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.



(a) FRR with distance on USRP (lab) **(b) FRR with distance on commodity devices (lab)**

Figure 20: FRR with distance

8.3.3 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 CC2540 development kit. Fig. 20a depicts the FRR when USRP N210 emulates the ZigBee frames with the transmission power of 0dBm, the maximum energy level allowed in BLE standard [34]. 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 increasing of distance, due to the lower SNR. Even so, in all the experiments, the FRRs are all above 85%. The experiments on commodity CC2540 development kit have similar trend. During these experiments, the FRR is above 73% for all the different transmission distances.

8.3.4 Impact of Frame Duration. In BLE specification 4.2 [34], 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. 21, 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 decreases FRR, about 2% decrease. That is because a longer frame is usually more vulnerable to environment noise and interference [33]. Even so, over 90% FRR shows BlueBee's resistance to the impact of long frame.

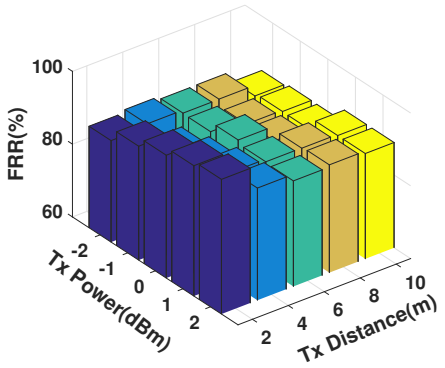


Figure 22: FRR with Tx power and distances

8.3.5 Impact of Tx Power and Tx Distance. In Fig. 22 we study the frame reception ratio (FRR) of BlueBee with 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 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.

8.3.6 Protection in the link layer-multiple header. In Fig. 23 we study the performance of our link layer protection by repeated preambles. 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 doubles the length of 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. 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.

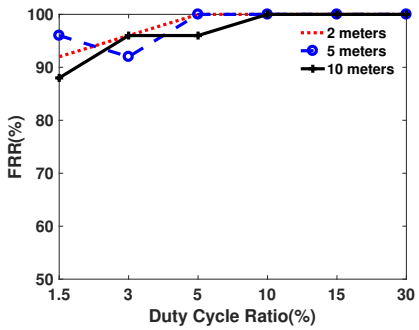


Figure 25: BluBee's support for low duty cycle network

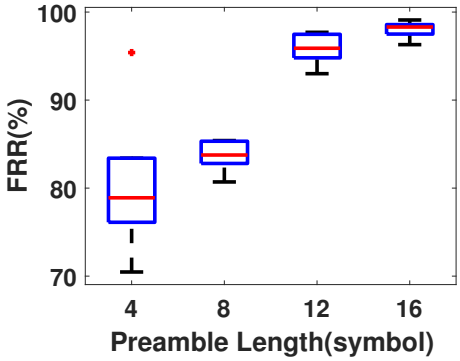


Figure 23: FRR with different preamble length out door

8.4 BlueBee Channel Scheduling

In this section, we evaluate the performance of the BlueBee scheduling algorithm to evenly distribute the BlueBee emulation frames. Three TelosB nodes are set to ZigBee channel 22, 24, and 26, which have the same central frequencies with BLE channel 27, 32, and 39 respectively. The BLE sender is implemented on the USRP N210-platform, with a total number of 999 emulation frames. To test the performance, BlueBee adopts out traffic adaptive algorithm to evenly distribute the CTC traffic among ZigBee channels, i.e. 333 frames at each ZigBee channel (i.e., 33% of all packets).

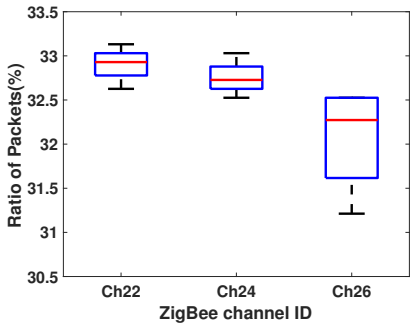


Figure 24: Concurrent CTC on three ZigBee channels

Fig. 24 depicts 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 efficiency of the traffic adaption method based on the existing BLE channel bitmap. Note that the ratio of received packet will be slightly lower in ZigBee channel 26 because it overlaps with BLE advertising channel 39, which is very busy.

8.5 Low Duty Cycle Support

In this section, we study the BLE's support to the low duty cycle network due to the fact that ZigBee devices are usually work on low duty cycle mode to save energy. Low duty cycle scenario becomes even critical due to the fact that the BLE transmitter will do frequency hopping. In the experiment, we transmit BLE frame from USRP to MICAz, a commodity ZigBee device. The BLE transmitter's hopping interval is set to be $10ms$, within the range of available hopping interval in the standard. From Sec. 5.1 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 frames 20 times. As shown in Fig. 25, 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%, a FRR of at least 88% still can be reached. This experiment indicate that BlueBee has the potential to be used in a low duty cycle as a long lasting coordinator.

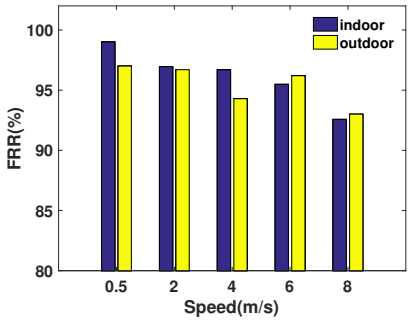


Figure 26: BluBee's support for mobile scenario

8.6 Mobility Support

In this part, we study the impact of mobility on the performance of BlueBee because BLE radios are widely used in mobile scenario such as in smart wristbands. 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., MICAz node, is walking, jogging, and running with different speed at about $10m$ away. As shown in Fig. 26, there is only a slightly 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.

8.7 Application

Application I: Channel Coordination In this section, we demonstrate one possible application built upon BlueBee, i.e. the channel coordination between incompatible ZigBee and BLE. Note that BlueBee enables many possible benefits as stated in Section II, and

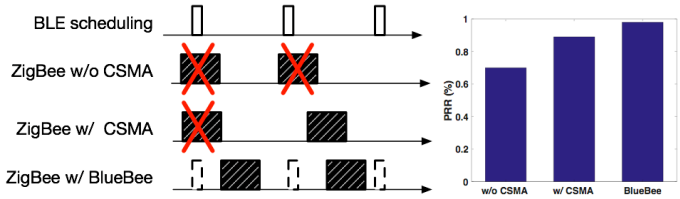


Figure 27: Channel coordination between ZigBee and BLE

we only introduce its channel coordination due to the limitation of space as shown in the left part of Fig. 27.

In this experiment, the two ZigBee TelosB devices are communicating on ZigBee channel 26, to avoid other possible ISM band interference. One BLE sender is transmitting its advertising frames on frameBLE channel 39, which overlaps with the ZigBee channel 26. Since BLE does not perform CSMA before transmission, the BLE frames might corrupt the ZigBee frames when they collide into each other.

To evaluate the performance, we conduct experiments on different coordination methods, such as no CSMA, with CSMA, and our channel scheduling method. In our channel scheduling, when the BLE wants to transmit the BLE frames, it first broadcasts the scheduling frame using BlueBee, which contains the future channel usage of BLE. After successfully receiving these frames, the ZigBee transmitter will coordinate the timing of the transmitted frames accordingly.

In the right part of Fig. 27 shows the experimental results. Compared with no CSMA, and with CSMA, BlueBee successfully improves the frame reception ratio to 98%, clearly demonstrating the channel efficiency of BlueBee's coordination. This implies that effective radio coordination can be achieved through CTC, which opens a door for cross-technology MAC design in the future.

Figure 28: BlueBee smart light bulb control

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. 28, 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 [16] and any hardware modification at either side. BlueBee can be easily generalized to other IoT control scenario. It is a key enabler for other IoT cross-technology control design under commodity ZigBee and BLE devices.

9 RELATED WORK

With the rapid development of various wireless technologies, the ISM band suffers from significant cross-technology interference (CTI) [4, 5, 8–10, 15, 20, 25, 27, 30]. To alleviate this this, there has been numerous researches on alleviating the CTI by detecting and avoiding the interference, or recovering the corrupted signal from the interference[31–33, 36, 38, 39, 41, 42, 44–46]. However, this line of methods force the receivers to adapt to the interference pattern, resulting in the unfairness between various technologies.

To address this issue, researchers propose cross-technology communication (CTC) which directly builds the communication between heterogeneous devices [7, 11, 22, 42, 43]. 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. However, the existing CTC technologies commonly use coarse packet-level information, thus suffer from the significant low throughput and long transmission delay.

In contrast to these packet-level CTC methods, BlueBee is the first approach to achieve the PHY-level CTC to the best of our knowledge. The core idea of signal emulation in the BlueBee is inspired by several recent works studying the signal manipulation[6, 19, 21, 23, 28]. In addition, in the LTE system, Ultran [6] emulates the WiFi packets via a LTE transmitter to coordinate between LTE and WiFi, while it requires the modification of existing LTE standard. Different from these approaches, BlueBee does not require andy modification to existing hardware, and is fully compatible with existing commodify Bluetooth and ZigBee hardware.

In summary, BlueBee is the first PHY-level CTC which does not require any hardware modification. It is fully compatible with existing Bluetooth and ZigBee hardware, and achieves high CTC throughput with little transmission delay.

10 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× 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 in BLE operation. 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.

ACKNOWLEDGMENTS

This work was supported in part by the NSF CNS-1444021, NSF CNS-1718456, NSF CNS-1717059, and NSF China 61672196. We sincerely thank our shepherd Abusayeed Saifullah and anonymous reviewers for their valuable comments and feedback.

REFERENCES

[1] 1999. IEEE 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification* (1999).
[2] 2013. TI cc2540 development kit. http://www.ti.com/tool/cc2540dk/. (2013).
[3] 2016. Scapy radio. https://github.com/BastilleResearch/scapy-radio. (2016).
[4] Fadel Adib, Swarun Kumar, Omid Aryan, Shyamnath Gollakota, and Dina Katabi. 2013. Interference alignment by motion. In *MobiCom '13*. ACM, 279–290.
[5] Paramvir Bahl, Ranveer Chandra, Thomas Moscibroda, Rohan Murty, and Matt Welsh. 2009. White space networking with wi-fi like connectivity. *ACM SIGCOMM Computer Communication Review* 39, 4 (2009), 27–38.
[6] Eugene Chai, Karthik Sundaresan, Mohammad A Khojastepour, and Sampath Rangarajan. 2016. LTE in unlicensed spectrum: are we there yet?. In *MobiCom '16*. ACM, 135–148.
[7] Kameswari Chebrolu and Ashutosh Dhekne. 2009. Esense: communication through energy sensing. In *Proceedings of the 15th annual international conference on Mobile computing and networking*. ACM, 85–96.
[8] Bo Chen, Yue Qiao, Ouyang Zhang, and Kannan Srinivasan. 2015. Airexpress: Enabling seamless in-band wireless multi-hop transmission. In *MobiCom '15*. ACM, 566–577.
[9] Bo Chen, Vivek Yenamandra, and Kannan Srinivasan. 2015. Interference alignment using shadow channel. In *INFOCOM 2015*. IEEE, 2128–2136.
[10] Lin Chen, Ruolin Fan, Kaigui Bian, Mario Gerla, Tao Wang, and Xiaoming Li. 2015. On heterogeneous neighbor discovery in wireless sensor networks. In *INFOCOM '15*. IEEE, 693–701.
[11] Zicheng Chi, Yan Li, Hongyu Sun, Yao Yao, Zheng Lu, and Ting Zhu. 2016. B2W2: N-Way Concurrent Communication for IoT Devices. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 245–258.
[12] Inc Gartner. 2016. *Gartner Report*. Available at urlhttp://cloudtimes.org/2013/12/20/gartner-theinternet-of-things-will-grow-30-times-to-26-billion-by-2020/.
[13] Minkeun Ha, Seong Hoon Kim, Hyungseok Kim, Kiwoong Kwon, Nam Giang, and Daeyoung Kim. 2012. SNAIL gateway: Dual-mode wireless access points for WiFi and IP-based wireless sensor networks in the internet of things. In *2012 IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, January 14-17, 2012*.
[14] Minkeun Ha, Kiwoong Kwon, Daeyoung Kim, and Peng-Yong Kong. 2014. Dynamic and Distributed Load Balancing Scheme in Multi-gateway Based 6LoWPAN. In *2014 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, and IEEE Cyber, Physical and Social Computing, iThings/GreenCom/CPSCom 2014, Taipei, Taiwan, September 1-3, 2014*.
[15] Tian Hao, Ruogu Zhou, Guoliang Xing, Matt W Mutka, and Jiming Chen. 2014. Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks. *IEEE Transactions on mobile computing* 13, 6 (2014), 1379–1392.
[16] Mayur Hawelikar and Sunil Tamhankar. 2015. A design of Linux based ZigBee and Bluetooth low energy wireless gateway for remote parameter monitoring. In *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*. IEEE, 1–4.
[17] Jun Huang, Guoliang Xing, Gang Zhou, and Ruogu Zhou. 2010. Beyond co-existence: Exploiting WiFi white space for ZigBee performance assurance. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*. 305–314.
[18] Ieee802.org. 2012. IEEE 802.15.4. (2012).
[19] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. 2016. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 356–369.
[20] Tao Jin, Guevara Noubir, and Bo Sheng. 2011. WiZi-Cloud: Application-transparent dual ZigBee-WiFi radios for low power internet access. In *INFOCOM*. 1593–1601.
[21] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. 2016. Passive wi-fi: Bringing low power to wi-fi transmissions. In *NSDI '16*. USENIX Association, 151–164.
[22] Song Min Kim and Tian He. 2015. FreeBee: Crosstechnology Communication via Free Sidechannel. In *MOBICOM, 2013 Proceedings ACM*.
[23] Zhenjiang Li, Yaxiong Xie, Mo Li, and Kyle Jamieson. 2015. Recitation: Rehearsing wireless packet reception in software. In *MobiCom '15*. ACM, 291–303.
[24] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. 2010. Surviving Wi-fi Interference in Low Power ZigBee Networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*.
[25] Rajesh Mahindra, Hari Viswanathan, Karthik Sundaresan, Mustafa Y Arslan, and Sampath Rangarajan. 2014. A practical traffic management system for integrated LTE-WiFi networks. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 189–200.
[26] Stefan Nastic, Hong Linh Truong, and Schahram Dustdar. 2015. SDG-Pro: a programming framework for software-defined IoT cloud gateways. *J. Internet Services and Applications* 6, 1 (2015), 21:1–21:17.
[27] Georgios Nikolaidis, Mark Handley, Kyle Jamieson, and Brad Karp. 2015. COPA: cooperative power allocation for interfering wireless networks. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 18.
[28] Jiajue Ou, Yuanqing Zheng, and Mo Li. 2014. MISC: Merging incorrect symbols using constellation diversity for 802.11 retransmission. In *INFOCOM 2014*. IEEE, 2472–2480.
[29] Soheil Qanbari, Negar Behinaein, Rabee Rahimzadeh, and Schahram Dustdar. 2015. Gatica: Linked Sensed Data Enrichment and Analytics Middleware for IoT Gateways. In *3rd International Conference on Future Internet of Things and Cloud, FiCloud 2015, Rome, Italy, August 24-26, 2015*. 38–43.
[30] Saravana Rathinakumar, Bozidar Radunovic, and Mahesh K Marina. 2016. CPRecycle: Recycling Cyclic Prefix for Versatile Interference Mitigation in OFDM based Wireless Systems. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 67–81.
[31] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Ranveer Chandra, and Jie Liu. 2016. SNOW: Sensor network over white spaces. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*.
[32] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. 2011. No time to countdown: Migrating backoff to the frequency domain. In *MobiCom '11*. ACM, 241–252.
[33] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. 2013. Successive interference cancellation: Carving out MAC layer opportunities. *IEEE Transactions on Mobile Computing* 12, 2 (2013), 346–357.
[34] Bluetooth specification. 2011. Bluetooth technology website. (2011). http://www.bluetooth.com/
[35] Dominic Spill and Andrea Bittau. 2007. BlueSniff: Eve Meets Alice and Bluetooth. *WOOT* 7 (2007), 1–10.
[36] Karthikeyan Sundaresan, Srikanth V Krishnamurthy, Xinyu Zhang, Amir Khojastepour, Sampath Rangarajan, et al. 2015. TRINITY: A Practical Transmitter Cooperation Framework to Handle Heterogeneous User Profiles in Wireless Networks. In *MobiHoc '15*. ACM, 297–306.
[37] Zhimeng Yin, Wenchao Jiang, Song Min Kim, and Tian He. [n. d.]. C-Morse: Cross-technology Communication with Transparent Morse Coding. In *Proceedings of INFOCOM 2017*.
[38] Sangki Yun and Lili Qiu. 2015. Supporting WiFi and LTE co-existence. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 810–818.
[39] Xinyu Zhang and Kang G Shin. 2011. Enabling coexistence of heterogeneous wireless systems: Case for ZigBee and WiFi. In *MobiHoc '11*. ACM, 6.
[40] Xinyu Zhang and Kang G. Shin. 2013. Cooperative Carrier Signaling: Harmonizing Coexisting WPAN and WLAN Devices. *IEEE/ACM Trans. Netw.* 21, 2 (April 2013).
[41] Xinyu Zhang and Kang G Shin. 2013. Cooperative carrier signaling: Harmonizing coexisting WPAN and WLAN devices. *Networking, IEEE/ACM Transactions on* 21, 2 (2013), 426–439.
[42] Xinyu Zhang and Kang G Shin. 2013. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 3094–3101.
[43] Yifan Zhang and Qun Li. 2013. HoWiES: A holistic approach to ZigBee assisted WiFi energy savings in mobile devices. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 1366–1374.
[44] Ruogu Zhou, Yongping Xiong, Guoliang Xing, Limin Sun, and Jian Ma. 2010. ZiFi: wireless LAN disc overy via ZigBee interference signatures. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 49–60.
[45] Wenjie Zhou, Tarun Bansal, Prasun Sinha, and Kannan Srinivasan. 2014. Bbn: throughput scaling in dense enterprise wlans with bind beamforming and nulling. In *MobiCom '14*. ACM, 165–176.
[46] Wenjie Zhou, Tanmoy Das, Lu Chen, Kannan Srinivasan, and Prasun Sinha. 2016. BASIC: backbone-assisted successive interference cancellation. In *MobiCom '16*. ACM, 149–161.