Taming the Errors in Cross-Technology Communication: A Probabilistic Approach

XIUZHEN GUO, YUAN HE, JIA ZHANG, HAOTIAN JIANG, ZIHAO YU, and XIN NA,
Tsinghua University, China

Cross-Technology Communication (CTC) emerges as a technology to enable direct communication across different wireless technologies. The state of the art on CTC employs physical-level emulation. Due to the protocol incompatibility and the hardware restriction, there are intrinsic emulation errors between the emulated signals and the legitimate signals. Unresolved emulation errors hurt the reliability of CTC and the achievable throughput, but how to improve the reliability of CTC remains a challenging problem. Taking the CTC from WiFi to BLE as an example, this work first presents a comprehensive understanding of the emulation errors. We then propose WEB, a practical CTC approach that can be implemented with commercial devices. The core design of WEB is split encoding: based on the probabilistic distribution of emulation errors, the WiFi sender manipulates its payload to maximize the successful decoding rate at the BLE receiver. We implement WEB and evaluate its performance with extensive experiments. Compared to two existing approaches, WEBee and WIDE, WEB reduces the SER (Symbol Error Rate) by 54.6% and 42.2%, respectively. For the first time in the community, WEB achieves practically effective CTC from WiFi to BLE, with an average throughput of 522.2 Kbps.

CCS Concepts: • **Networks** → **Network protocol design**; **Sensor networks**; **Cross-layer protocols**; • **Computer systems organization** → **Embedded systems;**

Additional Key Words and Phrases: Cross-technology, WiFi to BLE, split encoding

ACM Reference format:
Xiuzhen Guo, Yuan He, Jia Zhang, Haotian Jiang, Zihao Yu, and Xin Na. 2021. Taming the Errors in Cross-Technology Communication: A Probabilistic Approach. *ACM Trans. Sen. Netw.* 18, 1, Article 3 (October 2021), 20 pages.
https://doi.org/10.1145/3469031

1 INTRODUCTION

The proliferation of **Internet of Things (IoT)** applications calls for ubiquitous connectivity among heterogeneous wireless devices [2, 15, 29]. To this end, **cross-technology communication (CTC)** has emerged to enable direct communication among devices that follow different

This work is supported in part by National Key R&D Program of China No. 2017YFB1003000, National Science Fund of China under grant No. 61772306, and the R&D Project of Key Core Technology and Generic Technology in Shanxi Province (2020XXX007).
Authors' address: X. Guo, Y. He (corresponding author), J. Zhang, H. Jiang, Z. Yu, and X. Na, School of Software, Tsinghua University, 30 Shuangqing Rd, Haidian, Beijing, China 100084; emails: guoxiuzhen94@gmail.com, heyuan@mail.tsinghua.edu.cn, {j-zhang19, jht19, zh-yu17, nx20}@mails.tsinghua.edu.cn.

1550-4859/2021/10-ART3 $15.00
https://doi.org/10.1145/3469031

communication standards [3, 17, 33]. CTC not only provides a new way to manage wireless networks [6, 22, 31, 34, 36] but also enhances the ability of interoperation and data exchange among heterogeneous devices [10, 14, 21, 27, 35].

In many attractive applications, CTC has great significance. Figure 1 shows several example scenarios: (1) in mobile health, the wearable BLE (Bluetooth Low Energy) sensors can directly communicate with the WiFi devices to realize real-time data sharing [5, 30]; (2) in smart offices and stores, CTC achieves cross-technology localization where people carrying a BLE device can get navigation service and product information by the messages from WiFi landmarks [7]; (3) in smart home, the commercial WiFi APs with CTC become powerful gateways to connect and control heterogeneous wireless devices, such as speakers, watches, sensors, and cameras [4]; and (4) CTC enables more efficient channel coordination and collision management in wireless networks, and control messages can be explicitly exchanged among coexisting heterogeneous devices [34].

The state of the art focuses on the physical-level CTC. The basic idea is to directly emulate the legitimate signals of the receiver with the sender's radio [18, 23]. Working at the PHY layer, physical-level CTC usually achieves relatively high communication throughput [11, 19, 25] compared to the early proposed packet-level CTC [20, 28, 32].

In spite of the positive progress of CTC, the quality of communication is often overlooked: due to the protocol incompatibility and the hardware restriction, there are intrinsic emulation errors between the legitimate signals and the emulated signals. They are likely to cause failures in packet decoding at the receiver side. Whether errors are resolved largely determines the achievable performance of CTC.

Most of the existing works, however, merely rely on the built-in error-tolerance mechanism on a receiver to tolerate emulation errors. The decoding process of WEBee [23] maps multiple WiFi-emulated chip sequences to one ZigBee symbol. When errors in chips are below a preset threshold, the decoding process succeeds with a correct result. The actual emulation errors, however, often exceed the preset threshold and result in serious decoding errors. As reported in [23], WEBee has only around a 50% packet reception ratio for WiFi-to-ZigBee CTC, which means around 50% packets contain excessive errors. By using a more flexible emulation scheme, the ZigBee packet reception ratio of WIDE [11] is improved, which ranges from 41.7% to 86.2%. In both cases [11, 23], it generally requires multiple retransmissions to successfully deliver a packet, which means excessive communication cost.

Unfortunately, solely relying on retransmissions can't guarantee the successful delivery in many significant scenarios. In some wireless standards, e.g. BLE, the above-mentioned error-tolerance mechanism doesn't exist at all. That means that if the errors in CTC are not appropriately controlled or eliminated, almost all the emulated packets will be corrupted. How to realize reliable and efficient CTC under such contexts remains a challenging problem. Moreover, considering that many devices concerned by CTC, e.g., BLE-based wearables and ZigBee-based sensors, are energy constrained, improving the reliability of CTC has undoubted significance.

In order to address the above problem, our work in this article conducts in-depth analysis on the errors in CTC. Taking the CTC from WiFi to BLE as an example, we find that the errors in CTC mainly come from two sources: QAM errors and **Cyclic Prefix (CP)** errors. We further find that different errors have different impact on the eventual decoding result. So we build a probability model that characterizes the decoding behavior of the CTC receiver, given the existence of the above-mentioned errors. Based on this model, we propose a novel scheme called *Split Encoding*, which tackles the bandwidth asymmetry between CTC sender and receiver and probabilistically minimizes the decoding errors.

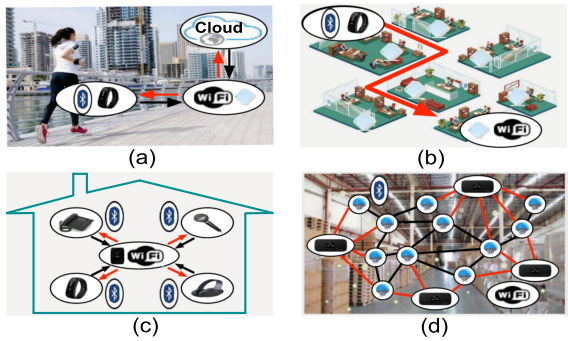


Fig. 1. IoT applications built upon the cooperation among heterogeneous wireless technologies.

Our contributions can be summarized as follows:

- We present comprehensive understanding on the errors of CTC and instantiate them in the physical emulation from WiFi to BLE. We clearly show the sources of errors and how they impact the decoding result of CTC. Meanwhile, by taking the potential errors into account, we build a probability model that characterizes the decoding behavior of the CTC receiver.
- We propose the scheme of Split Encoding, which mainly consists of three modules: phase sequence generation, phase shift adjustment, and phase sequence optimization. Split encoding tackles the bandwidth asymmetry between CTC sender and receiver and probabilistically minimizes the decoding errors.
- By employing split encoding, we implement a practically applicable CTC approach (WEB) that emulates BLE signals with a WiFi radio. Compared to the two existing approaches WEBee and WIDE, WEB reduces the Symbol Error Rate **(SER)** by 54.6% and 42.2%, respectively. For the first time in the community, WEB achieves practically effective CTC from WiFi to BLE, with an average throughput of 522.2 Kbps.

The rest of this article is organized as follows. Section 2 discusses related works. Section 3 presents the background knowledge of physical-level emulation, and Section 4 analyzes the emulation errors. We elaborate on our design and discussion in Section 5 and Section 6. Section 7 presents the evaluation results. We conclude this work in Section 8.

2 RELATED WORKS

CTC enables direct communication among heterogeneous wireless devices. Early works propose *packet-level CTC*, which manipulates packets and uses a certain metric of the packet as information carrier, e.g., the received signal strength [12, 20, 28], the packet length [37], the transmission timing [32], and the channel state information [9, 13]. Since a packet only carries very limited data, the throughput of packet-level CTC is limited to tens of bps.

In recent years, *physical-level CTC* was proposed and became the mainstream technique. Physical-level CTC aims at creating compliance between wireless technologies and building the CTC channel at the PHY layer. WEBee [23] proposes physical-level emulation. It sets the payload of a WiFi frame so that a portion of this WiFi frame is recognized as a legitimate ZigBee frame by the receiver. Using a similar method, BlueBee [25] modifies the payload of BLE to emulate the signal of ZigBee. XBee [19] proposes CTC from ZigBee to BLE, which decodes a ZigBee packet by observing the bit patterns perceived at the receiver. Decoding the ZigBee signal is realized by the cross-demapping process based on a pre-configured mapping table.

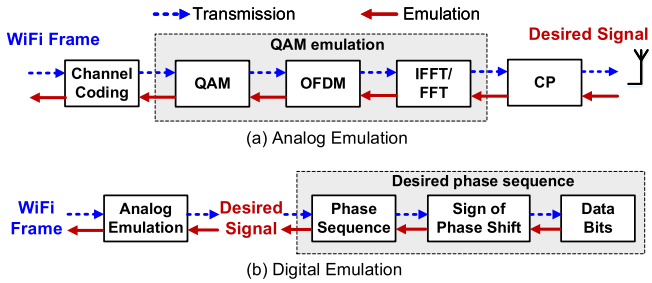


Fig. 2. The typical workflow of physical-level CTC.

The reliability and efficiency of existing physical-level CTC approaches are far from being satisfactory due to the intrinsic emulation errors between the legitimate signals and the emulated signals. WEBee achieves only 50% ZigBee packet reception for WiFi-to-ZigBee CTC reported in [23]. If the high reliability has to be provided, the emulated packets have to be sufficiently retransmitted. Both the efficiency and throughput of physical-level CTC will degrade. Recent works focus more on the reliability and applicability of CTC. WIDE [11] proposes the method of digital emulation. Instead of emulating the legitimate phase with the WiFi QAM points, WIDE modifies the payload of WiFi to modulate the phase shift of ZigBee signals, which enhances the communication reliability. By using the chip-combining coding scheme and the concentration of transmission power, TwinBee [8] and LongBee [24] also improve the CTC reliability.

Most of the existing works overlook the problem of emulation errors, which is the root cause of the failure of emulated packet reception. As we will analyze in Section 4, the applicability of CTC is questionable for some wireless standards, e.g., BLE, which sets a much higher bar for emulation accuracy. In light of the existing works in the community, our work in this article can be positioned at the pivot point across two dimensions: on one hand, our study brings to light the problem of emulation errors and tackles it with split encoding; on the other hand, our work contributes to the missing piece in the state of the art, namely the CTC from WiFi to BLE.

3 PRELIMINARIES

3.1 Physical-level Emulation

Analog Emulation refers to the CTC approach that emulates the standard time-domain signals of a receiver. Taking WEBee [23] as an example, the emulation process is shown in Figure 2(a). The desired signals at the ZigBee receiver are the standard time-domain signals. The WiFi sender feeds those signals into the FFT module to select the nearest constellation points. Those selected constellation points are modulated into different subcarriers using OFDM. The **Inverse Fast Fourier Transform (IFFT)** module then transforms those subcarriers into time-domain signals. The WiFi sender further adds the CP to the time-domain signals before transmitting them. Since the emulation is transparent to the receiver, the ZigBee receiver can directly decode the received signals.

Differing from the analog emulation, **Digital Emulation** is tailored to the scenarios where the receiver (e.g., ZigBee) uses phase shift rather than the phase itself to decode signals. Instead of emulating the original time-domain waveform, the sender emulates the phase shifts associated with the desired signals [11]. The process of digital emulation is shown in Figure 2(b). Given the desired data bits of the receiver, the sender calculates the signs of phase shifts. The bit "1" and "0" correspond to the phase shift "+" and "-," respectively. The sender generates a ladder-shaped phase sequence, which matches the signs of phase shifts. The duration of each phase value is equal to the decoding period of the receiver. The ladder-shaped phase sequence corresponds to a waveform,

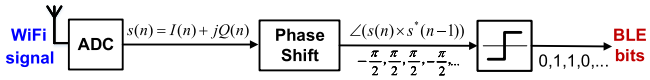


Fig. 3. The workflow of BLE receiver.

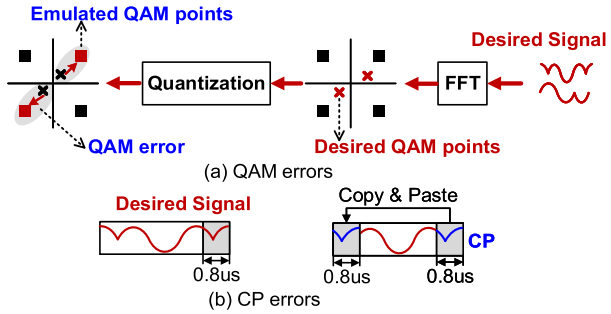


Fig. 4. Emulation errors.

which is then emulated by using analog emulation. Clearly there are multiple different phase sequences that correspond to the same signs of phase shifts. Digital emulation has better flexibility in setting the phase values in the emulated signals, which is effective in reducing emulation errors and enhancing the reliability of CTC.

3.2 Emulation of BLE Signals

We instantiate the above method of physical-level emulation to the CTC from WiFi to BLE.

First, we find that a desired frequency shift can be emulated from a phase shift. BLE adopts GFSK modulation, where each BLE bit indicates either a positive or a negative frequency shift. Note that phase is the integral of frequency:

$$s(t) = Acos(2\pi(f + \Delta f)t) = Acos(2\pi f t + \varphi(t)). \tag{1}$$

The demodulation of BLE is realized by continuously tracing the phase shifts in the received signals. As shown in Figure 3, on the BLE receiver, the phase shift between two consecutive samples $s(n)$ and $s(n-1)$ is calculated by $arctan(s(n) \times s^*(n-1))$, where $s^*(n-1)$ denotes the conjugate of $s(n-1)$. In this way, the phase shifts are quantized to BLE bits "0" or "1," according to the sign ("-" or "+") of the phase shifts. This analysis reveals the feasibility of emulating BLE signals by using WiFi signals.

4 EMULATION ERRORS

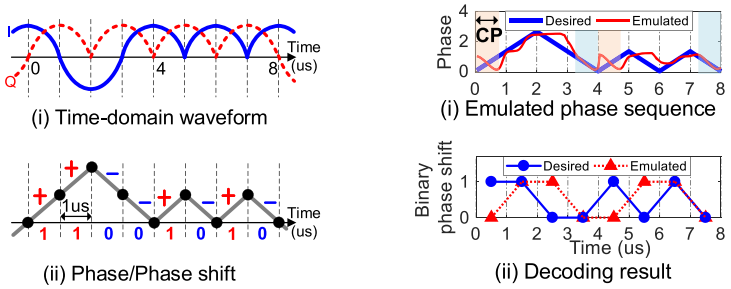
4.1 Source of Errors

In spite of the progress in physical-level emulation, an important and critical fact is often overlooked: the emulated signals from the sender can't perfectly match the desired signals of the receiver, due to the difference in communication standards and the hardware restrictions. There is more or less distance between the emulated and the desired signals, incurring emulation errors. The errors mainly come from two sources:

QAM errors occur in the process of QAM emulation, as shown in Figure 4(a). The desired signals of the receiver are fed into the FFT module to find the corresponding QAM constellation points. Note that WiFi OFDM typically has a fixed number of QAM points (16, 64, 256, etc.), which

Chips	Symbol
11011001110000110101001000101110	0000
.....	.....
10011100001101010010001011101101	1110
11001001011000000111011110111000	1111

Fig. 5. Chip-to-symbol mapping table in ZigBee.



(a) Standard waveform and phase sequence

(b) Emulated standard BLE phase sequence

Fig. 6. Standard and emulated BLE sequence.

are discretely located in the I-Q domain. Those QAM positions can't perfectly match the frequency components of the desired signals, as is the origin of QAM errors.

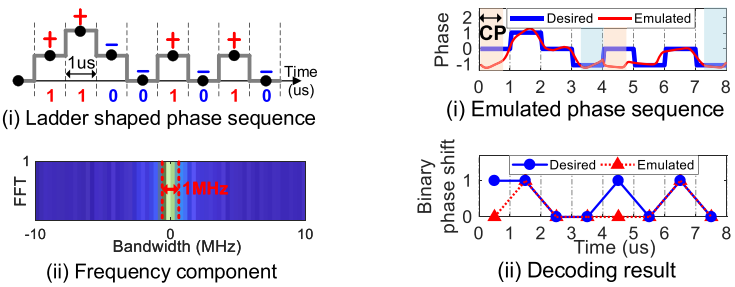
CP errors are another source of emulation errors. CP errors are induced by the WiFi CP. As shown in Figure 4(b), CP is a 0.8μs guard interval in each WiFi symbol, which is copied from the end of a WiFi symbol to the front of this symbol. That is to say, the first and the last 0.8μs segments of a WiFi symbol are always kept the same. CP is a unique feature enforced by WiFi modulation. Other wireless technologies, e.g., BLE and ZigBee, don't have such a cyclic mechanism in encoding. Due to the existence of CP, a WiFi symbol can't be arbitrarily manipulated in the emulation process, which potentially introduces emulation errors.

It is worth noticing that QAM errors and CP errors intrinsically exist with all physical-level emulation techniques. The ZigBee receiver employs the **Direct Sequence Spread Spectrum (DSSS)** [26] for demodulation, as shown in Figure 5. In this way, the demodulation of ZigBee has certain resiliency to tolerate emulation errors.

4.2 The Case with WiFi Emulated BLE

Since BLE doesn't have a DSSS-like mapping mechanism, any error in the emulated signal leads to a wrong decoded bit. That means the BLE decoding sets a strict requirement of the emulation accuracy. Can the existing emulation techniques meet that requirement?

We first examine the case of analog emulation. Figure 6(a) shows an 8μs time-domain waveform of BLE and the corresponding phase sequence. Each BLE bit lasts for 1μs and the decoding period of the BLE receiver is also 1μs. When the sign of phase shift in a decoding period is positive, the BLE bit is decoded as "1." Otherwise, it is decoded as "0." Using analog emulation, the original time-domain waveform of BLE is fed into the WiFi QAM emulation model (Figure 2(a)). Accordingly, the WiFi sender chooses a specific payload to emulate the desired BLE signal. One WiFi symbol is 4μs. So we need two WiFi symbols to emulate 8 BLE bits. Note that according to the rule of WiFi CP, for each 4μs WiFi symbol, the first 0.8μs is identical with the last 0.8μs. The emulation result is shown in Figure 6(b). Due to QAM errors and CP errors, the emulated phase sequence can't match



(a) Ladder shaped phase sequence

(b) Emulated ladder shaped phase sequence

Fig. 7. Ladder-shaped and emulated BLE sequence.

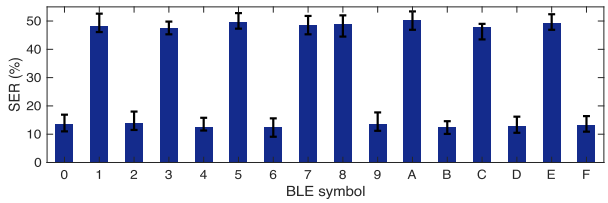


Fig. 8. Symbol error rate of different emulated BLE symbols without split encoding.

the desired phase sequence as shown in Figure 6(b)(i). There are four wrong decoded bits as shown in Figure 6(b). The **bit error rate (BER)** after emulation is 50%. An emulated packet with such a high BER is very likely to be discarded.

We then examine the case of digital emulation. Using the method introduced in [11], digital emulation generates a ladder-shaped phase sequence, which satisfies the phase shift requirement of BLE bits "11001010," as shown in Figure 7(a)(i). The corresponding emulation result is shown in Figure 7(a)(i). Compared with Figure 6(b)(i), we find that the emulated phase sequence is more similar with the desired phase sequence. The decoding result of the BLE receiver is shown in Figure 7(b)(ii). There are two wrong decoded bits. The BER is 25%. Although it is lower than that of analog emulation, it is still unacceptable for successful packet reception, due to the strict decoding mechanism of BLE.

We conduct experiments to quantify the emulation errors and their effects on CTC performance. We evaluate the decoding accuracy of all 16 kinds of emulated BLE symbols from "0000" (0x0) to "1111" (0xF) by using the method of digital emulation (WIDE [11] in the revised paper) without split encoding. For each kind of BLE symbol, the WiFi sender transmits 10,000 corresponding emulated symbols and we repeat the experiment 10 times. We conduct the experiment in a lab; the distance between the WiFi sender and the BLE receiver is 4m. The evaluation result is shown in Figure 8. For the first kind of BLE symbol, whose first bit is the same as the fourth bit, the decoding accuracy without split encoding varies from 12.4% to 13.6%. For the second kind of BLE symbol, whose first bit is different from the fourth bit, the decoding accuracy without split encoding varies from 47.6% to 50.2%. The harmful impact of WiFi CP on the second kind of BLE symbol is more serious. The high SER caused by emulation errors also results in low PRR of WiFi-BLE CTC (close to 0). The experiment results reveal the severity of the emulation errors problem. Therefore, we must reduce emulation errors including QAM errors and CP errors to meet the requirement of BLE decoding.

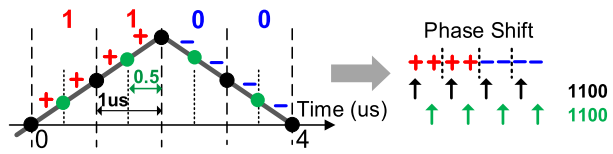


Fig. 9. The asymmetry between sampling rate and decoding rate of BLE receiver.

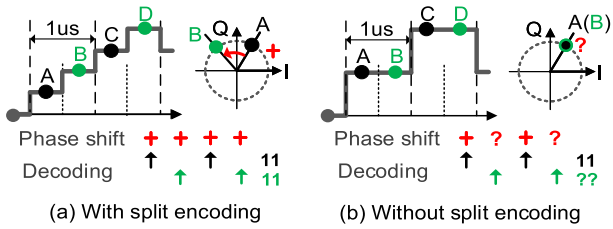


Fig. 10. Phase shift calculation with vs. without split encoding.

Based on the above observations, we realize that the existing physical-level emulation techniques have several limitations: First, the emulation errors incurred by WiFi CP aren't well resolved. Second, the generation of phase sequence, no matter in analog or digital emulation, doesn't meet BLE's strict requirement of emulation accuracy. Those two limitations lead to apparently high BER and low packet reception ratio of WiFi-to-BLE CTC. Last but not least, we find that the modulation capacity of the WiFi sender isn't fully utilized. Figure 7(a)(ii) shows the FFT result corresponding to the ladder-shaped waveform in Figure 7(a)(i). The frequency components are concentrated within 1MHz bandwidth. The bandwidth of WiFi overlapped with BLE is actually 2MHz. That means the other 1MHz of the WiFi modulation bandwidth is also usable for emulation, but not used by the existing approaches.

5 SPLIT ENCODING

5.1 Design Considerations

The primary motivation of split encoding is to leverage the bandwidth asymmetry between the sender and the receiver to control the emulation errors. Specifically in this study, the WiFi sender's encoding rate is 20MHz, within which 2MHz are overlapped with the bandwidth of the BLE receiver. Accordingly, the BLE receiver's sampling rate (the frequency to sample the perceived waveform) is also 2MHz. However, the BLE receiver's decoding rate (the frequency to decode bits from the samples) is just 1MHz. Therefore, the BLE receiver samples (and calculates) the phase shift between two consecutive samples every 0.5μs, while there are two sampled phase shifts in every 1μs. As a result, the BLE receiver will select one of every two consecutive phase shifts for decoding. So the actual decoded result is determined by the selected phase shift. Note that the timing of sampling is indeed random; the sampled points are uniformly distributed within the 1μs. Figure 9 shows the phase shift sequence of BLE bits "1100" ("+ + + + − − −−") as an example. The BLE receiver will select either the (first, third, fifth, and seventh) signs, or the (second, fourth, sixth, and eighth) signs for decoding, respectively, with 50% probability.

Suppose the encoding rate of WiFi is aligned with the decoding rate of BLE (1MHz); the phase value within a 1μs segment will be the same. We will encounter potential errors in phase shift calculation as shown in Figure 10(b). When calculating the second phase shift in each 1μs segment, the result is actually uncertain. The corresponding sign of phase shift may be "+" or "-," respectively,

with 50% probability. So we have to use "?" to denote it. The correct decoding result should be "11," but the actual decoded result may be "11" or "??". In order to minimize the probability of the uncertain errors, we propose the scheme of split encoding.

First, the decision of splitting a BLE symbol doesn't rely on the sampling rate of WiFi (sender) or BLE (receiver), but depends on the overlapping channel bandwidth of WiFi and BLE. The channel bandwidth of WiFi is 20MHz and the channel bandwidth of BLE is 2MHz. One WiFi channel is divided into 64 different subcarriers and there are 6.4 subcarriers overlapped (6 completely overlapped and 1 partially overlapped). Therefore, the asymmetry of bandwidth determines that although the sampling rate (equal to the bandwidth) of WiFi is 20MHz, WiFi can only emulate BLE signals with the subcarriers overlapped. In practice, we use seven WiFi subcarriers overlapped to emulate BLE signals.

Second, we clarify that a segment of BLE phase sequence doesn't correspond to a WiFi subcarrier. The bit rate of BLE is 1Mbps, and 4 bits make up a BLE symbol. If we split 1 bit into two phases by using split encoding, the bandwidth of a BLE symbol's phase sequence is 2MHz. If we split 1 bit into three phases by using split encoding, the bandwidth of a BLE symbol's phase sequence increases to 3MHz, which exceeds the overlapping bandwidth of WiFi and BLE. Therefore, we split 1 bit into two phases and use the WiFi subcarriers overlapped with BLE to emulate one BLE symbol.

Given the 2MHz encoding rate, indeed the WiFi sender can split every 1μs segment of signals into two halves and modulate two different phase values therein, as shown in Figure 10(a). Split encoding enables a WiFi sender to manipulate the phase sequence at doubled granularity. By appropriately setting the phase values, the WiFi sender can generate a ladder-shaped phase sequence that matches the desired phase shifts, no matter where the BLE-sampled positions are. In this way, the uncertain emulation errors will be eliminated as much as possible.

Split encoding mainly consists of three modules: phase sequence generation, phase shift adjustment, and phase sequence optimization. The WiFi sender first generates a ladder-shaped phase sequence to meet the phase shift requirement of BLE signals. Then phase shift adjustment is executed to minimize the emulation errors based on a decoding probability method. Finally, we obtain the optimal phase sequence, which is robust against potential channel distortion. In the rest of this section, we introduce these modules respectively.

5.2 Phase Sequence Generation

WEB first generates a phase sequence with split encoding to satisfy the requirement of BLE phase shift. Figure 11 illustrates phase sequence generation. For ease of illustration, we define the term *BLE symbol* denoted by $\beta = (b_0, b_1, b_2, b_3)$, which includes 4 BLE bits, and the value of $b_k (k = 0, 1, 2, 3)$ is "1" or "0." Bit "1" means the sign of phase shift is positive. Bit "0" means the sign of phase shift is negative. The sampling period and decoding period of BLE are 0.5μs and 1μs, respectively. We split every 1μs segment of signals into two halves and make the sign of phase shift be identical within each half. We denote the phase shift signs of BLE symbol β by $\epsilon = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$, which satisfies that

$$s_i = sign(b_{\lfloor \frac{i}{2} \rfloor}), (i = 0, 1, \ldots, 7). \tag{2}$$

We generate the phase sequence, which satisfies the requirement of phase shift signs. $\phi = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ denotes the phase sequence, where $x_i (i = 0, 1, \ldots, 7)$ is the phase value. Each phase value lasts for 0.5μs. The absolute phase shift between two consecutive phase values is $\delta = (\delta_0, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7)$. The initial value of $\delta_i ((i = 0, 1, \ldots, 7))$ is equal to a phase shift

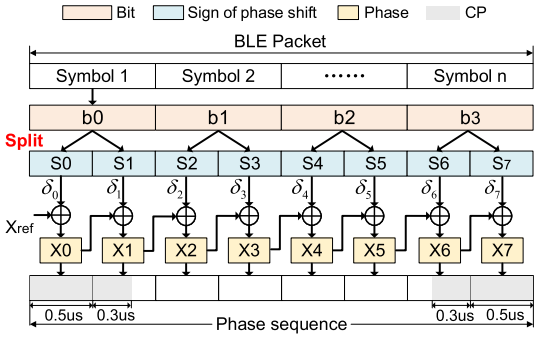


Fig. 11. The illustration of phase sequence generation.

unit Δ and $\Delta \in (0, \pi)$. The phase sequence can be generated by

$$\phi = \begin{cases} x_{-1} = x_{ref} \\ x_i = x_{i-1} + s_i \times \delta_i \quad (i = 0, 1, \ldots, 7), \end{cases} \tag{3}$$

where the x_{ref} is the reference phase, which is equal to the last phase value of the previous BLE symbol. If the BLE symbol is the first symbol of the BLE packet, x_{ref} is set to 0. In this way, we can obtain an initial phase sequence. Generally there are emulation errors between the emulated phase sequence and the desired phase sequence. We analyze the theoretical decoding probability at the BLE receiver as follows.

Decoding probability model. The decoding probability of BLE is affected by WiFi CP. However, the impact of WiFi CP with the duration of $0.8\mu s$ will not saturate the whole $1\mu s$ decoding period of BLE. So we consider the impact of sampling position during the derivation of the decoding probability model. As shown in Figure 11, phase x_7 and the last 0.3μs of phase x_6 are affected by WiFi CP. Let $P(A)$ denote the probability of the sampling position within the first 0.2μs of x_6. $P(B)$ denotes the probability of the sampling position within the last 0.3μs of x_6. $P(A) = \frac{0.2}{0.5} = 40\%$ and $P(B) = \frac{0.3}{0.5} = 60\%$. $P(W|A)$ denotes the error probability when the sampling position is within the first 0.2μs of x_6. $P(W|B)$ denotes the error probability when the sampling position is within the last 0.3μs of x_6. Then the probability of correct decoding at the BLE receiver P can be calculated by

$$P = 1 - (P(A)P(W|A) + P(B)P(W|B)). \tag{4}$$

In Equation (4), $P(A)$ and $P(B)$ are constant. In order to maximize the correct decoding probability P, we need to minimize the value of $P(W|A)$ and $P(W|B)$ by adjusting the phase shifts. We present the detailed method in the next subsection.

5.3 Phase Shift Adjustment

We aim at adjusting the phase shift value to minimize the emulation errors caused by WiFi CP and maximize the decoding probability. Due to the WiFi CP, the first $0.8\mu s$ phase sequence and the last $0.8\mu s$ phase sequence will affect the decoding of the first bit (b_0) and the fourth bit (b_3) in an emulated BLE symbol ($b_0b_1b_2b_3$). According to the sign of b_0 and b_3, we propose different phase shift adjustments for these two different cases.

Case 1: $b0 \neq b_3$. This case includes eight BLE symbols, namely "0001," "0011," "0101," "0111," "1000," "1010," "1100," and "1110." When $b0 \neq b_3$, CP errors are harmful because the phase affected by CP is opposite to the BLE phase shift requirement. According to Equation (4), decoding probability is related to the sampling position. We control the phase shift between the sample points

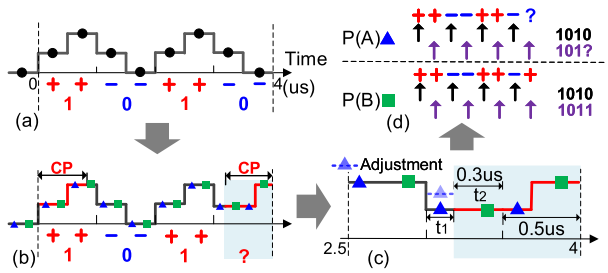


Fig. 12. Phase sequence of BLE symbol "1010."

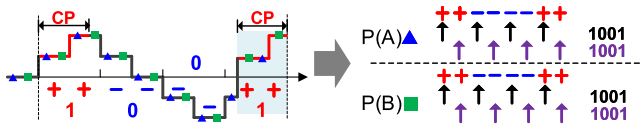


Fig. 13. Phase sequence of BLE symbol "1001."

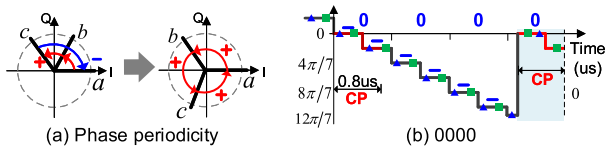


Fig. 14. Consistent phase shifts owing to phase periodicity.

affected by CP and the sample points not affected by CP to maximize the decoding probability. Specifically, we adjust the absolute phase shift value of δ_6 satisfying $\delta_6 = \frac{\Delta}{2}$. We take "1010" ("++--++--") as an example and the phase adjustment process is shown in Figure 12.

Case 2: $b0 = b_3$. This case includes eight BLE symbols, namely "0000," "0010," "0100," "0110," "1000," "1010," "1100," and "1111." In theory, when $b0 = b_3$, CP errors can be neglected because the impact of CP on the phase is consistent with the BLE phase shift requirement. For example, Figure 13 shows the phase sequence of "1001" after adding WiFi CP. According to the decoding probability analysis, we find that $P(W|A) = P(W|B) = 0$. No matter where the sampling position is, the decoding probability is 100%. Moreover, we can leverage the *phase periodicity* to remove the impact of WiFi CP as shown in Figure 14.

Summary: For the first kind of BLE symbols, whose first bit is equal to the fourth bit, CP errors can be removed and the theoretical decoding probability of the BLE symbol is 100%. For the second kind of BLE symbols, whose first bit is different from the fourth bit, CP errors can't be eliminated completely, and the theoretical decoding probability of the BLE symbol is 70%.

In fact, in our design, whether the emulated BLE packet is corrupted or not depends on the emulation errors and sampling positions. Only when the sampling point falls on the error signals will the emulated BLE packet be corrupted and discarded. The sampling position is randomly and evenly distributed. By using our method, emulation errors are tamed to specific positions and it is possible to sample the signals without emulation errors.

5.4 Phase Sequence Optimization

The goal of phase sequence optimization is to make the phase shift sequence robust against potential channel distortion. Specifically, we modify the phase sequence for a BLE symbol (b_0, b_1, b_2, b_3)

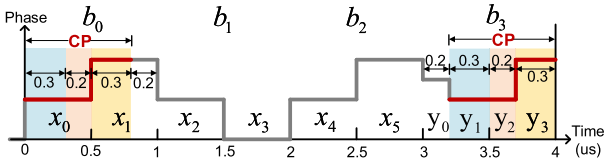


Fig. 15. Phase sequence for a BLE symbol.

as $\phi = (x_0, x_1, x_2, x_3, x_4, x_5, y_0, y_1, y_2, y_3)$, as shown in Figure 15. Considering the impact of WiFi CP, we divide the phase within $(3, 4)\mu s$ into four parts, namely $y_0 \in [3, 3.2)\mu s$, $y_1 \in [3.2, 3.5)\mu s$, $y_2 \in [3.5, 3.7)\mu s$, and $y_3 \in [3.7, 4)\mu s$. The phase shift signs of the BLE symbol are $\epsilon = (s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7)$, according to Equation (2). The absolute phase shifts between every two consecutive phase values are $\delta = (\delta_0, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7)$. The modified phase sequence ϕ can be generated by

$$\phi = \begin{cases} x_{-1} = x_{ref} & \\ x_i = x_{i-1} + s_i \times \delta_i & i = 0, \ldots, 5 \\ y_0 = x_5 + s_6 \times \delta_6 & \\ y_1 = x_0 + 2k\pi & \\ y_2 = x_0 + 2k\pi & k = 0, \pm 1. \\ y_3 = x_1 + 2k\pi & \end{cases} \tag{5}$$

Considering the channel distortion, we expect the generated phase shift between every two consecutive sampling points to be large enough. Note that the absolute phase shift $\delta_i (i = 0, 1, \ldots, 7)$ is affected by the phase shift unit Δ as analyzed in the above subsections. Therefore, we formulate an optimization problem to find the optimal phase sequence with minimal emulation errors as follows:

$$\max \ \Delta \tag{6}$$

$$s.t. \quad \begin{cases} (y_1 - x_5) \times s_6 > 0 & \\ (y_2 - y_0) \times s_7 > 0 & \\ (y_3 - y_1) \times s_7 > 0 & (when \quad b_0 = b_3). \\ \frac{\sum_{i=1}^{i=64} E[i] - \sum_{i=1}^{i=7} E[i]}{\sum_{i=1}^{i=64} E[i]} < \gamma & \end{cases}$$

The above formulation contains several conditions to be satisfied during the process of finding the optimal phase sequence of a BLE symbol.

(i) The phase values affected by WiFi CP need to satisfy the requirement of phase shift signs, as denoted by the first, second, and third constraints.

(ii) The WiFi emulation ability is limited. WiFi can only use seven subcarriers (2MHz bandwidth) overlapping with BLE for emulation, as denoted by the fourth constraint. Signal energy calculated by the FFT is used to constrain the available WiFi subcarriers. $E(i)(i = 1, 2, \ldots, 64)$ is the FFT coefficient. γ is the energy threshold, which means the energy leakage beyond seven WiFi subcarriers.

In practice, we adopt Binary Search shown in Algorithm 1 to solve the above optimization problem and find the optimal phase sequence. The search range of Δ is restricted in (η_{min}, η_{max}), where $\eta_{min} = 0$ and $\eta_{max} = \pi$. We increase the value of phase shift unit Δ until the desired bandwidth exceeds 2MHz. θ is the threshold to end the loop. γ is the threshold of acceptable energy leakage. In our evaluation, we set $\theta = \frac{\pi}{24}$ and $\gamma = 0.3$.

The process of phase optimization is offline and the obtained optimal phase sequence is emulated by WiFi [11, 23] with only 2MHz bandwidth. Moreover, our method doesn't require that the center frequency of BLE strictly aligned with WiFi. We use the WiFi subcarriers overlapped with BLE to

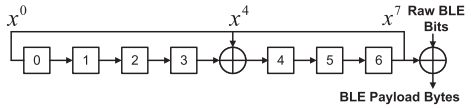


Fig. 16. Process of BLE data whitening.

ALGORITHM 1:

Input: η_{min}, the lower bound of phase shift unit. η_{max}, the upper bound of phase shift unit. θ, threshold of phase shift of ending loop. γ, threshold of acceptable energy leakage.
Output: value of phase shift unit Δ.

while $\eta_{max} - \eta_{min} > \theta$ **do**
  $\Delta = (\eta_{min} + \eta_{max})/2$
  Generate phase sequence ϕ according to Equation (5)
  $s(n) = cos(\phi) + jsin(\phi)$
  $E[i] = FFT(s(n)), i = 1, 2, \ldots, 64$
  $lek = (\sum_{i=1}^{i=64} E[i] - \sum_{i=1}^{i=7} E[i])/(\sum_{i=1}^{i=64} E[i])$
  if $lek > \gamma$ **then**
    $\eta_{max} = \Delta$
  else
    $\eta_{min} = \Delta$
  end if
end while
return Δ

emulate BLE signals. For example, WiFi channel 3 (center frequency of 2,422MHz) overlaps with BLE channel 9 (center frequency of 2,420MHz) and BLE channel 10 (center frequency of 2,422MHz). In order to achieve WiFi-to-BLE CTC, we use WiFi subcarriers [22, 23, 24, 25, 26, 27, 28] to emulate the BLE signals in channel 9 and WiFi subcarriers [29, 30, 31, 32, 33, 34, 35] to emulate the BLE signals in channel 10.

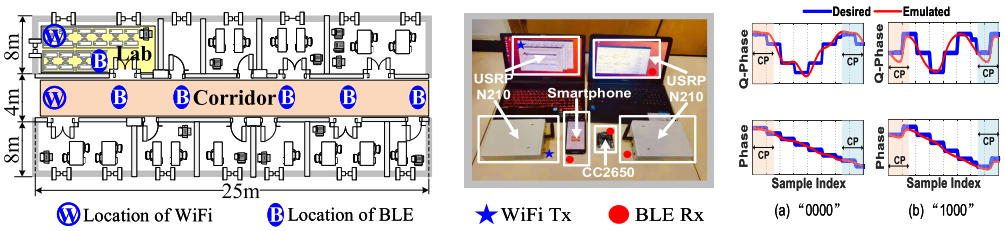
6 DISCUSSION

6.1 Reverse BLE Data Whitening

We have assumed that we can directly access the raw BLE bits from the BLE receiver, whereas we only have access to the BLE payload bytes in a commercial BLE device. There is a data whitening process between the raw BLE bits and the BLE payload bytes. The data whitening is accomplished by the 7-bit **linear feedback shift register (LFSR)** circuit with the polynomial $x7 + x4 + 1$ shown in Figure 16. We convert the BLE payload bytes into the raw BLE bits by reversing the whitening process and leveraging XOR operation. In this way, we can obtain the raw BLE bits, which make up the desired BLE symbols to be emulated by WiFi.

6.2 Working with Existing BLE Network

We design the MAC layer association so that the WiFi device can work with the existing BLE network. A WiFi device, acting as the BLE slave device, keeps broadcasting its connection availability with a specific access address at the BLE advertising channel. If the BLE master device is willing to connect to a WiFi device, it will reply with the request connection message, such as the hopping increment and hopping interval, to the WiFi slave. Because there is no physical-level CTC from BLE to WiFi, we adopt the packet-level CTC with energy modulation [16] to achieve the reverse transmission. According to the received RSSI pattern, the WiFi slave can start to transmit



(a) The floor plan of experimental scenes (b) Experiment setting (c) Emulation result

Fig. 17. Experiment setting and emulation result.

CTC symbols to the BLE device. The channel hopping strategy of WEB still follows the BLE channel hopping protocol and only blacklists the channels outside WiFi bandwidth. The WiFi device modifies the payload of different subcarriers overlapping with different BLE channels to achieve the channel hopping. Moreover, the WiFi device also supports the direct broadcasting mode on the BLE advertising channel. The CTC from WiFi to BLE will not disrupt normal BLE connection, which is scheduled in different time slots.

6.3 Overhead of Split Encoding

The process of split encoding includes two steps: phase sequence selection of BLE symbols and WiFi payload setting for emulating the BLE packet. The phase sequence selection is completed offline and it doesn't affect the efficiency of the practical running of WEB. After phase sequence selection, a mapping table from the 16 BLE symbols to the desired phase sequences can be calculated and loaded onto a WiFi device prior to running WEB. During the runtime, the WiFi sender sets the packet payload to emulate the desired phase sequence of the BLE packet. As shown in Figure 2 in the revised paper, FFT operation is the most time-consuming operation in the process of WiFi emulation, and its computational complexity is $O(nlgn)$, where n is the length of the BLE packet. For a BLE packet with length of 38 BLE symbols, it costs the WiFi sender in our implementation (an USRP N210 device with gnuradio running on Ubuntu14.4) only about 0.05ms to complete the emulation.

6.4 The Generality of the Proposed Split Encoding

Our method is suitable for the physical-level CTC in which the transmitter has CP errors and the receiver leverages phase shifting to decode data. For example, LTE adopts the modulation of SCFDMA and with predefined **guarding intervals (GIs)** in the frame structure, which is similar with the WiFi CP. Our proposed model to solve the CP errors in Section 5.4 can be extended to support CTC from an LTE sender. Meanwhile, ZigBee and BLE both leverage phase shifting to decode data. Therefore, it is theoretically feasible to migrate our method to enable more physical-level CTCs, e.g., WiFi to ZigBee, LTE to BLE, and LTE to ZigBee.

7 EVALUATION

7.1 Experiment Setup

We implement WEB on the USRP platform and the commercial off-the-shelf device as shown in Figure 17(b). The WEB transmitter is a USRP N210 device with 802.11 g/n PHY. The WEB receivers include a USRP N210 device with 802.15.1 PHY, a commodity BLE chip CC2650, and an iPhone Xs Max running iOS version 12.4. We set the central frequency of the WiFi channel at 2,427MHz (WiFi

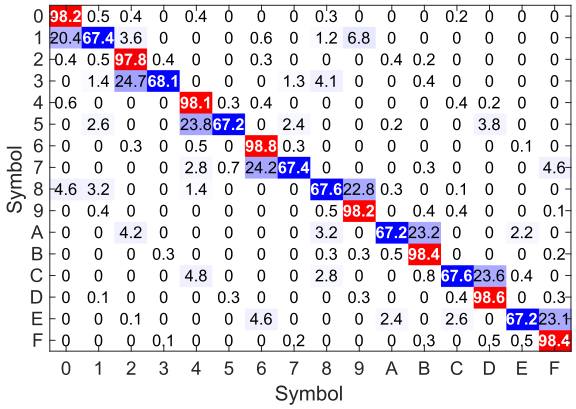


Fig. 18. Decoding accuracy of different symbols.

channel 4), which overlaps with the BLE advertising channel 38. Moreover, we also implement WEB to emulate the iBeacon [1] frame.

7.2 Effectiveness of Emulation

First, we observe the desired signal and the emulated signal to verify the feasibility of our method. We take the BLE symbols "0000" and "1000" as examples. The emulation result is shown in Figure 17(c).

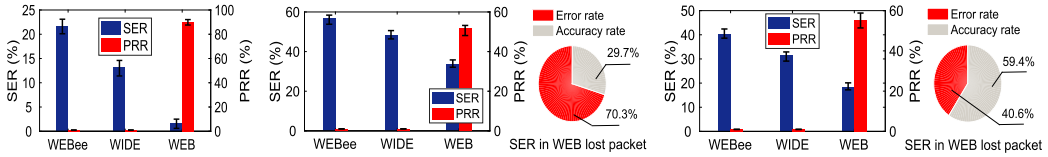
BLE has 16 kinds of symbols from "0000" (0x0) to "1111" (0xF). We further evaluate the decoding accuracy of all different BLE symbols. The evaluation result is shown in Figure 18, which can be divided into two categories. The average decoding accuracy of BLE symbols, whose first bit is the same as the fourth bit, varies from 97.8% to 98.8%. For the other BLE symbols, the average decoding accuracy varies from 67.2% to 68.1%, which is close to the theoretical 70% decoding probability.

7.3 WEB vs. Physical-level CTC

We compare the performance of WEB with WEBee [23] and WIDE [11], two advanced physical-level CTCs, to validate the emulation capability of WEB. We instantiate WEBee and WIDE to the CTC from WiFi to BLE.

We define two types of BLE packets according to the content of the BLE symbols within the packet. From the experiment results shown in the above subsection, we find that the decoding accuracy of different BLE symbols ($b_0b_1b_2b_3$) is different, which depends on the first bit (b_0) and the fourth bit (b_3) of the BLE symbol. Therefore, the reception of the emulated BLE packet is related to the content of BLE symbols within the packet. We suppose there are two types of BLE packets. *The first type of BLE packets* is made up of eight kinds of BLE symbols with equal quantity, whose first bit is the same as the fourth bit ($b_0 = b_3$). *The second type of BLE packets* is made up of the other eight kinds of BLE symbols with equal quantity, whose first bit is different from the fourth bit ($b_0 \neq b_3$). In our experiment, the WiFi sender first emulates these two types of BLE packets with the length of 40 bytes. The distance between the WiFi sender and the BLE receiver is 4m. The WiFi sender transmits 1,000 emulated BLE packets in each experiment and we repeat the experiment 10 times. We evaluate the SER and the PRR of the emulated packets.

7.3.1 Performance Comparison for the First Type of BLE Packets. The evaluation result for the first type of BLE packets is shown in Figure 19(a). The SER of BLE symbols emulated by WEBee and WIDE are 21.6% and 13.2%, respectively, which are worse than the performance reported in



(a) The first type of BLE packets (b) The second type of BLE packets (c) The legitimate BLE packets

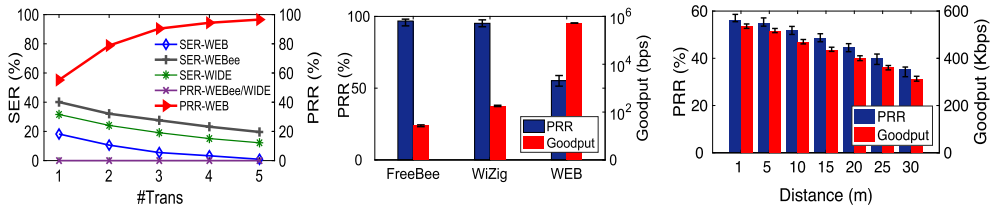
Fig. 19. Performance comparison for BLE packets.

[23] and [11]. Because there is no built-in error-tolerance mechanism like DSSS of ZigBee at the BLE receiver, the PRR of BLE packets emulated by WEBee and WIDE are close to 0, whereas the PRR of BLE packets emulated by WEB can be up to 90.2% with the SER of 1.7%.

7.3.2 Performance Comparison for the Second Type of BLE Packets. The evaluation result for the second type of BLE packets is shown in Figure 19(b). The SER of BLE symbols emulated by WEBee and WIDE increases to 56.6% and 48.4%, respectively. Because CP errors become more serious when $b_0 \neq b_3$, in this condition, the SER of BLE symbols emulated by WEB also increases to 33.8%, whereas the PRR of BLE packets emulated by WEB is 51.9%, which is owing to the split encoding method. By split encoding, emulation errors are tamed to specific positions and are no longer randomly distributed. The BLE symbol errors emulated by WEB are related to the sampling position. If the sampling position of the first BLE symbol is suitable for correct decoding, other subsequent BLE symbols in this packet can also be decoded correctly with a high probability. Otherwise, there will be consecutive symbol errors due to the incorrect sampling position, which will result in the failure of BLE packet reception. So the BLE symbol errors are not evenly distributed across all packets. We further observe the distribution of BLE symbol errors in the lost packets of WEB. As shown on the right of Figure 19(b), we find that the SER is up to 70.3% in the lost BLE packets, which is the reason for the high average SER of WEB. In this condition, WEB achieves 51.9% PRR because all the wrong BLE symbols are concentrated in the lost packet. In the following experiments, the metric we used is the average SER in all packets whether the packet is received or lost.

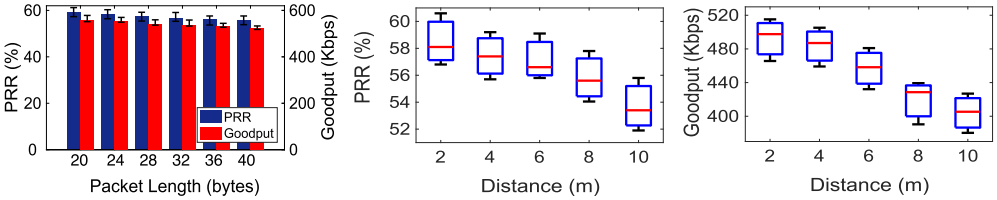
7.3.3 Performance Comparison for All Legitimate BLE Packets. In practice, a typical *legitimate BLE packet* is composed of all 16 kinds of BLE symbols. Furthermore, we use the methods of WEBee, WIDE, and WEB to emulate a legitimate BLE packet with the length of 40 bytes. Each BLE packet contains 16 kinds of symbols and the number of each symbol is the same. The evaluation result is shown in Figure 19(c). The SER of BLE symbols emulated by WEBee and WIDE is 40.1% and 31.6%, respectively, which results in that the PRR of WEBee and WIDE is close to 0. The PRR of BLE packets emulated by WEB is 55.2% and the average SER of BLE symbols is 18.2%. The wrong BLE symbols are concentrated in the lost packet as shown in the right of Figure 19(c). Specifically, the SER in the lost packet is 40.6%. In this condition, the goodput of WEB is 522.2 Kbps. From the above experiments, we find that the performance of WEB is much better than other existing physical-level CTCs. Therefore, our method can tame the emulation errors effectively and maximize the decoding rate of the BLE packet.

7.3.4 Performance Comparison after Retransmissions. We conduct experiments to evaluate the relationship of PRR and the number of packet transmissions. The evaluation result is shown in Figure 20(a). After five transmissions, the SER of BLE symbols emulated by WEBee and WIDE decreases from 40.1% to 19.6% and from 31.6% to 12.2%, respectively. We find that the effectiveness of retransmission for WEBee and WIDE is limited because of the serious and uncontrollable emulation errors in WEBee and WIDE. As a result, the PRR of BLE packets emulated by WEBee



(a) Number of transmissions (b) Comparison (c) Different distances

Fig. 20. WEB performance with different conditions.



(a) Different packet lengths (b) Corridor (c) NLoS

Fig. 21. WEB performance with different lengths and environments.

and WIDE is still 0 after limited retransmissions, whereas the retransmission strategy is effective for WEB. The PRR of WEB exceeds 96.6% and the SER decreases to 0.9% when one WEB packet is transmitted five times.

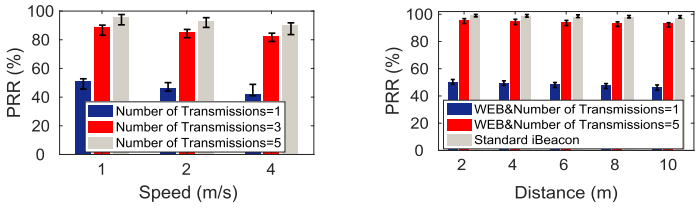
7.4 WEB vs. Packet-level CTC

From the above experiments, we find that the existing physical-level CTCs are not suitable for the CTC from WiFi to BLE. So we compare WEB with another two typical packet-level CTCs, FreeBee [32] and WiZig [16]. The evaluation result is shown in Figure 20(b). The PRR of FreeBee and WiZig is close to 1 because the RSSI in the packet level is more stable and less prone to distortion compared with emulated signal in the physical level. The goodput of FreeBee and WiZig is bounded due to the coarse granularity of packet manipulation. The goodput of WEB is 522.2 Kbps, which is a 18,000X and 2,800X improvement over FreeBee and WiZig.

7.5 WEB Performance under Different Settings

7.5.1 Impact of Distance. Figure 20(c) shows the PRR and goodput of WEB with the variation of distance. We find that the PRR and goodput decrease with the increase of distance. When the distance is 2 m, the goodput of WEB is 535.5 Kbps with the PRR of 56.2%. When the distance increases to 30 m, the goodput of WEB decreases to 312.6 Kbps with the PRR of only 35.2%. The increase of distance causes the amplitude attenuation and the phase distortion, which impacts the performance of WEB. The emulated BLE signal from the WiFi device can't achieve the comparable communication distance as the standard WiFi signal from the WiFi device and standard BLE signal from the BLE device.

7.5.2 Impact of BLE Packet Length. Figure 21(a) shows the evaluation results of PRR and goodput, respectively. The PRR and goodput decrease with the increase of BLE packet length since the longer packet brings more accumulated errors. When the packet length varies from 20 bytes to 40 bytes, the PRR of WEB decreases from 59.1% to 55.2% and the goodput of WEB decreases from 558.5 Kbps to 522.2 Kbps.



(a) Mobility (b) Emulated iBeacon

Fig. 22. WEB performance under mobility and the emulation of iBeacon.

7.5.3 Impact of Environment. **Corridor & Lab.** The PRR of WEB in the corridor is shown in Figure 21(b). We can find that the PRR of WEB in the corridor is higher than the PRR of WEB in the lab. This is because the environment in the lab is more complicated than that in the corridor, which leads to more serious multipath influence and interference on the received signals. When the distance is 10 m, the PRR of WEB in the lab (Figure 20(c)) and corridor is 51.8% and 53.4%, respectively.

LoS & NLoS. Non-Line-of-Sight (NLoS) propagation of signals affects the SINR of WEB at the BLE receiver. The goodput of WEB (Figure 20(c)) in the LoS scenario is higher than that in the NLoS scenario shown in Figure 21(c). When the distance is 10 m, the goodput of WEB in LoS and NLoS scenarios is 468.2 Kbps and 405.3 Kbps.

7.5.4 Impact of Mobility. We also evaluate the performance of WEB under mobility. A volunteer carrying the BLE receiver walks, jogs, and runs at a speed of 1 m/s, 2 m/s, and 4 m/s, respectively. In this experiment, we adopt the method of retransmission to improve the PRR of WEB packets. Figure 22(a) shows the PRR of WEB with different numbers of transmissions. When each WEB packet is transmitted 5 times, the PRR of WEB is 96.1%, 92.8%, and 90.2% for the speed of 1 m/s, 2 m/s, and 4 m/s.

7.6 Application: Proximity Service Based on WiFi Emulated iBeacon

iBeacon [1, 7] is widely implemented based on BLE to provide proximity estimation services. The iBeacon packet consists of 92 BLE symbols and the total packet length is 368 bits. Based on the method of WEB, the WiFi sender can transmit the emulated iBeacon packet to the BLE device. In this way, we can use the WiFi AP to replace the special iBeacon station, which reduces the deployment cost and makes the proximity-based services more ubiquitous.

We conduct several experiments to evaluate the performance of WiFi emulated iBeacon. The WiFi sender transmits the emulated iBeacon frame on the BLE broadcast channel. An iPhone Xs Max running iOS 12.4 is used to receive iBeacon packets. We vary the distance between the WiFi and the smartphone from 1m to 10 m. We obtain the average result of 10 experiments, each of which sends 2,000 iBeacon packets. Moreover, we also observe the performance of iBeacon reception when the transmitter is a standard iBeacon chip. The evaluation result is shown in Figure 22(b). The PRR of emulated iBeacon packet increases with the increase of the number of transmissions. When the emulated iBeacon packet is transmitted 5 times, the PRR at the smartphone increases to 95.1% if the smartphone is 4 m away from the WiFi sender. We find that the PRR of WEB after five transmissions is close to that of a standard iBeacon chip. Therefore, the WiFi emulated iBeacon packet can be received by the smartphone successfully.

8 CONCLUSIONS

This work addresses a significant problem in CTC, namely emulation errors. Taking the CTC from WiFi to BLE as an example, we tackle this problem with split encoding, which fully exploits the

encoding capacity of WiFi and modulates the phase sequence at doubled granularity. The generated phase sequence therefore meets the strict requirement of emulation accuracy with maximized probability. Compared with the existing CTC approaches, WEB shows great advantages in communication throughput and reliability.

REFERENCES

[1] Apple Inc. 2013. Apple iBeacon. https://developer.apple.com/ibeacon/.

[2] Naser AlDuaij, Alexander Van't Hof, and Jason Nieh. 2019. Heterogeneous multi-mobile computing. In *MobiSys*. ACM.

[3] Zhenlin An, Lei Yang, and Qiongzheng Lin. 2018. Cross-frequency communication: Near-field identification of UHF RFIDs with WiFi. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'18)*.

[4] Sumathi Balakrishnan, Hemalata Vasudavan, and Raja Kumar Murugesan. 2018. Smart home technologies: A preliminary review. In *Proceedings of International Conference on Information Technology (ICIT'18)*.

[5] George Boateng, Vivian Genaro Motti, Varun Mishra, John A. Batsis, Josiah Hester, and David Kotz. 2019. Experience: Design, development and evaluation of a wearable device for mHealth applications. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'19)*.

[6] Yoon Chae, Shuai Wang, and Kim Song Min. 2018. Exploiting WiFi guard band for safeguarded ZigBee. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'18)*.

[7] Dongyao Chen, G Shin Kang, Yurong Jiang, and Kyu-Han Kim. 2017. Locating and tracking BLE beacons with smartphones. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'17)*.

[8] Yongrui Chen, Zhijun Li, and Tian He. 2018. TwinBee: Reliable physical-layer cross-technology communication with symbol-level coding. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'18)*.

[9] 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 ACM Conference on Embedded Networked Sensor Systems (SenSys'16)*.

[10] Zicheng Chi, Yan Li, Yao Yao, and Ting Zhu. 2017. PMC: Parallel multi-protocol communication to heterogeneous IoT radios within a single WiFi channel. In *Proceedings of IEEE International Conference on Network Protocols (ICNP'17)*.

[11] Xiuzhen Guo, Yuan He, Jia Zhang, and Haotian Jiang. 2019. WIDE: Physical-level CTC via digital emulation. In *Proceedings of IEEE/ACM International Conference on Information Processing in Sensor Networks (IPSN'19)*.

[12] Xiuzhen Guo, Yuan He, and Xiaolong Zheng. 2020. WiZig: Cross-technology energy communication over a noisy channel. *IEEE Transactions on Networking* 28, 6 (2020), 2449–2460.

[13] Xiuzhen Guo, Yuan He, Xiaolong Zheng, Liangcheng Yu, and Omprakash Gnawali. 2020. ZigFi: Harnessing channel state information for cross-technology communication. *IEEE Transactions on Networking* 28, 1 (2020), 301–311.

[14] Xiuzhen Guo, Yuan He, Xiaolong Zheng, Zihao Yu, and Yunhao Liu. 2021. LEGOFi: Transmitter-transparent CTC with cross-demapping. *IEEE Internet of Things Journal* 8, 8 (2021), 6665–6676.

[15] Xiuzhen Guo, Longfei Shangguan, Yuan He, Jia Zhang, Haotian Jiang, Awais Ahmad Siddiqi, and Yunhao Liu. 2020. Aloba: Rethinking on-off keying modulation for ambient LoRa backscatter. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'20)*.

[16] Xiuhzen Guo, Xiaolong Zheng, and Yuan He. 2017. WiZig: Cross-technology energy communication over a noisy channel. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'17)*.

[17] Hassan Iqbal, Muhammad Hamad Alizai, Ihsan Ayyub Qazi, Olaf Landsiedel, and Zartash Afzal Uzmi. 2018. Scylla: Interleaving multiple IoT stacks on a single radio. In *Proceedings of ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'18)*.

[18] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. 2016. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM'16)*.

[19] Wenchao Jiang, Kim Song Min, Zhijun Li, and Tian He. 2018. Achieving receiver-side cross-technology communication with cross-decoding. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'18)*.

[20] Chebrolu Kameswari and Dhekne Ashutosh. 2009. Esense: Communication through energy sensing. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'09)*.

[21] Yan Li, Zicheng Chi, Xin Liu, and Ting Zhu. 2018. Chiron: Concurrent high throughput communication for IoT devices. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'18)*.

[22] Yan Li, Zicheng Chi, Xin Liu, and Ting Zhu. 2018. PassiveZigBee: Enabling ZigBee transmissions using WiFi. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'18)*.

[23] Zhijun Li and Tian He. 2017. WEBee: Physical-layer cross-technology communication via emulation. In *Proceedings of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'17)*.

[24] Zhijun Li and Tian He. 2018. LongBee: Enabling long-range cross-technology communication. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'18)*.
[25] Zhijun Li, Yan Li, Wenchao Jiang, and Tian He. 2017. BlueBee: Physical-layer cross-technology communication via emulation. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'17)*.
[26] 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 ACM Conference on Embedded Networked Sensor Systems (SenSys'20)*.
[27] Ruofeng Liu, Zhimeng Yin, Wenchao Jiang, and Tian He. 2019. LTE2B: Time-domain cross-technology emulation under LTE constraints. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'19)*.
[28] Philip Lundrigan, Neal Patwari, and Sneha K. Kasera. 2019. On-off noise power communication. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'19)*.
[29] Saeed Mirzamohammadi, Justin A. Chen, Ardalan Amiri Sani, Sharad Mehrotra, and Gene Tsudik. 2017. Ditio: Trustworthy auditing of sensor activities in mobile & IoT devices. In *Proceedings of ACM Conference on Embedded Networked Sensor Systems (SenSys'17)*.
[30] Xin Na, Xiuzhen Guo, Yuan He, and Rui Xi. 2021. Wi-attack: Cross-technology impersonation attack against iBeacon services. In *Proceedings of IEEE International Conference on Sensing, Communication and Networking (SECON'21)*.
[31] Ramanujan K. Sheshadri, Karthikeyan Sundaresan, Eugene Chai, Amir Khojastepour, Sampath Rangarajan, and Dimitrios Koutsonikolas. 2017. BLU: Blue-printing interference for robust LTE access in unlicensed spectrum. In *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT'17)*.
[32] Kim Song Min and Tian He. 2015. FreeBee: Cross-technology communication via free side-channel. In *Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom'15)*.
[33] Shuai Wang, Zhimeng Yin, Zhijun Li, and Tian He. 2018. Networking support for physical-layer cross-technology communication. In *Proceedings of IEEE International Conference on Network Protocols (ICNP'18)*.
[34] Zhimeng Yin, Zhijun Li, Kim Song Min, and Tian He. 2018. Explicit channel coordination via cross-technology communication. In *Proceedings of ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'18)*.
[35] Zihao Yu, Chengkun Jiang, Yuan He, Xiaolong Zheng, and Xiuzhen Guo. 2018. Crocs: Cross-technology clock synchronization for WiFi and ZigBee. In *Proceedings of ACM International Conference on Embedded Wireless Systems and Networks (EWSN'18)*.
[36] Zihao Yu, Pengyu Li, Carlo Alberto Boano, Yuan He, Meng Jin, Xiuzhen Guo, and Xiaolong Zheng. 2021. BiCord: Bidirectional coordination among coexisting wireless devices. In *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS'21)*.
[37] Yifan Zhang and Qun Li. 2013. HoWiES: A holistic approach to ZigBee assisted WiFi energy savings in mobile devices. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM'13)*.

Received September 2020; revised May 2021; accepted May 2021