IoT Networking: Path to Ubiquitous Connectivity

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Wenchao Jiang

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Tian He, Adviser

August, 2019

Acknowledgements

There are many people who have earned my gratitude for their contribution to my five years of Ph.D. journey in graduate school.

First of all, I would like to thank my advisor Prof. Tian He, for his guidance, support, inspiration, as well as the great confidence and the tremendous patience he has shown me. Without him, this dissertation would not be possible. I am also grateful for Prof. Zhi-Li Zhang, Prof. Stephen McCamant, and Prof. Jarvis Haupt, for serving as my dissertation committee and for their insightful suggestions and feedback. Also, I am extremely thankful to Prof. Chenyang Lu at the Washington University in St. Louis, Prof. Radu Stoleru at Texas A&M University, and Dr. Yu Gu at Visa for their support on my work and future career.

I was fortunate to be a part of MiNDs group, where I have spent a large part of my time. I sincerely appreciate the help and contributions of the former and current members including Dr. Song Min Kim, Dr. Desheng Zhang, Dr. Shuai Wang, Dr. Zhijun Li, Dr. Yongle Chen, Dr. Deming Gao, Zhimeng Yin, Ruofeng Liu, Song Liu, Shuai Wang, Ling Liu, Weiwei Chen, Sen Liu, Jin Ye, Nan Yu, and Songyuan Li. It is a pleasure to work with them and I have learned and got a lot help from them. I would also thank my collaborator Dr. Zehua Guo for the help to extend my research areas. Furthermore, I would like to thank my parents and my brother. Without their love and support in my hard time, I would not have come so far. Last but not least, I gratefully acknowledge financial support from the Department of Computer Science and Engineering at the University of Minnesota, the National Science Foundation, ACM, and IEEE.

Dedication

To my parents and brother.

Abstract

Internet of Things (IoT) is upon us with the number of IoT connected devices reaching 17.68 billion in the year 2016 and keeps an increasing rate of 17%. The popularity of IoT brings the prosperity and diversity of wireless technologies as one of its foundations. Existing wireless technologies, such as WiFi, Bluetooth, and LTE, are evolving and new technologies, such as SigFox and LoRa, are proposed to satisfy various needs under emerging application scenarios. For example, WiFi is evolving to provide higher throughput with the novel 802.11ac technology and the Bluetooth SIG has proposed the Bluetooth Low Energy (BLE) technology to support low-power applications.

However, wireless technologies are victims of their own success. The vastly increasing wireless devices compete for the limited wireless spectrum and result in the performance degradation of each device. What makes it worse is that diverse wireless devices are using heterogeneous PHY and MAC layers designs which are not compliant with each other. As a result, sophisticated wireless coordination methods working well for each homogeneous technology are not applicable in the heterogeneous wireless scenario for the failure to communicate among heterogeneous devices.

This dissertation aims at fundamentally solving the burden of communication in today's heterogeneous wireless environment. Specifically, we try to build the direct communication among heterogeneous wireless technologies, referred to as the cross-technology communication (CTC). It is counter-intuition and long believed impossible, but we find two opportunities in both the packet level and physical (PHY) layer to make the challenging mission possible.

First, wireless devices are commonly able to do energy sensing of wireless packets in the air. Energy sensing is capable to figure out packet level information, such as the packet duration and timing. Based on the energy sensing capability, we design DCTC, a CTC technology that piggyback cross-technology messages within the timing of transmitted wireless packets. Specifically, we slightly perturb the timing of packets emitted from a wireless device to form detectable energy patterns to establish CTC. Testbed evaluation has shown that we can successfully transmit information at $760bps$ while keeping the delay of each packet no longer than $0.5ms$ under any traffic pattern.

Second, in the PHY layer, high-end wireless technologies are flexible, i.e., a larger symbol set, in the modulation and demodulation. With careful choices of symbols, those wireless technologies are able to emulate and decode the PHY layer signal of a low-end one. We propose two systems BlueBee and XBee which aim at building direct communication between two heterogeneous IoT technologies, Bluetooth and ZigBee, with the idea of signal emulation and cross-decoding respectively. The former achieves signal emulation by carefully choosing the Bluetooth payload bits so that the output signal emulates a legitimate ZigBee packet which can be successfully demodulated by a commodity ZigBee devices without any changes. The latter proposes a general method to support the bidirectional communication in the PHY-layer CTC by moving the complexity to the high-end receiver for the demodulation of signal from a low-end transmitter. Our testbed evaluation has shown that our technologies successfully boost the data rate of the state of the arts by over 10,000x times, which is approaching the ZigBee standard. This result makes CTC possible to play more roles in real-time applications, such as network coordination.

In summary, this dissertation provides a new communication paradigm in a heterogeneous wireless environment, which is to provide direct communication for heterogeneous wireless devices. Such communication is built upon two opportunities: (i) wireless devices are capable to sense energy in the air so that specifically designed energy patterns can transmit cross-technology information; (ii) a high-end wireless technology is more flexible and possible to emulate and demodulate the signal from a low-end technology for communication. The technologies developed in the dissertation will be the building blocks for the future designs of efficient channel coordination and ubiquitous data exchange among heterogeneous wireless devices.

Contents

List of Figures

Chapter 1

Introduction

IoT is a foundational technology that aims at connecting every physical object in the world to the Internet to achieve ubiquitous resource access and management. It has affected all aspects of our life from home automation, transportation, health care, manufacturing, to the environment protection. The White House has announced a $160M investment in the "Smart City" initiative in 2015. As a fundamental technology in IoT, wireless technologies have grown rapidly both in the number and variety. According to a study from Statista [1], the IoT connected devices reaching 17.68 billion in the year 2016 and is increasing at a rate of 17%. New wireless technologies, such as the LoRa, NB-IoT, and SigFox are also developed to satisfy various application scenarios.

Along with the convenience brought by the boom of wireless technologies is the even crowded and heterogeneous network environment. Due to the rare spectrum resource, wireless technologies, such as WiFi, ZigBee, and Bluetooth share the common unlicensed industrial, scientific and medical (ISM) radio band. The coexistence of wireless technologies bring great challenges to the wireless connection, spectrum share, as well as data fusion, and also known as one of the main sources of wireless interference and poor link quality. Most existing wireless coordination methods that work well for the homogeneous wireless technology are not applicable or perform poorly in today's heterogeneous environment. It is still an open question how to mitigate the cross-technology interference (CTI) and better coordinate wireless technologies [2, 3, 4, 5].

1.1 Thesis Statement

The most challenging part of CTI is that heterogeneous wireless technologies are believed not able to communicate with each other due to the heterogeneous physical (PHY) layers. In other words, the signal from a heterogeneous wireless device is regarded as noise and can not be demodulated. However, in this dissertation, we break this stereotype and build direct communication between heterogeneous wireless technologies, which we refer to as the cross-technology communication (CTC). With CTC, the new paradigm of wireless communication, we not only ease the CTI problem but also pave the way to the global optimization of the wireless spectrum allocation and usage.

Technically, CTC is possible for two reasons. First, each wireless device no matter the PHY layer is commonly able to sense the wireless packet energy in the air, known as the received signal strength (RSS). If carefully designed, we are able to carry cross-technology information through the energy level of wireless packets in the air. Second, the wireless signal is flexible and redundant. A high-end wireless technology has much more degree of freedom in the signal modulation than a low-end one. Such large flexibility makes it possible for a high-end technology to emulate the signal of a low-end one in the PHY layer. The emulated signal follows the standards of the low-end technology and can be directly demodulated without any hardware or software modifications. A life analogy is that each language contains some borrowed words from a foreign language which can be understood directly by people speaking the foreign language. With these two insights, we pave the way to the CTC and get benefits from the wireless coexistence.

However, there are full of challenges in achieving CTC in both approaches. In the first approach, although energy detection is commonly available in wireless devices, the energy of a single wireless packet is always submerged in noise and hard to detect. It is a question how to detect cross-technology message from the background noises. In the second approach, it is a question how to bridge heterogeneous PHY layer (de)modulation. And in both approaches, we need to carefully consider the cost of deployment. In other words, the modification in the hardware and software should be small enough so that the system is compatible with commercial off-the-shelf devices.

1.2 Outline and Contributions

This dissertation studies how to build CTC both in the packet level and the PHY layer. The outline and the primary contributions of the dissertation are as follows

- **Packet Level Cross-technology Communication (Chapter 4)**

This chapter studies how to build CTC based on the data packets energy patterns universally interpretable regardless of underlying PHY layers, referred to as DCTC. Unlike the state-of-the-art works which require injecting dummy packets of certain sizes [6, 7], constructing customized preambles [8], or relying on the periodic beacon packets [9], the idea here is more general and friendly to existing wireless protocols. The highlight of the DCTC design is that it resolves two unique challenges in the wireless data traffic: (i) the traffic pattern is hard to estimate due to the variety of upper layer applications and (ii) small perturbation in the wireless packets may have a huge impact on the throughput. We tackle these issues by proposing a novel packet perturbation method that is independent of the traffic pattern while the delay upperbound of the perturbation method is small and guaranteed. Our experiment results have shown that we can achieve a data rate of $760bps$ while guaranteeing the delay as small as $0.5ms$ so that it won't affect the upper layer applications.

- **PHY Layer Cross-technology Communication (Chapter 5)**

Although packet-level CTC methods work, they are intrinsically limited in the data rate compared to the standard wireless protocols, such as the $250k$bps ZigBee and the $1M$bps Bluetooth, due to the sparse wireless packets in the air. To boost the CTC data rate, in this chapter, we introduce another way to construct CTC. Instead of relying on energy detection, we construct CTC based on the PHY layer signal emulation to dramatically improve the CTC data rate approaching that of the standard wireless protocols. Without loss of generality, we propose BlueBee, a PHY-layer CTC technology from Bluetooth to ZigBee. BlueBee emulates normal ZigBee signal in the payload of a normal Bluetooth packet without any hardware or firmware modification. The emulated packet is fully compatible with the ZigBee standard so that it can be demodulated at any commercial ZigBee receivers. Our extensive experiments have shown that BlueBee achieves a data rate of $225kbps$, approaching the cap ZigBee data rate, which is enough for many real-time applications such as network coordination.

- **Bidirectional Cross-technology Communication (Chapter 6)**

  Signal emulation enables CTC from a high-end device to a low-end device but can hardly provide feedback due to the lack of degree of freedom. In other words, a low-end wireless technology cannot emulate the signal of a high-end one. The lack of bidirectional communication greatly restricts the applications of PHY-layer CTC in real network protocols. In this chapter, we study how to overcome the challenges of CTC from a low-end transmitter to a high-end receiver. The main idea is to make the transmitter unchanged but move the complexity to the high-end receiver. By cross-decoding the native packets emitted from the low-end transmitter, we successfully enable bidirectional communication for the PHY-layer CTC. We have evaluated our system on the commercial Bluetooth and ZigBee platforms and achieved an over 90% success rate of bidirectional communication which covers the gap left in existing PHY-layer CTC.

Chapter 2

Background

To provide sufficient context, this chapter discusses essential background concepts related to this dissertation, including IoT networking and wireless heterogeneity and coexistence.

2.1 IoT Networking

IoT is short for "Internet of Things" which aims at connecting every physical object to the Internet so that services and controls can reach every corner of our life. IoT will enable a lot of smart applications including but not limited to home automation, transportation, health care, manufacturing and environmental protection. To be non-intrusive to the mobility of physical objects, wireless technologies such as WiFi, LTE, ZigBee, and Bluetooth are the fundamental technologies in IoT networking.

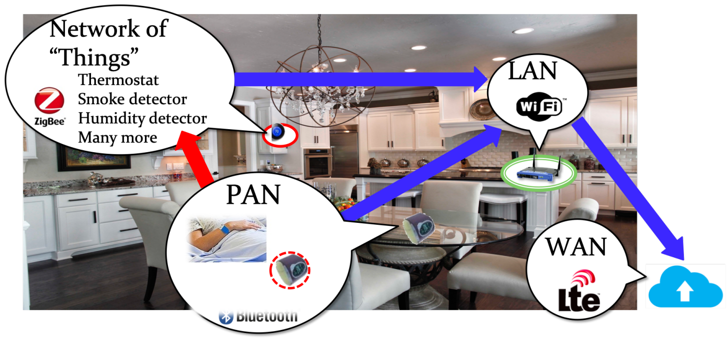


Figure 2.1: Heterogeneous IoT networks in a smart home scenario

Figure 2.1 demonstrates a typical IoT network in a smart home scenario where the heterogeneous wireless technologies form networks of different scales. The Bluetooth radio is popular in most wearable devices to form the personal area network (PAN). The ZigBee radio is a low power technology suitable for battery operated machine-to-machine communication, such as thermostats, smoke detectors, and humidity detectors. WiFi and LTE are the choices for wide-band data communication the local area network (LAN) and wide area network (WAN) respectively. These wireless technologies work collaboratively to collect, forward, and process data to achieve home automation.

Table 2.1: Wireless technology specifications

	ZigBee (IEEE 802.15.4)	**Bluetooth (IEEE 802.15.1)**	**WiFi (IEEE 802.11a/b/g/n)**
Freq. band	2.4GHz ISM	2.4GHz ISM	2.4 & 5GHz ISM
Channel bandwidth	2MHz	1MHz	20/40/22MHz
Modulation	OQPSK	GFSK	BPSK, QPSK, M-QAM
Spreading	DSSS	FHSS	DSSS, CCK, MIMO, OFDM
Range	10-100 m	10m	100m
Max data rate	250kbps	1Mbps	600Mbps
Energy consumption	Very low	Low	High

2.2 Wireless Heterogeneity and Coexistence

In Table 2.1, we have listed the technical specifications of the three most popular wireless technologies, ZigBee, Bluetooth, and WiFi. To serve different applications and scenarios, these three technologies are designed with heterogeneous channel bandwidth and adopt different modulation and spread schemes. As a result, they show quite different communication ranges, cap data rate, as well as energy consumption. For example, WiFi occupies a wide bandwidth (20/40/22 MHz) and sophisticated and various modulation schemes to provide reliable and speedy data communication. In contrast, Bluetooth is designed for data exchanging over short distances. It sacrifices data communication ranges (10m) for high data rate (1Mbps) with only 1MHz bandwidth. ZigBee, however, is optimized in the reliability and energy consumption especially suitable for long-term sensor deployment in the wild. All these technologies form separate networks in real applications.

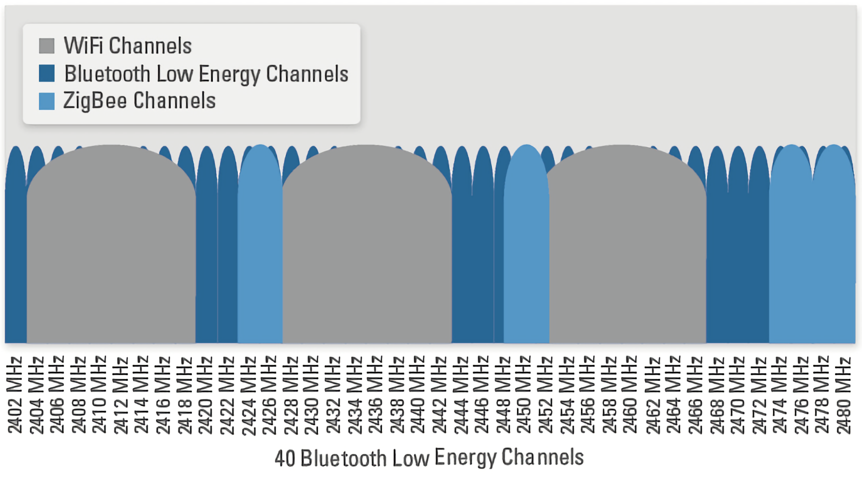


Figure 2.2: Overlapping wireless channels in the ISM band

However, in Table 2.1, we observe all these technologies coexist in the 2.4GHz ISM (industrial, scientific and medical) band for the rareness of the spectrum. In fact, the channels of these technologies are partially or completely overlapped in the 2.4GHz ISM band as shown in Fig. 2.2. Wireless packets will be disrupted if they collide in the air. Although wireless MAC protocols design is a well-studied topic, sophisticated and efficient methods such as time-division multiple access (TDMA) and Request to Send / Clear to Send (RTS/CTS) are not applicable in the cross-technology scenario for the lack of communication. As a result, the cross-technology interference is still an unsolved and challenging problem in academia.

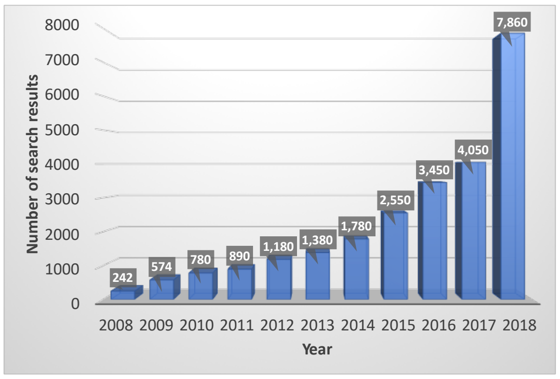


Figure 2.3: Number of news articles about wireless interference over the last decade

In our life, the public has been aware of the wireless interference and reported in various domains. The number of news articles about wireless interference from Google in a decade is depicted in Fig. 2.3. The number of news articles with the keyword of ‘wireless interference’ was 242 a decade ago and increase by $32\times$ to $7,860$ in 2018. These articles cover a wide range of wireless applications from poor WiFi connectivity at home to the interference of 5G radios with the weather satellites. So it is critical to understand the mutual impact between heterogeneous wireless technologies in the overlapped spectrum to mitigate or even take advantage of the interference for a harmonious wireless environment.

Chapter 3

Related Work

The diversity of wireless technologies makes the harmonious coexistence of heterogeneous wireless technologies on the common ISM band an even more challenging and urgent problem. The efforts in the academia trying to relieve and solve this problem can be categorized as follows: (i) wireless communication under cross-technology interference, (ii) packet-level cross-technology communication, and (iii) physical layer signal manipulation between heterogeneous wireless devices.

3.1 Communication under Cross-Technology Interference

Popular wireless technologies on the 2.4G ISM band, such as WiFi, ZigBee, Bluetooth, and LTE-U are competing for the spectrum. However, their heterogeneous PHY and asymmetric link blind them from detecting the existence of each other, which bring significant cross-technology interference(CTI)[2, 3, 4, 5]. To alleviate this issue, there have been numerous research works on alleviating the CTI by detecting and avoiding the interference, or recovering the corrupted signal from the interference[10, 11, 12, 13, 14, 15, 16, 17].

Although these solutions are effective, they generally require modification of the PHY layer mechanisms, or suffer from the dynamic interference patterns. In contrast, CTC requires no changes in the commodity devices and takes advantage of the heterogeneous wireless signal to build direct communication. It is potential to fundamentally alleviate the CTI problem.

3.2 Packet-level Cross-Technology Communication

In recent years, researchers propose cross-technology communication (CTC) which directly builds the communication between heterogeneous devices [9, 18, 6, 7, 8]. 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, all existing methods have some limitations. For example, Esense [6], and HoWiES [7] build the CTC from WiFi to ZigBee by sending out multiple dedicated WiFi packets with specific packet durations to distinguish the CTC packets from background noises. The injected packets will saturate the wireless channel and disrupt existing wireless communication. FreeBee [9] is a non-disrupt CTC design because it relies on the mandatory WiFi beacons, and embed CTC message in a free channel by changing the transmission timings of existing beacons. However, the performance of the design is limited by the sparse number of beacons in the air. GSense[8] attach customized preambles before heterogeneous wireless packets which restricts its deployment. Finally, B^2W^2 [18] modulating the energy level of Bluetooth packets and then demodulate through WiFi CSI at the receiver side, which is specific to the frequency-hopping Bluetooth protocol and not generalized.

3.3 Physical Layer Signal Manipulation

Despite packet-level solutions, our ideas are also inspired by several recent works studying the signal manipulation[19, 20, 21, 22, 23]. In [21], researchers build the legacy WiFi packets via customized tags by utilizing the backscatter effect. In addition, in the LTE system, Ultron [19] emulates the WiFi packets via a LTE transmitter to coordinate between LTE and WiFi. However it requires the modification of existing LTE standard, and the sent frames are no longer LTE-compliant MAC frames. WEBee[24] is the first to introduce the ideal of signal manipulation into wireless communication, referred to as signal emulation, without hardware or firmware modifications on the COTS devices. However, signal emulation in WEBee requires the transmitter (i.e., the emulator) to be much more powerful than the receiver to guarantee the emulation performance. It can not be directly applied to the resource limited IoT devices such as ZigBee and Bluetooth.

Chapter 4

Packet Level Cross-technology Communication

4.1 Introduction

This chapter studies how to build CTC in the packet level. The principal behind is that wireless devices, despite of the technologies they use, are available and mandatory to sense the energy of the channel (i.e., RSSI) in order to access the channel and communicate with each other. Although devices using different technologies can not decode each other's information due to incompatible physical layer, they are all able to identify the existence of each other through channel energy sensing.

State-of-the-art techniques in this category commonly suffer from channel inefficiency [6, 7], low throughput [9], or disruption to existing networks [8], leaving significant room for further improvement. To take advantage of all kinds of wireless packets in the air while being non-intrusive to the upper layer applications, we introduce CTC via data packets (DCTC). The key concept is utilizing existing data packets which are the dominating portion of wireless network traffic. By slightly perturbing the transmission timings of enqueued data packets, we construct specific energy patterns that can be demodulated by heterogeneous wireless technologies. DCTC overcomes the limitations of state-of-the-art techniques. More specifically, DCTC (i) significantly enhances CTC throughput, while (ii) keeping the technology transparent to upper layer protocols and applications.

It is a non-trivial task. First, the dynamic nature of the volume, both bursty and sparse, affect the opportunity for CTC modulation. Moreover, delay sensitivity of the applications, especially real-time applications, such as audio/video streaming applications and online games, require the perturbation delay to be small enough to comply with applications' delay requirements.

To the best of our knowledge, this is the first work that achieves CTC with existing data packets, naturally flowing through any network entity. It features (i) specific CTC modulation independent of the arrival of future data packets, (ii) packet perturbation with bounded delay even without the knowledge of incoming packet distribution, and (iii) dynamic configuration accommodating CTC throughput to the amount of available data packets. We have implemented our design on a wireless open-access research platform (WARP[25]) and MICAz[26], compliant to the WiFi (i.e., 802.11) and ZigBee (i.e., 802.15.4) standards respectively. Our experiment results demonstrate the average perturbation delay of only $0.5ms$ while achieving 95% accuracy in transmitting DCTC symbols from a WiFi AP to a ZigBee node, showcasing the reliability of DCTC as well as its transparency to upper layer applications. We have also implemented our design from a ZigBee node to a WiFi device to achieve bidirectional communication.

Our contributions in this chapter are as follows.

- We propose DCTC, a novel CTC framework that explores existing data traffic. DCTC is designed to take advantage of the large volume of data traffic in the wild, and achieves significantly higher throughput compared with the state-of-the-art works, without incurring traffic overhead.

- Modulation in DCTC is done by perturbing the transmission timings of data packets, where the amount of perturbation delay incurred is shown to be small and bounded. This ensures transparency of our design to upper layer protocols and applications.

- DCTC requires no hardware modification and is compatible with off-the-shelf devices. Implementation of DCTC on WiFi and ZigBee platforms shows that DCTC offers reliable symbol delivery with an average delay of less than $0.5ms$. In addition, DCTC has little impact on the performance of popular network applications, such as FTP, video streaming, and audio streaming.

4.2 Motivation

This section begins with the insight on abundant CTC opportunities under today's typical environment, followed by the challenges in perturbing the timing of data packets.

4.2.1 Opportunity for CTC

State-of-the-art literatures achieve CTC either by injecting dummy packets [6, 7] or utilizing mandatory beacons [9]. The former approach introduces additional traffic that is disruptive to existing networks, with its significant spectrum occupancy. Although the latter is free from the issue as it utilizes existing beacons, it suffers from low CTC throughput due to the confined number of beacons and hence limited CTC opportunities. Therefore, from the two methods, we note that the best of both worlds can be reached by utilizing existing traffic with a sufficient volume.

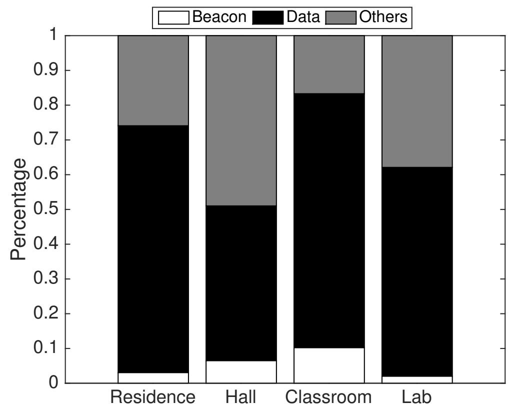


Figure 4.1: Distribution of network packets in four everyday life scenarios: residence, hall, classroom, and lab.

Fig.4.1 demonstrates the compositions of WiFi traffic experimentally observed in four typical everyday environments, including residence, hall, classroom, and lab. The figure shows that, unlike the small volume of beacons, the data packets take up the majority portion of the traffic in all four scenarios to reach over 40%. It is more than 7× of the amount of beacons. This indicates utilizing existing data traffic not only enables establishing CTC with high spectrum efficiency (i.e., without dummy packets), but also provides sufficient opportunities to reach high CTC throughput.

4.2.2 Challenge of DCTC

Although the huge volume of data packets provide us with huge opportunity to establish CTC, perturbation of data packets may degrade the performance of upper layer applications, such as the drop in throughput.

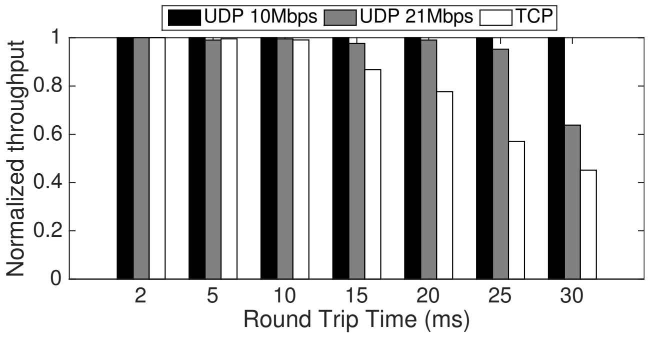


Figure 4.2: The impact of packet delay on throughput, normalized to the highest throughput of each experiment.

To study how severe packet delay (measured by round trip time, RTT) will affect the throughput of upper layer applications. We test the normalized throughput (normalized to the highest throughput) of both UDP and TCP traffic with different bandwidth using Iperf[27]. The UDP bandwidth is chosen from 10Mbps to 21Mbps (saturation bandwidth), while TCP throughput is tested by setting a large enough TCP window size, due to the fact that TCP bandwidth can not be directly set in Iperf. In Fig. 4.2, we find that, the throughput of TCP traffic decreases dramatically when RTT is higher than 10ms. That is because (i) TCP flow control mechanism restricts TCP throughput, making it inversely proportional to RTT in theory, and (ii) ACK timeout may be triggered when data packets are perturbed for a long time. Even though UDP traffic doesn't have flow control and ACK, its throughput starts to degrade when RTT is 20ms in saturation case. That is due to transmission queue overflow.

From the study, we find that the throughput of both TCP and UDP traffic, especially TCP traffic, will be affected by packet delay. Thus how to minimize packet perturbation time, so that to minimize the increment of packet delay, is the main challenge in establishing CTC via data packets.

4.3 System Design

In this section, we first introduce the objective of our design, followed by an overview of our system design, and finally the details of DCTC modulation/demodulation design.

4.3.1 Design Objective

Due to the critical impact of packet delay on the application throughput, our design aims at establishing CTC via data packets (known as DCTC) that enhances CTC throughput while keeping transparent to upper layer network applications. More specifically, our design features (i) packet perturbation with bounded delay even without the knowledge of incoming packet distribution, and (ii) dynamic DCTC throughput fitting incoming packet rate.

4.3.2 Design Overview

DCTC is a bidirectional communication system between two heterogeneous wireless communication technologies, WiFi and ZigBee. Without loss of generality, we use the communication from a WiFi sender to a ZigBee receiver to illustrate our main design. The opposite communication is based on similar principle. Fig 5.1 describes DCTC's design architecture in both the WiFi sender side and ZigBee receiver side, where the grey boxes are existing protocols/applications, and the white boxes are DCTC's layers.

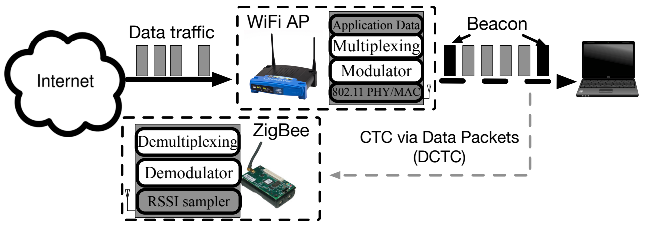


Figure 4.3: An overview of DCTC system design.

At the sender side, the top layer lies the existing applications. DCTC's opportunity comes from the data traffic of internet applications, without any modification to existing applications. Then, DCTC has its multiplexing layer, which supports concurrent transmission of multiple senders. Next, DCTC has its modulation layer, which shifts transmission timing of the packets in the transmission buffer with reference to WiFi beacons to construct DCTC symbols. Finally, since DCTC is built upon the existing 802.11 MAC/PHY layers, it follows the existing protocols, such as channel sensing, packet ACK and retransmission. These features enable DCTC to be transparent to the existing physical layer protocols.

At the receiver side, DCTC utilizes the existing sampling ability of the ZigBee nodes to sense the energy (RSSI) of the channel. Although the ZigBee receiver can not demodulate WiFi signal directly due to the incompatible physical layers, DCTC demodulator can demodulate DCTC symbols through RSSI sampling results. In addition, the DCTC receiver relies on its de-multiplexing layer to recognize different senders.

4.3.3 Modulation

In CTC scenario, WiFi beacon has been proved to be reliably demodulated by heterogeneous radio through its energy pattern [9]. However, employing that idea directly to data traffic is not practical. That is because regulating dynamic data traffic to be periodic may cause huge increase in packet delay or severe buffer control problem. We propose the modulation of DCTC which establishes detectable energy patterns while introducing little and delay bounded perturbation to existing data traffic.

Without loss of generality, we use the process of modulating 1-bit information at a single WiFi AP as an example. We regard the packets transmitted by other wireless transmitters as background noise.

A WiFi AP will broadcast WiFi beacons periodically due to the 802.11 protocol. Within a beacon period, we set some *critical time points* with period Δ, which is a user defined parameter shared with both sides in advance. Critical time points have alternating labels to indicate bit “0” and bit “1” respectively. To modulate a data packet with minimum delay, we perturb it to cover the nearest critical time point.

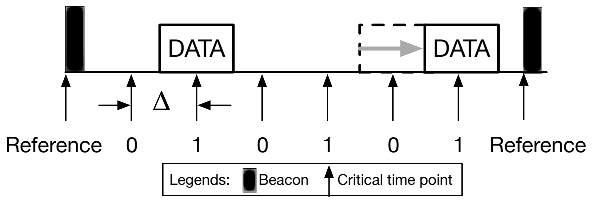


Figure 4.4: Illustration of DCTC symbol modulation

In Fig. 4.4, we illustrate how to modulate bit “1”. To do that, we perturb all the data packets to cover their closest time point “1”. The first packet has already covered critical time point “1”, so we don’t move it. The second one, however, doesn’t cover critical time point “1”, so we perturb it to cover the next time point “1”. Note we do not regulate the random incoming data packets to be periodic, i.e., to cover all time point “1”s, but to slightly adjust the energy pattern according to its original incoming pattern. In this example, there are three critical time point “1”s, but data packets only cover two of them. To extend the design to multi-bit case, we can either cut a beacon period into small segments or a more advanced method discussed in Section 4.4.1.

4.3.4 Demodulation

To demodulate DCTC symbols, the ZigBee receiver needs to first synchronizes with the WiFi sender by detecting beacons of the transmitter from all kinds of noise packets in the air. Then it extracts DCTC symbols with reference to the beacons of the transmitter.

• **Step 1: Synchronization**: An 802.15.4-compliant ZigBee node recognizes WiFi beacons under the presence of channel noise by detecting their periodic energy pattern. The ZigBee node continuously senses the energy of the channel, and records the RSSI values from the RF chip in its flash. Then it quantizes the captured RSSI values to be high and low to indicate busy and idle channels (denoted as dark and white boxes in Fig. 4.5). The threshold is set to be -75dBm following the CCA (clear channel assessment) threshold for the 802.15.4 standard [28]. The sampling rate of the ZigBee node is $7.8KHz$, which makes a measurement spanning of 128μs to avoid time gaps in sampling adjacent data packets.

Next, the ZigBee node applies *folding* to the quantized RSSI sequence, which is a signal processing technique to extract periodic signal from noise [29]. The process of folding is as follows. The received RSSI sample sequence is divided by the beacon interval, which is a preset value for WiFi APs. All the samples are stacked together, and a column-wise sum is applied to calculate the number of samples with high RSSI values in each column, which is denoted as *fold sum.* Since WiFi beacons are periodic, they will align column-wise and preserve a higher fold sum than other signals.

In Fig. 4.5, we illustrate how to detect beacon positions from a quantized RSSI sequence. We cut the quantized RSSI sequence into 3 subsequences of length $T = 5$,

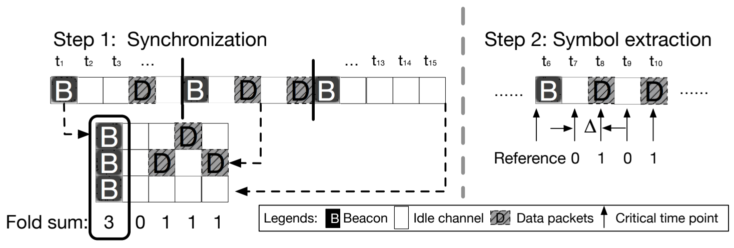
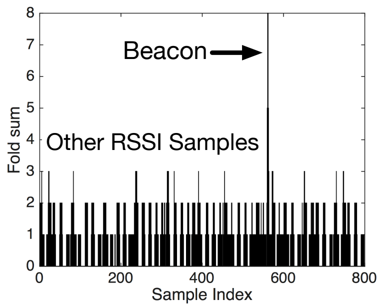
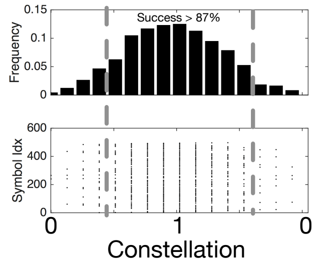

Step 1: Synchronization
Step 2: Symbol extraction
Reference 0 1 0 1
Fold sum: 3 0 1 1 1
Legends: Beacon Idle channel Data packets Critical time point

Figure 4.5: Illustration of DCTC symbol demodulation

where T is the period of beacons. Then we stack them together and calculate the fold sum. We find that the fold sum of the first column is the highest among all the columns. Then we detect the beacon position at the first column.



(a) Synchronization



(b) Symbol extraction

Figure 4.6: DCTC symbol demodulation in practice

In practice, as illustrated in Fig. 4.6(a), the beacon period is set to be 102.4ms, which means there are 800 ZigBee RSSI samples in one beacon period. After folding, we find that the fold sum at index 572 exceeds the fold sums of the others, and thus we demodulate the beacon position.

• **Step 2: DCTC Symbols Extraction**: With reference to the WiFi beacon, we check quantized RSSI samples every Δ seconds at critical time points. Just as in the modulation process, critical time points have alternating labels “0” or “1”. If the channel RSSI is high at a certain critical time point, we regard the successful reception of an according DCTC symbol “0” or “1”. Then we accumulate the DCTC symbols we received within a beacon period. The dominating symbol in the whole beacon period is interpreted as the symbol conveyed.

In Fig. 4.6(b), we illustrate how to extract DCTC symbol in practice. We depict the RSSI samples with high RSSI values in constellation figure. We find that over 87% of the high RSSI values are near critical time point “1”, with the highest frequency appear at exactly the time point “1”. Since the frequency of time point “1” is larger than that of time point “0”, the beacon period is demodulated as symbol “1”.

4.3.5 Design Features

The main features of DCTC design is that the packet perturbation in the CTC modulation has limited impact on the original network traffic and protocol shown as follows.
• **Bounded Traffic Delay**: The perturbation operation guarantees a bounded packet delay. Since we perturb data packets to the nearest critical time point, each data packet will be perturbed at most 2Δ to cover the correct time point. Considering that the incoming data packets arrive in random, so the amount of perturbation is uniformly distributed in the interval $[0, 2\Delta)$. So the average amount of perturbation is Δ.
• **Limited Impact on Throughput**: The perturbation operation has limited impact on the packet throughput for two reasons. First, in most time, packet intervals are larger than the critical time point interval, i.e., data packets are perturbed to cover different critical time points. In that case, perturbation of one packet won’t affect the timing of the next one. Second, even though the incoming traffic is dense, we still have room to flush out surplus data packets. We can flush out data packets between two critical time points, because the receiver only detects at critical time points. Data packets flushed within two critical time points won’t be demodulated as any symbol, thus won’t affect symbol demodulation. In this way, the perturbation has little impact on traffic throughput.
• **Compatibility with CSMA**: The perturbation operation is compatible with CSMA random backoff. In CSMA, a random backoff timer is set when there are data packets to transmit. The timer will count down whenever the channel is sensed to be idle. The actual time backed off depends on the environment and hard to estimate, which affects the accuracy of modulation. In our design, the modulation of data packets is after CSMA’s random backoff, i.e., we add small additional time delay after normal CSMA backoff. Thus we can control the timing of packets accurately while maintaining compatible with CSMA protocol.

4.4 Advanced Design

In this section, we propose several advanced designs that help improve the throughput of DCTC while still maintaining transparent to upper layer applications. One is a multi-bit embedding technique, the other is a multiplexing technique.

4.4.1 Multi-bit Embedding

Although we can modulate n bits within a beacon period by separating a beacon period into n small segments. Its performance degrades at dynamic packet rate. Sometimes there are insufficient data packets within one segment to modulate a symbol while surplus data packets within another. To address that issue, the sender should be able to dynamically decide when to modulate a bit depending on the incoming packet rate. To do that, instead of using the whole beacon period to modulate one bit, the sender can repeat a symbol a certain number of times, above some threshold, then turn to the next bit. The threshold is set according to the channel noise level. Detailed discussion is in Section 4.5.1.

One issue that needs to be solved is that, without using beacon as boundary, it becomes hard to distinguish neighboring symbols, especially when we repeat a symbol multiple times. For example, when the receiver receives four bit "1"s, we can not tell whether the sender is transmitting four repeats of a single bit "1" or two bits that are the same, i.e., two repeats of the first symbol "1" and two repeats of the second symbol "1".

To address this issue, we adopt a coding method similar to the hybrid ternary code [30], which transfers any 0-1 sequence to a new sequence where no consecutive bits are the same. To do that, we introduce a new bit "R", which means "Redo". To encode any 0-1 sequence, we simply scan from left to right, whenever we find a bit that is the same as its previous one, we change it to be "R". To decode the code, we just interpret "R" as its previous symbol. For example, a sequence "111100" will be coded as "1R1R0R", where no consecutive bits are the same. The only change to the basic modulation is that we now have three types of critical time points, "0", "1", and "R", instead of two.

To demodulate the symbols, the receiver scans at critical time points and uses three counters to count the number of each type of bits respectively. Whenever it has detected a threshold number of certain bit (e.g., five bit “1”s), it demodulates a bit (e.g., bit “1”). Then the receiver clears the three counters and disables the counter for the bit already demodulated (e.g. counter of bit “1”). That is because we know no consecutive bits are the same. We will enable the bit counter after we have demodulated another bit.

With the multi-bit embedding technique, we can achieve multi-bit embedding adapting to the dynamic data traffic at the sender side, while still being able to demodulate symbols at receiver side.

4.4.2 Multiplexing

Here we propose how to achieve multiplexing among multiple DCTC senders. That means multiple DCTC senders can transmit at the same time while each of their receivers can still demodulate DCTC symbols correctly. In this way, the accumulated DCTC throughput can be improved.

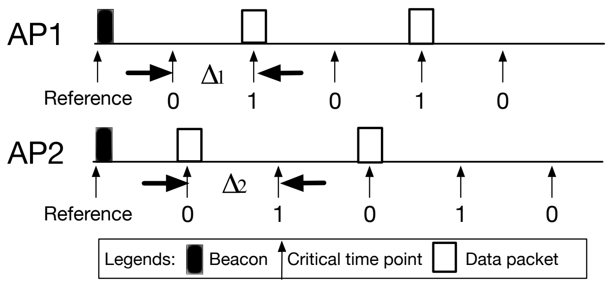
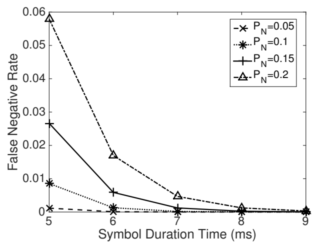


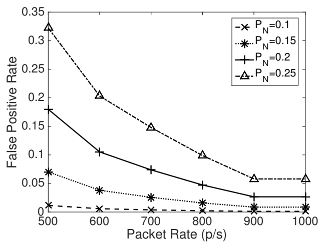
Figure 4.7: Illustration of DCTC multiplexing.

We achieve multiplexing by carefully choosing the critical time point intervals assigned to different APs, so that the critical time points from any two different WiFi APs will meet at most once within any beacon periods. As illustrated in Fig. 4.7, AP1 and AP2 have their critical time points with interval Δ_1 and Δ_2 respectively. They can modulate their DCTC symbol at their own critical time points without interfering with each other. To find such pairs of critical time point intervals, we have the following theorem.

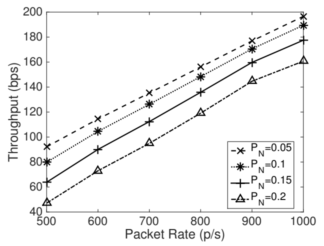
Theorem 4.4.1. *Let CTP_1 be a sequence of critical time points with interval Δ_1 and CTP_2 be a sequence of critical time points with interval Δ_2. CTP_1 and CTP_2 will*



(a) False negative error



(b) False positive error



(c) Throughput

Figure 4.8: Simulation of DCTC's performance under noise

overlap at most once within beacon period T if

$$LCM(\Delta_1, \Delta_2) \geq T, \tag{4.1}$$

where LCM is the the least common multiple.

Proof. We prove by contradiction. Assuming CTP_1 and CTP_2 overlap more than once with T. Without loss of generality, we say they meet at least at time t_1 and t_2, where $0 < t_2 - t_1 < T$. Then $t_2 - t_1$ is the common multiple of Δ_1 and Δ_2, and it is less than T, which conflicts with the fact that $LCM(\Delta_1, \Delta_2) \geq T$. □

Theorem. 4.4.1 guarantees that two critical time point sequences have minimum cross-talk interference between them. Thus we can achieve concurrent transmission of multiple DCTC senders.

4.5 Analytics

In this section, we analyze the symbol error rate as well as the throughput of DCTC.

P_N	Noise rate
N_s	Number of repeat symbols at the sender
N_r	Threshold number of repeat symbols at the receiver
R_s	Packet rate at the sender
Δ	Critical time point interval

Table 4.1: Parameters

4.5.1 Symbol Error Rate

The wireless channel is shared by multiple wireless devices and is always crowded with wireless traffic. In CTC, packets from devices other than DCTC senders are regarded as noise. So the noise ratio P_N can also be regarded as the channel occupancy rate. The noise in the air brings two problems to the accuracy of DCTC: (i) it may occupy the correct time point at the sender side (but not heard by the receiver, due to hidden terminal issue) and prevent the sender to perturb data packets to the right time point, which we refer to as the *false negative problem*, or (ii) it may occupy the wrong time point at the receiver side, causing the false detection of a wrong time point, which we refer to as the *false positive problem*.

To address both problems, we repeat one symbol multiple times, denoted as N_s, at the sender. If the receiver can detect certain number of the repeats, denoted as N_r, it successfully demodulates a symbol. Thus the false negative rate P_{FN} can be formulated as

$$P_{FN} = \sum_{k=0}^{N_r-1} \binom{N_s}{k} (1-P_N)^k P_N^{N_s-k}, \tag{4.2}$$

which means the probability that we can detect more than or equal to N_r correct time points at the receiver side.

The false positive rate P_{FP} is affected by the packet rate at the sender R_s. The intuition is that, when packet rate is low, it takes more time for the sender to send one symbol. So the receiver has more chance to be affected by the false negative issue during one symbol time. The average time to send one symbol can be derived as $\frac{N_s}{R_s}$, which means the time DCTC sender takes to transmit N_s packets. During that time, the number of critical time point sampled at the receiver side is $\frac{N_s}{R_s \times \Delta}$. Among them, half number of the critical time points are the wrong ones. So the total number of wrong critical time points at the receiver side is $N_w = \frac{N_s}{2R_s \times \Delta}$. False positive problem happens when noise packets occupy more than N_r wrong critical time points, which can be formulated as

$$P_{FP} = 1 - \sum_{k=0}^{N_r-1} \binom{N_w}{k} P_N^k (1-P_N)^{N_w-k}. \tag{4.3}$$

With the false negative and false positive rate, we can derive the symbol error rate

(SER) as

$$SER = (1 - P_{FN}) \times (1 - P_{FP}). \tag{4.4}$$

In Fig. 4.8(a) we depicts the false negative rate with increasing symbol duration, which is $\frac{N_s}{R}$ (R is fixed to $1000p/s$). We find that when we increase the number of symbols repeated at the sender, the false negative rate decreases dramatically. It decreases to under 1% when we repeat a symbol 7 times at the sender. In Fig. 4.8(b), we study the false positive rate with increasing packet rate while fixing N_s to be 5. We find that false positive rate decreases under 10% when packet rate is larger than $800p/s$. That is because when packet rate increases, N_w decreases, so the receiver will have less chance to be affected by noise.

4.5.2 Throughput

We need to achieve maximum DCTC throughput, which depends on both the demodulation accuracy and DCTC data rate. To modulate a symbol more data packets (higher N_s), we can guarantee higher symbol accuracy (P_s), but decrease the DCTC data rate ($\frac{R}{N_s}$). To maximize DCTC throughput, we formulate it to be an optimization problem,

$$max_{N_s,N_r} SER \times \frac{R}{N_s}. \tag{4.5}$$

Solving the optimization problem is out of the scope of this dissertation. We can use some optimization toolboxes to find the solution.

In Fig. 4.8(c), we depict the DCTC throughput with increasing packet rate. We find that when the packet rate increases, DCTC throughput also increases. When the error rate is 20%, the DCTC throughput increases from $47bps$ to $160bps$. The throughput increases faster when packet rate is large because the decrease in SER.

4.6 Evaluation

In this section, we evaluate DCTC's performance in throughput, transparency, error rate under noise, multiplexing and receiver overhead.

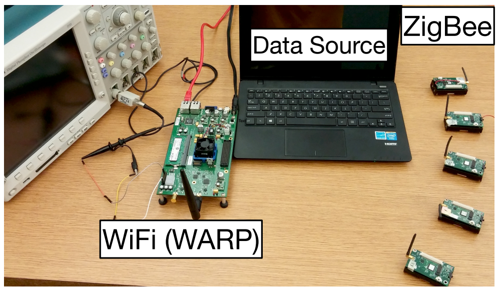


Figure 4.9: Experiment setting.

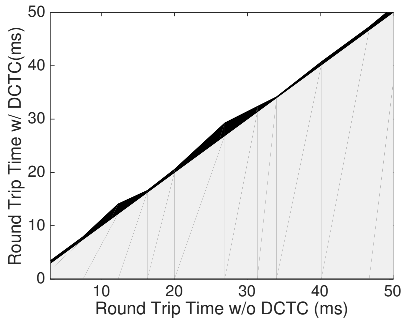
4.6.1 Setting

We have implemented bidirectional DCTC communication between WiFi and ZigBee nodes through data packets from applications. Fig. 4.9 shows the experiment setting of our design. We use a laptop as a data packet generator through network applications. The laptop is connected to a WiFi AP equipped with DCTC design. Although we have implemented DCTC on both off-the-shelf WiFi chipset (laptop) and WARP [25], which is a wireless research platform implemented with WiFi PHY/MAC layers, the following evaluation results are based on WARP platform, for its convenience in changing parameter setting. We also have 5 ZigBee-compliant MICAz nodes, which can sample the channel and capture RSSI values at $7.8KHz$, and records them within its $512KB$ on-board flash. WiFi AP works on WiFi channel 1, and ZigBee works on ZigBee channel $11-14$, overlapped with WiFi channel 1.

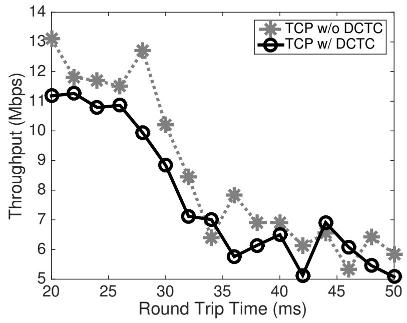
4.6.2 Throughput

We here compare the throughput of DCTC with the state of the art.

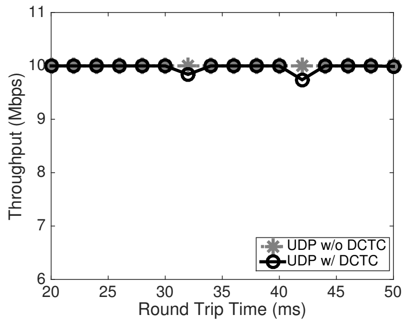
• **WiFi→ZigBee**: Fig. 4.11 compares the achievable throughput of DCTC from WiFi to ZigBee with two state-of-the-art works, ESense and FreeBee. According to the author, with multiple ESense senders working in parallel, ESense achieves a bit rate of $1.63Kbps$. Since five consecutive packets are need in ESense to guarantee reliable transmission of a symbol, the actual throughput of ESense is $326bps$. Under the same setting, asynchronous FreeBee achieves $560bps$ in throughput via concurrent transmission of hundreds of FreeBee senders. For DCTC, the maximum throughput is restricted by the sampling interval at the receiver side. The ZigBee receiver samples one RSSI value



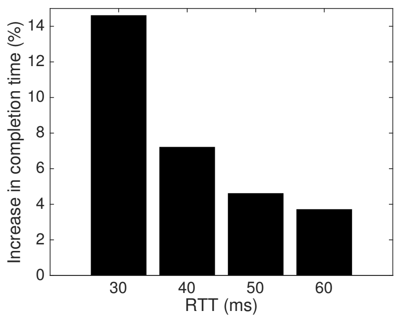
(a) RTT w/ and w/o DCTC



(b) TCP throughput w/ and w/o DCTC



(c) UDP throughput w/ and w/o DCTC



(d) Increment % of FTP file transmission time

Figure 4.10: DCTC's transparency to upper layer applications

every $128\mu s$, and we need one time point for each of the two symbols. In addition, we can guarantee reliable ($> 95\%$) symbol detection by repeating a symbol 5 times, which results in a throughput of $760bps$ for DCTC, which is 1.3× of FreeBee, and 2.3× of ESense.

• **ZigBee→WiFi**: Here we test the performance of DCTC from ZigBee network to WiFi. FreeBee achieves throughput of $146bps$ from ZigBee to WiFi. In DCTC, since WiFi can sample the channel much faster, the bottleneck is the transmission rate of ZigBee node. The maximum transmission rate of MICAz is $1000p/s$, also we can guarantee over 95% accuracy by demodulating 5 data packets. So DCTC's throughput from ZigBee to WiFi is $190bps$, which is 1.3× of FreeBee.

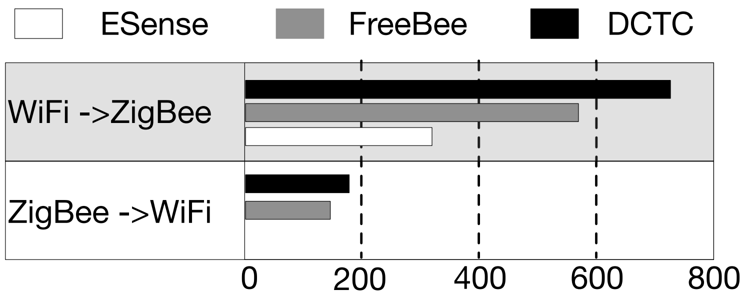


Figure 4.11: Throughput of bidirectional communication between WiFi and ZigBee.

4.6.3 Transparency

In this part, we evaluate DCTC's transparency in packet round trip time and throughput. In addition, we will also use some popular network applications, such as FTP, audio streaming, and video streaming to evaluate DCTC's impact on real network applications.

• **End-to-End Delay** Here we study DCTC's impact on network end-to-end delay. In the experiment, the critical time point interval is set to be $512\mu s$, and we use the Linux command "Ping" to test the RTT, which is an indicator of end-to-end delay, w/ and w/o perturbation. In addition, we use DummyNet to emulate different network end-to-end delay.

In Fig. 4.10(a), the black area is the additional delay introduced by DCTC. We find that the RTT w/ DCTC is within $2ms$ of RTT w/o DCTC. Considering that it is natural to have $1ms$ fluctuation in the normal RTT due to dynamic network situation, results in Fig. 4.10(a) can show that our design guarantees a bounded perturbation delay.

• **End-to-End Throughput** Here we compare how our design will affect the throughput of normal WiFi traffic, both TCP and UDP traffic. We use Iperf[27] to generate TCP and UDP traffic. TCP is given a large enough window size and UDP is set a constant bandwidth of $10Mbps$.

From Fig. 4.10(b), we find that when RTT is 20ms, the throughput of TCP traffic degrades about 15.3%. As the RTT increases, the throughput difference between TCP w/ and w/o DCTC becomes smaller. That is because DCTC has a bounded delay to original TCP traffic, and the throughput of TCP traffic is almost inverse proportional to

RTT. As a result, when RTT is small, throughput degradation is large and it decreases as RTT increases. The impact of DCTC on TCP throughput can be omitted when RTT is larger than or equal to 40ms. From Fig. 4.10(c), we find that the throughput of UDP traffic keeps stable as the round trip time increases. We find that the UDP throughput w/ and w/o DCTC are almost the same, which verifies the transparency of DCTC to the network traffic.

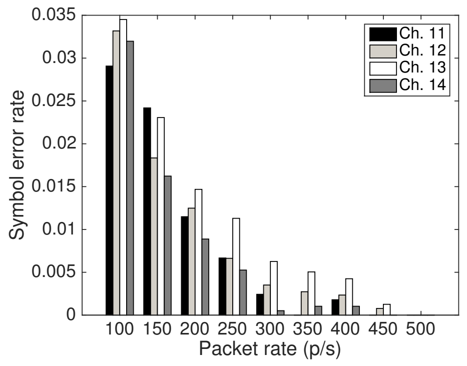
• **Impact on Network Applications** Here we use 3 applications: FTP, Audio streaming, and Video streaming. The performance of the applications are as follows:

(i) FTP traffic: We construct FTP traffic with the tool vsftpd [31] on Linux platform. We transfer a large file of size 55MB through an AP equipped with DCTC layer. Without DCTC, it takes about $48s$ to finish transmitting the file ($RTT = 30ms$). With DCTC, the increment of ftp completion time is shown in Fig. 4.10(d). The increment of ftp completion time decreases as the RTT increases. When network RTT is $30ms$, the increment in completion time is 14.6%, and the increment decreases to 3.7% when network RTT is $60ms$. It shows that the impact on FTP is small and won't affect normal functionality, especially for long-distance file transmission. Since FTP is upon TCP protocol, the result is consistent with DCTC's impact on TCP throughput.

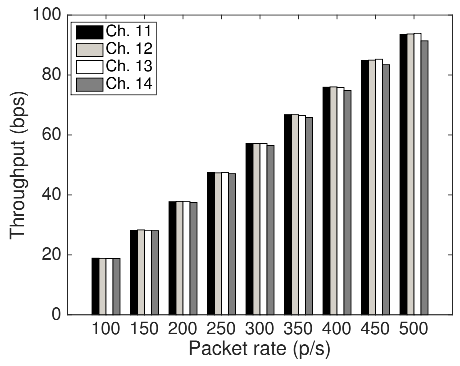
(ii) Audio & Video traffic: We set up an audio & video streaming server using the tool PLEX [32] and connect it to an AP equipped with DCTC layer. Then we play the audio & video stream in web browser from a remote laptop. We find that even in a network with high RTT ($60ms$) we can play a MP3 audio stream and a high quality video stream (> 720p) smoothly. The reason is that DCTC has a limited impact on the TCP and UDP throughput. So applications built upon them can work well as usual.

4.6.4 SER

We here evaluate the SER for ZigBee nodes to demodulate DCTC symbol. In the experiment, one laptop is connected to WiFi AP (WARP) and generates UDP packets. The WiFi AP establishes DCTC symbols using the data packets flowing through it at WiFi. The WiFi AP works at the 20MHz WiFi channel 1, which covers four 2MHz ZigBee channel from 11 to 14. We transmit more than 3,000 DCTC symbols with the packet rate of laptop varies from $100p/s$ to $500p/s$ and the DCTC throughput fixed to be $10bps$.



(a) Demodulation error rate with different packet rate



(b) DCTC throughput with increasing packet rate

Figure 4.12: DCTC's SER and throughput with increasing packet rate

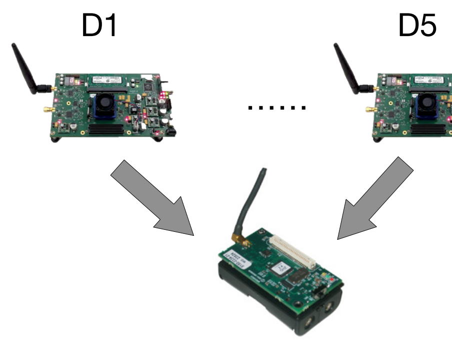
In Fig. 4.12(a), we find that the symbol error rate decreases as the packet rate increases. When packet rate is higher than $250p/s$, SER of all four ZigBee channels becomes lower than 1%. That is because with higher packet rate, we use more data packets to modulate one symbol, so that the symbol error rate is lower. We note that Fig. 4.12(a) shows that we can demodulate DCTC symbols with high accuracy at all four ZigBee channels, which provides the potential for cross-channel broadcast from WiFi to ZigBee through data traffic.

4.6.5 Impact of Traffic Volume

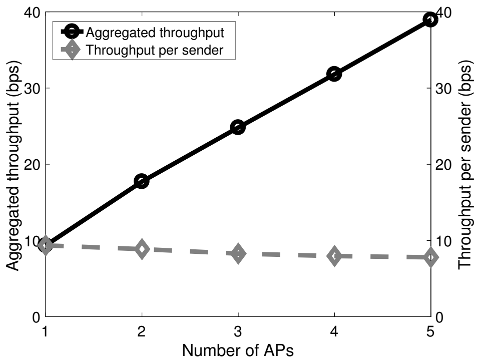
In Fig. 4.12(b), we study the DCTC throughput as WiFi packet rate increases. We find that the DCTC throughput increases as WiFi packet rate increases. When packet rate is $500p/s$, DCTC can reach about $90bps$ in throughput. That means DCTC can adapt to the volume of network traffic and make full use of DCTC opportunity. In addition, the DCTC throughputs at different ZigBee channels are close. That is because a WiFi sender covers four ZigBee channels and can broadcast to ZigBee devices at these channels.

4.6.6 Multiplexing

In Fig. 4.13, we study the concurrent DCTC symbol transmission. We have five WiFi APs that are embedded with DCTC. The critical time point interval of these APs are set $10ms$, $11ms$, $13ms$, $15ms$, and $17ms$ respectively, which have minimum cross-talk interference between each two of them. We open these WiFi APs one after another and evaluate the DCTC throughput of a single AP as well as the aggregated throughput of multiple APs.



(a)



(b)

Figure 4.13: Concurrent DCTC symbol transmission with up to 5 APs (D1 - D5)

We find that when only one AP is on, its DCTC throughput can be as high as $9.3bps$. As more APs add in, the DCTC throughput of the AP slightly decreases. When there are 5 APs working together, the throughput of a single DCTC sender is about $7.7bps$, due to noise in the busy channel. However, the aggregated throughput of the five APs is almost 4.1× the throughput of one AP.

4.6.7 Receiver Overhead

In this section we demonstrate the light-weight memory storage and computing cost for DCTC. So that it can work on off-the-shelf ZigBee nodes and won't disturb legacy network protocol.

• **Storage**: In DCTC, the most storage cost is in detecting the position of WiFi beacons. We need to store enough RSSI samples temporarily to detect the position of periodic WiFi beacon. We need to store 800 quantized RSSI samples in each beacon period, which occupies $800bits$. In addition, we need to fold 5 times to reliably demodulate beacon position, which causes a total storage of $500Bytes$, less than 0.1% of MICAz's $512KByte$ memory.

• **Computation**: The steps of DCTC modulation can be divided into 4 parts: (i) sampling RSSI, (ii) calculating fold sum, (iii) locating beacon position, (iv) checking critical time points, and (v) demodulating DCTC symbol. Among them, (i) is automatically done whenever ZigBee nodes are open. After collecting enough RSSI samples (i.e., $\rho \times f \times T$), we need to do $\rho \times f \times T$ addition or comparison in (ii) and (iii). The cost of step (iv) and (v) are related to critical time point interval Δ, the number of symbols in one beacon period n, and number of concurrent APs k. The total cost of addition and comparison is $k \times \frac{f \times T}{\Delta \times n}$ per symbol. That means the computation cost increases linearly with the number of APs.

4.7 Conclusion

In this chapter, we propose DCTC, a novel CTC system based on existing data traffic in the network. Different from other related studies in CTC, we focus on making full use of the large amount of data packets in the air to improve DCTC's throughput, while reducing its effect on original data traffic. Our approach not only generates detectable energy patterns, but also guarantees perturbation delay bound. We have implemented bidirectional DCTC communication between two heterogeneous wireless technologies, WiFi and ZigBee. Experiment results show that we can achieve reliable ($>$ 95%) bidirectional DCTC communication, while achieving 2.3× throughput compared to the state of the art.

Chapter 5

PHY Layer Cross-technology Communication

5.1 Introduction

Despite their effectiveness of packet level CTC methods, their data 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 data rate of BLE to ZigBee communication in the state of the art is limited to 18bps [9]. This not only constraints the usage, but also indicates spectrum inefficiency compared to 250kbps and 1Mbps if used for legacy ZigBee and Bluetooth.

To boost the CTC data rate, this chapter introduces exploring PHY-layer information for constructing CTC. Specifically, we propose 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 [9]. 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 [9], BlueBee's reliable throughput of 225kbps indicates performance gain of more than 10,000 times!

5.2 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 [33, 34, 35]. 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 [36, 37, 4, 38, 39]. 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 [40], thus leading to a better channel efficiency. Unfortunately, to our knowledge, existing CTC designs [9, 6, 7] rely on sparse packet level information such as the beacon timing [9] and multi-packet sequence patterns [41], 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 chapter, 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 5.7.1).

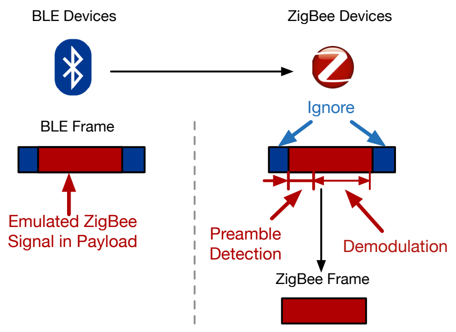


Figure 5.1: The system architecture of the BlueBee.

5.3 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. 5.1 – 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 [6]	Low	Low	Low	Not Support
FreeBee [9]	Low	Medium	Low	Not Support
B^2W^2 **[18]**	Low	Medium	Low	Support
BlueBee	**Low**	**High**	**High**	**Support**

Table 5.1: Comparison of BlueBee and existing CTC solutions

Unique Features. In Table 5.1, 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 [9, 41, 6] 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.

5.4 BlueBee Design

This section illustrates the BlueBee design in detail.

5.4.1 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.

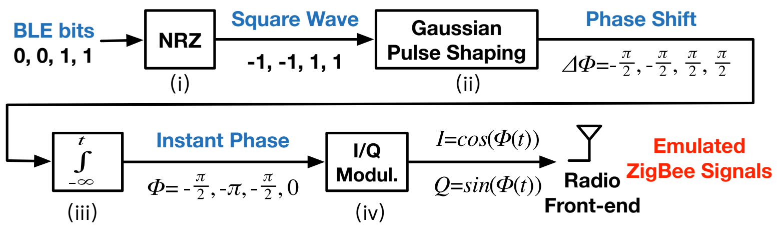


Figure 5.2: BLE as the transmitter with GFSK modulation.

BLE Transmitter. BLE uses Gaussian Frequency Shift Keying (GFSK) modulation, which is normally realized by phase shift over time [1] . Fig. 5.2 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.

ZigBee Receiver. As Fig. 5.3 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.

[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 $s(t) = Acos(2\pi ft \pm \Phi(t))$, where $\Phi(t) = 2\pi\Delta ft$.

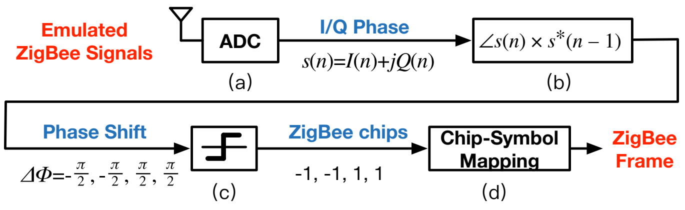


Figure 5.3: ZigBee as the receiver with OQPSK demodulation.

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.

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

Table 5.2: Symbol-to-chip mapping in ZigBee (802.15.4)

Finally, in (d), 32 ZigBee chips are mapped to a ZigBee symbol, by looking up a symbol-to-chip mapping table (Table 5.2) 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 phases may suffer from errors ($+ \leftrightarrow -$), which induce reversed chips ($1 \leftrightarrow 0$). In such case, the closest symbol with smallest Hamming distance is selected.

5.4.2 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 6.1), 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 [34]). 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.

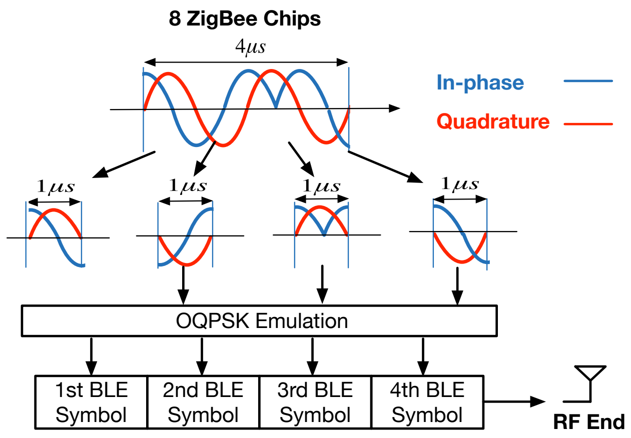
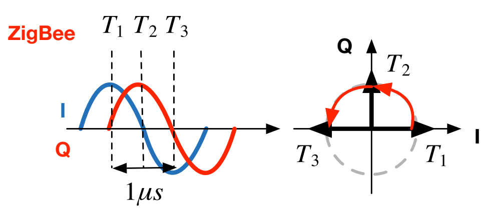


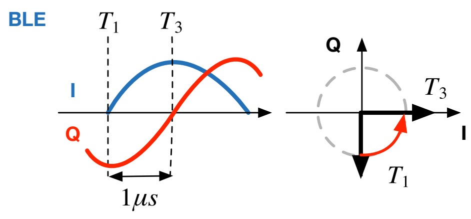
Figure 5.4: Emulating ZigBee with BLE

5.4.3 OQPSK Emulation

Emulating ZigBee's OQPSK modulation with BLE is a nontrivial problem due to the narrower bandwidth of BLE compared to ZigBee (1MHz vs 2MHz). Fig. 5.4 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’



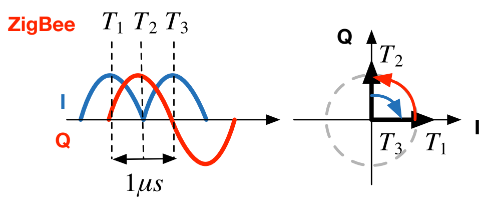
(b) Emulation of (a) by BLE

Figure 5.5: (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.

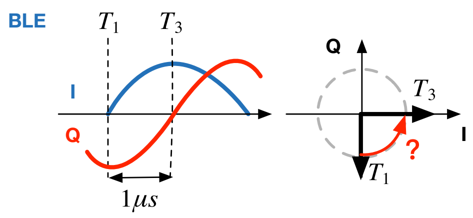
Let us now look into how the emulation is performed on a two-chip piece divided in Fig. 5.4. Recall that OQPSK (i.e., ZigBee) observes phase shifts between consecutive samples, whose signs are translated to chips of -1 and 1 (steps (a) and (b) in Fig. 5.3). The left in Fig. 5.5(a) depicts ZigBee signal (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 see ZigBee signal can be successfully emulated by BLE, which is demonstrated in the left in Fig. 5.5(b). Although the signal appears to be distinct from ZigBee signal (left in Fig. 5.5(a)), 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. 5.5(b) 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. 5.5(b). 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. 5.5(b)) indeed yields the same chip sequence of ‘11’ at the ZigBee receiver, as the ZigBee signal in the left in Fig. 5.5(a). 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. 5.5(b) 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.



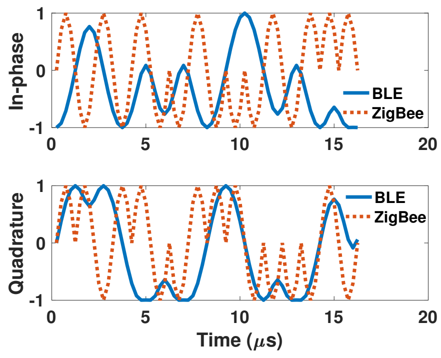
(a) Inconsistent phase shifts at ZigBee



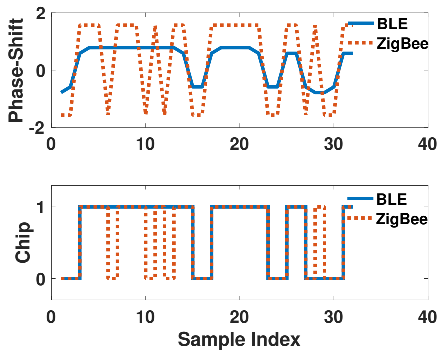
(b) Imperfect signal emulation at BLE

Figure 5.6: Impact of inconsistent ZigBee phase shifts

From the example in Fig. 5.5(b), 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. 5.6(a) demonstrates ZigBee chip sequence of ‘10’. As shown in Fig. 5.6(b) BLE emulates this to be ‘11’ 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 maximize the probability of DSSS to map the received chip sequences to the correct symbol.



(a) Time domain emulated signal



(b) Phase shifts of emulated signal

Figure 5.7: Comparison between BLE emulated signal and the desired ZigBee signal

As a proof of concept example, we emulate a 32-chip ZigBee symbol '0' (i.e., '0000' in Table 6.1) from BLE. In Fig. 5.7(a), 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. 5.7(b) 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. 5.7(b)), 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.

5.4.4 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 5.8 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 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:

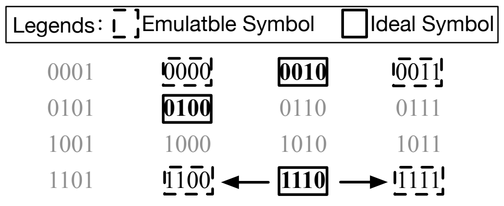


Figure 5.8: An example of optimized emulation

Definition 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 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. 5.8 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.

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. 5.9, 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.

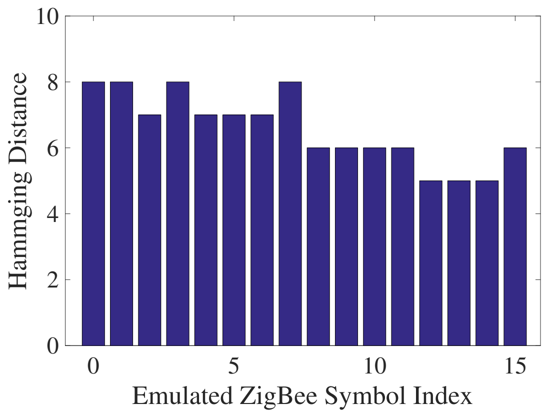


Figure 5.9: Intra symbol Hamming distance between emulated and ideal ZigBee symbols

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:

$$\underset{E_i}{\operatorname{argmax}} \quad min\{Dist(E_i, S_j), i \neq j\} \tag{5.1}$$

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.

5.4.5 Dealing with the BLE Data Whitening

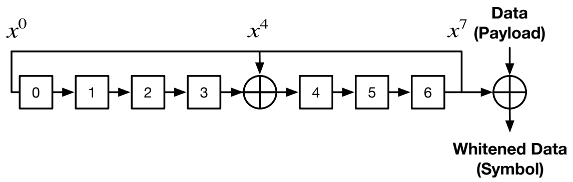


Figure 5.10: BLE data whitening through LFSR

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 [42, 20]; Technically, BLE uses the 7-bit linear feedback shift register (LFSR) circuit with the polynomial $x^7 + x^4 + 1$ as shown in Fig. 6.14. 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. 5.8.

5.5 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.

5.5.1 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.

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

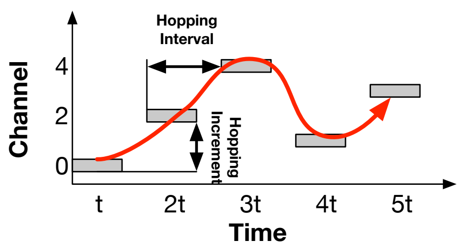


Figure 5.11: BLE normal frequency hopping

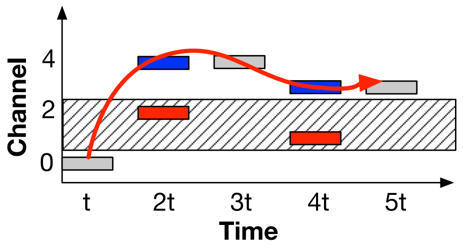


Figure 5.12: BLE adaptive frequency hopping

hopping interval, the BLE device will hop to the next channel with an increment of *hopping increment*. In formula

$$C_{next} = C_{current} + hoppingInc \ (mod 37), \tag{5.2}$$

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. 5.11 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

$$remapIndex = C_{next} \ mod \ |S_{good}|, \tag{5.3}$$

and C_{next} will be replaced by $S_{good}(remapIndex)$. For example, in Fig. 5.12, 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. 5.3.

5.5.2 BlueBee Channel Scheduling

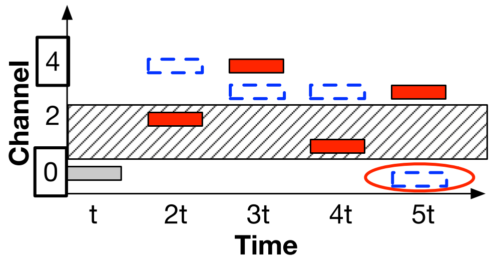
With AFH, the frequency to visit different channels becomes uneven. For example, in Fig. 5.12, channel 0 will only be visited once during one hopping period (i.e., 5 hops in the example) half the frequency of channel 3 and 4. In real network environment, AFH will cause unfair services to ZigBee nodes 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.

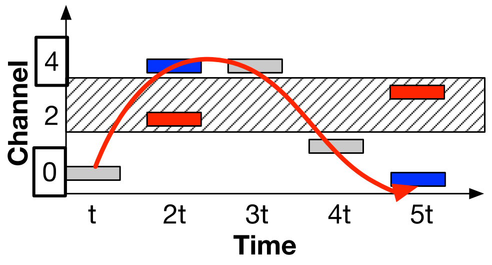
Maximum-throughput solution. By updating the channel bit map, we can control the set of channels BLE device can hop on. To maximize the throughput of concurrent BlueBee, we can leave the ZigBee-BLE overlapped channels in the channel bit map if they are marked as idle in the original channel bit map (i.e., 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 $HCI\_set\_AFH\_Channel\_Classification$ [43].

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 5.13: The steps of BlueBee channel scheduling

In Fig. 5.13(a) and Fig. 5.13(b) 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. 5.13(a) . 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. 5.13(b). 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.

5.6 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.

5.6.1 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 [44]. 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 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.

5.6.2 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. 5.14. 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.

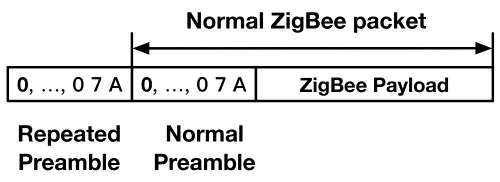


Figure 5.14: Reliable CTC with repeated preamble

5.7 Discussion

5.7.1 Compatibility with Bluetooth Classic

BLE is defined in Bluetooth core specification 4.0 [43]. 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.

5.7.2 Feasibility of Reverse Communication

Although in this chapter, 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.

5.8 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.

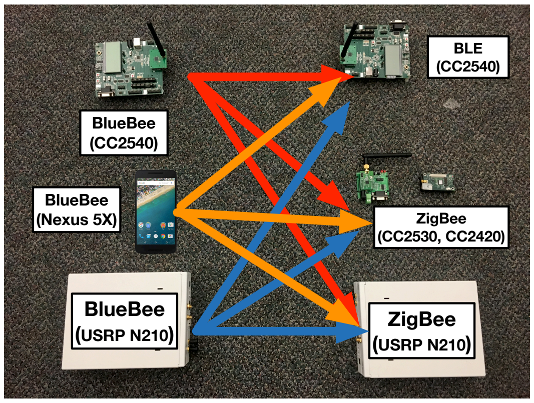


Figure 5.15: Experiment Setting for BlueBee

5.8.1 Platform Setting

Fig. 5.15 demonstrates the evaluation platform of BlueBee. We have implemented BlueBee as a sender on (i) a GNU radio BLE implementation called scapy radio [45] with a USRP-N210 platform, (ii) a commodity BLE CC2540 development kit [46], 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.

5.8.2 CTC Throughput

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

Compare with FreeBee

The only state-of-the-art CTC work on BLE to ZigBee communication is FreeBee [9]. FreeBee's throughput is $17bps$ with a single CTC transmitter, while the throughput of BlueBee is $225kbps$, $13,000\times$ 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.

Compare with Other Packet-Level CTC

Here we compare BlueBee with other state-of-the-art packet-level CTC technologies, including Esense (WiFi $\rightarrow$ ZigBee), and B^2W^2 (BLE $\rightarrow$ WiFi) in throughput. Note that, these CTC techniques have a high-bandwidth radio (i.e., $20MHz$ WiFi radio) either at the sender or at the receiver. From Fig. 5.16, we can see that BlueBee can surpass the state-of-the-art packet-level CTC by $70\times -100\times$. It indicates the intrinsic advantage of PHY-layer CTC over packet-level CTC.

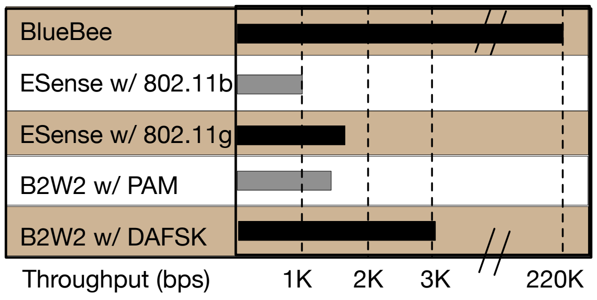
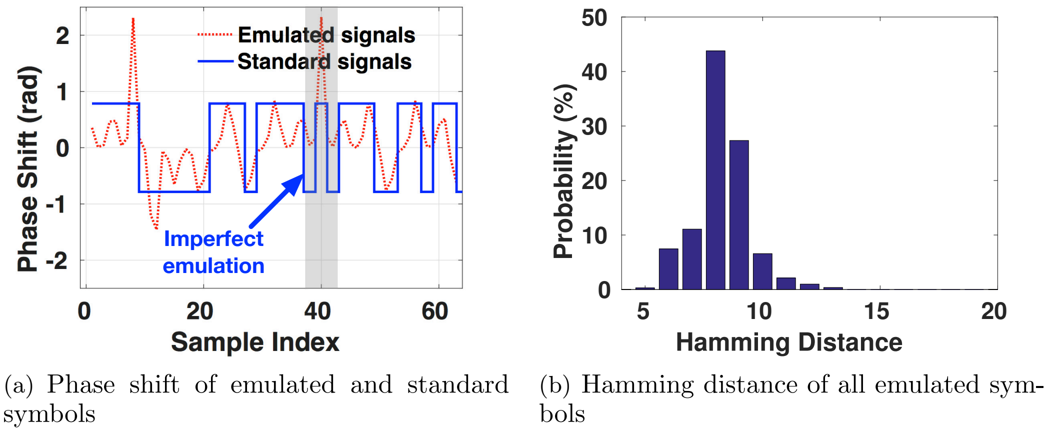


Figure 5.16: Comparison with the state of the arts

5.8.3 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 5.17: Performance of phase shift emulation

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. 5.17(a), 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 oversampling of commodity ZigBee devices. The 64 samples will then be decimated to 32 chips for decoding. In Fig. 5.17(b), 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.

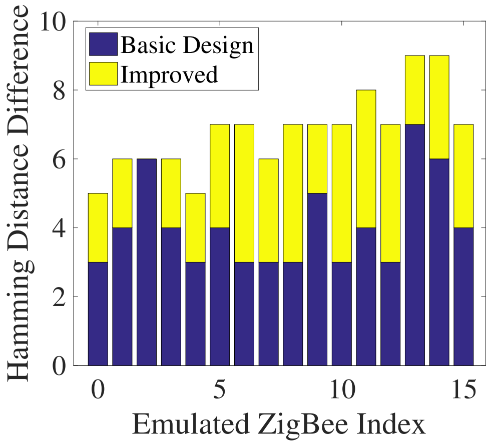


Figure 5.18: Hamming distance improvement of DSSS emulation

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. 5.18, 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 5.18. 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 5.18. This means that BlueBee can tolerate more background noises than the basic design, leading to a better reliability.

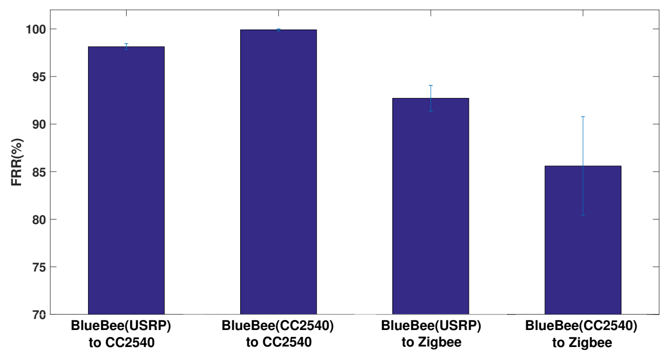


Figure 5.19: FRR comparison under BLE and ZigBee

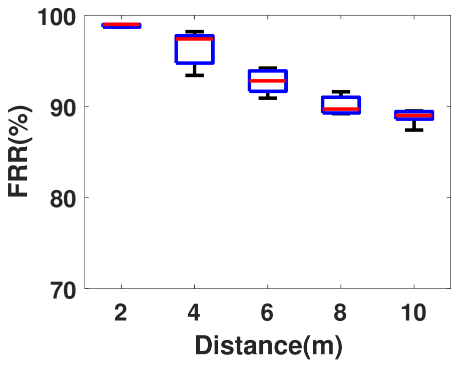
Dual-standard Compliance

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. 5.19, 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.

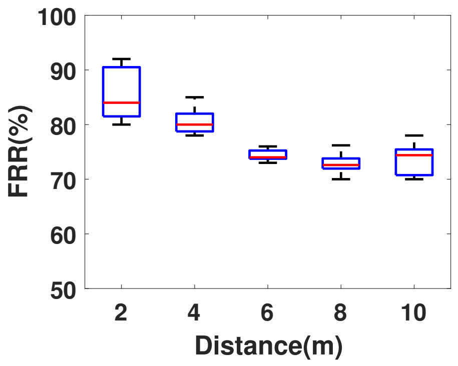
The characteristic of dual-standard compliance indicates BlueBee can achieve *cross-technology broadcast*. That means we can construct a dual-standard frame where part of it is a ZigBee frame and part of it is a BLE frame. Each technology can identify their parts by detecting legitimate preamble and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

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. 5.20(a) depicts the FRR when USRP N210 emulates the ZigBee frames with the transmission power of 0dBm, the maximum energy level allowed in BLE standard [43]. In all



(a) FRR with distance on USRP (lab)



(b) FRR with distance on commodity devices (lab)

Figure 5.20: FRR with distance

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.

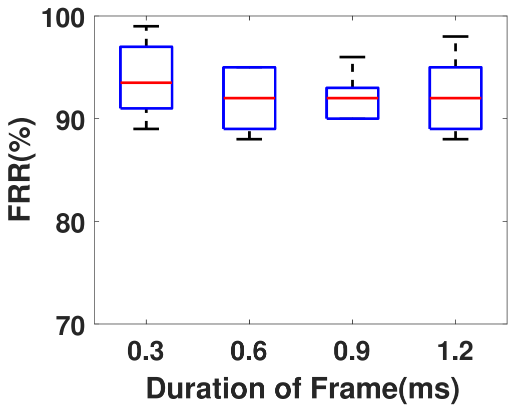


Figure 5.21: FRR with different frame duration

Impact of Frame Duration

In BLE specification 4.2 [43], the maximum payload for BLE has been extended from $39 bytes$ to $257 bytes$, which means the frame duration will grow from around $0.3 ms$ to over $2 ms$. So we here study the impact of frame duration on BlueBee's performance. In Fig. 5.21, we study the FRR with frame duration ranging from $0.3 ms$ to $1.2 ms$, 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 [12]. Even so, over 90% FRR shows BlueBee's resistance to the impact of long frame.

Impact of Tx Power and Tx Distance

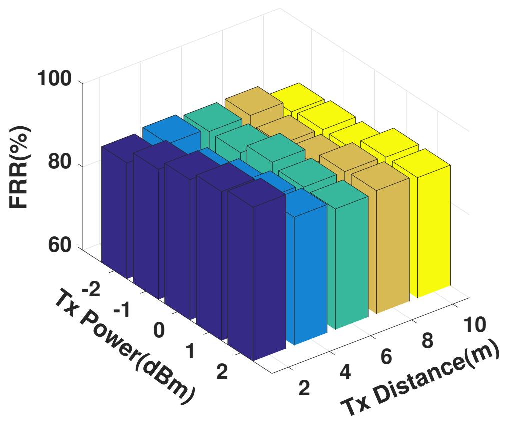


Figure 5.22: FRR with Tx power and distances

In Fig. 5.22 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.

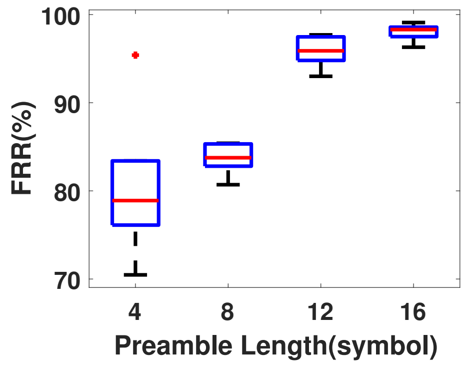


Figure 5.23: FRR with different preamble length out door

Protection in the link layer-multiple header

In Fig. 5.23 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.

5.8.4 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).

Fig. 5.24 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

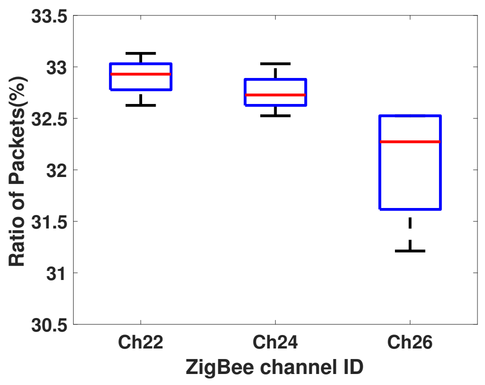


Figure 5.24: Concurrent CTC on three ZigBee channels

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.

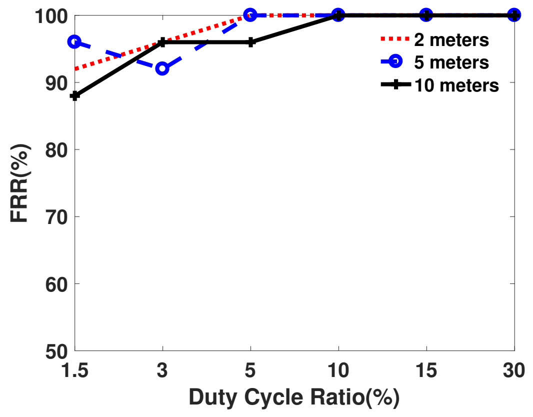


Figure 5.25: BluBee's support for low duty cycle network

5.8.5 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. 5.5.1 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. 5.25, 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.

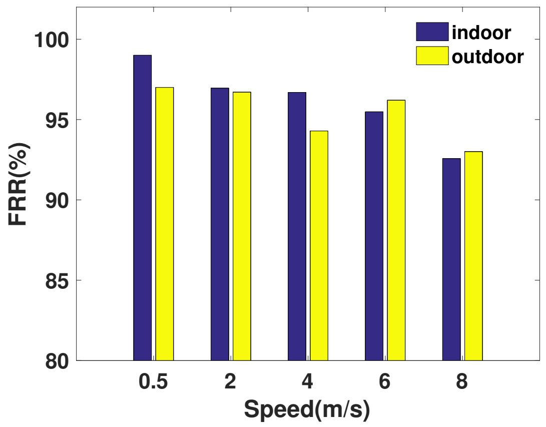


Figure 5.26: BluBee's support for mobile scenario

5.8.6 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. 5.26, 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.

5.8.7 Application

In this section, we will introduce two applications for BlueBee.

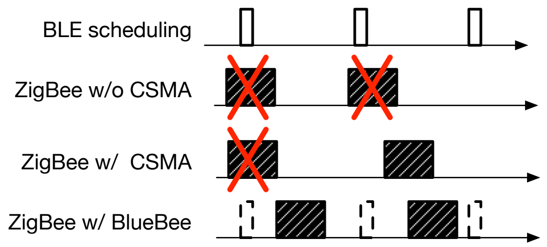


Figure 5.27: Channel coordination between ZigBee and BLE

Application1: 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 we only introduce its channel coordination due to the limitation of space as shown in Fig. 5.27.

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.

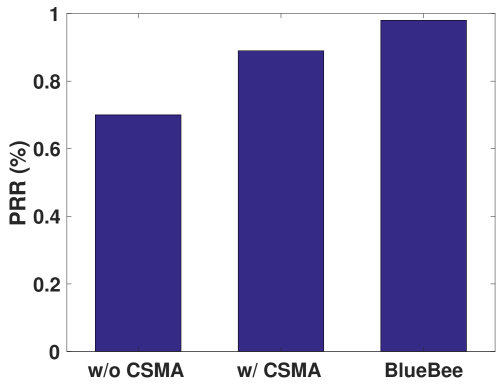


Figure 5.28: PRR under different coordination methods

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.

Fig. 5.28 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.

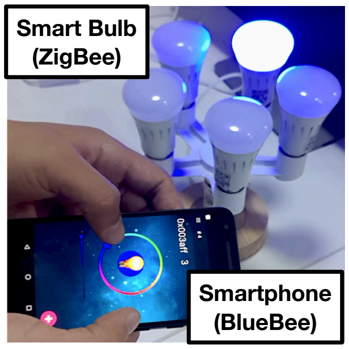


Figure 5.29: BlueBee smart light bulb control

Application2: 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. 5.29, we implement BlueBee on a Nexus 5X smartphone to control ZigBee light bulbs at one of the overlapped channels, i.e., 2.48GHz. Available commands including the on/off status, the color, the intensity, and which light bulb to control. BlueBee achieves direct control of ZigBee devices from a BLE radio without a ZigBee-BLE gateway [47] 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.

5.9 Conclusion

In this chapter we present BlueBee, a new PHY layer cross-technology communication technique proposing a direction of emulating legitimate ZigBee frames using BLE radio. BlueBee paves the road to practical CTC by offering over $10,000\times$ the throughput compared to the state-of-the-art CTC designs that rely on coarse-grained packet-level information. The emulation is achieved simply by selecting the payload bytes of BLE frames to provide unique dual-standard compliance and transparency where neither hardware nor firmware changes are required at the BLE senders and ZigBee receivers. BlueBee includes advanced features such as multi-channel concurrent CTC via adaptive frequency hopping 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.

Chapter 6

Bidirectional Cross-technology Communication

6.1 Introduction

PHY-layer CTC with signal emulation yields vastly increased rate reaching the limits defined by the standards. However, it is applicable mainly when the transmitter is a high-end platform (e.g., WiFi $\rightarrow$ ZigBee in [24]). This is because powerful radios support sophisticated modulations offering higher degrees of freedom in waveform control with greater capability in signal emulation. However, due to the asymmetric nature of CTC in terms of the transmitter and receiver, the reverse communication, i.e., from a low-end transmitter to a high-end receiver, is limited. This calls for a new fundamental technique that effectively enables the PHY-layer CTC in the case of low-end transmitter incapable of signal emulation due to the radio or computational limits.

In this chapter, we propose the first receiver-side CTC, which aims at moving the complexity to the receiver side, as opposed to the transmitter-side CTC with signal emulation. This is inspired by the observation that the bit stream yield by any native demodulator reflects some universal and intrinsic properties of the waveform in the air, such as the amplitude, frequency, or phase. In addition, different modulation techniques are intrinsically related. For example, the frequency-shift keying and phase shift keying are tied because the frequency is the time derivative of phase [48]. Therefore a frequency-shift keying demodulator is able to cross-decode phase-shift keying signal with an upper layer interpreter to recover the phase information from the demodulated bits, which indicates frequency shifts. Specifically, this work proposes XBee, a receiver-side CTC at Bluetooth Low Energy (BLE) for cross-decoding ZigBee packets. At the transmitter side, native ZigBee packets equivalent to homogeneous communication are sent; At the receiver side, the BLE receiver is able to cross-decode every ZigBee symbols upon the bits yielding by the native BLE demodulator. This is particularly challenging to achieve on commercial platforms, which hide physical layer signal via abstraction. Through the adoption of cross-decoding, XBee first achieves PHY-layer CTC from a low-end device to a high-end device. Its data rate is 15,000x higher than the existing packet-level ZigBee to BLE CTCs, and comparable with the state-of-the-art PHY-layer CTCs through signal emulation, enabling CTC bidirectionality.

The specific technical contribution of XBee can be summarized as:

- We design and implement XBee, a PHY-layer CTC uniquely based on cross-decoding at the receiver side. This is achieved solely by a careful examination of the bit patterns which are observable on commercial BLE devices, which enables XBee to operate without any hardware or firmware modification. Most importantly, the transmitter side stays the same as in homogeneous communication.

- Interestingly, XBee *explores* the opportunity within the sampling offset, to overcome the intrinsic uncertainty in cross-decoding. This is counterintuitive as sampling offset is detrimental to decoding, and is compensated in normal communication. XBee also features link layer designs including scheduling protocol that ensures compatibility with the BLE standard as well as non-disruptiveness to other devices in the BLE network.

- We implement XBee for extensive evaluations on its rate, reliability under various environment and parameter setting. Our experiment results have shown that XBee can achieve 250*kbps* with 85% accuracy, increasing the data rate of the existing packet-level CTCs by 15,000x, and comparable to the state-of-the-art PHY-layer CTCs.

6.2 Motivation

Spectrum sharing is becoming even more prevalent with the explosively growing body of wireless devices and standards, as well as expanding open spectrum – from traditional 5GHz, 2.4GHz, and 900MHz bands to 600MHz, TV spectrum, and 7GHz high-frequency which turned unlicensed recently between 2014-2016 [49, 50, 51]. Spectrum crowded with diverse wireless technologies with incompatible physical layers inevitably leads to cross-technology interference (CTI), which becomes one of the root causes of network performance degradation.

To this end, connectivity among heterogeneous technologies is critical to alleviate CTI and, at the same time, explore the potential of cross-technology collaboration. In other words, to draw the full capability of wireless-rich IoT. Recently, researchers propose the cross-technology communication (CTC) aiming at building the direct communication among heterogeneous wireless technologies. Existing CTC works can be categorized as packet-level CTC and the PHY-layer CTC. Packet-level CTC works have intrinsic limitations, while within PHY-layer CTC, all existing works are based on transmitter-side signal emulation.

• **Limitations of Packet-Level CTC.** Earlier set of CTC designs use packet level information, such as the packet duration [6, 7], beacon interval [9], data traffic pattern [52, 41, 53], and energy amplitude [54] to convey messages across technologies. Such approaches, due to the coarse packet-level granularity, have intrinsic limitations in the data rate confined to a few tens bps at the highest.

• **Limitations of Existing PHY-layer CTC.** To overcome these limits, the latest CTC designs [24, 55] utilize fine-grained physical layer information for high-speed CTC that approach the maximum rate defined by the standard. Technically, PHY-layer CTC introduced until now are based on transmitter-side signal emulation, where the transmitter approximates the target waveform by exploring the signal degree of freedom offered by the transmitter's modulator. For example in [24], a WiFi to ZigBee PHY-layer CTC is proposed. The WiFi transmitter carefully selects payload where the corresponding OFMD QAM constellation points approximate that of the ZigBee's OQPSK signal so that the emulated signal can be demodulated by the ZigBee receiver with its native demodulator.

Despite PHY-layer CTC's significant advancement in the data rate, the technique of signal emulation – which the current PHY-layer CTC designs are commonly dependent upon – applicable only for a higher-end transmitter to a lower-end receiver scenario. This is because sophisticated radios in higher-end systems offer higher degrees of freedom in modulation. In other words, they support assembling (i.e., modulating) complicated signals (e.g.., OFDM QAM in WiFi), and therefore are more capable of emulating simpler signals desired at lower-end receivers (e.g., OQPSK in ZigBee). Due to this technical reason and the asymmetric nature of CTC, PHY-layer CTC in reverse direction, from a lower-end transmitter to a higher-end receiver, is difficult to achieve through the known technique (i.e., signal emulation) and remains an open issue.

• **The Need for Receiver-Side Cross-decoding.** This work is motivated by the fundamental but missing piece – enabling PHY-layer CTC from lower-end transmitted to higher-end receiver – which is the key technique to achieve bidirectional communication in PHY-layer CTC. This is achieved by enabling cross-decoding of the native transmitter packet at the receiver. For example, a commercial BLE device (receiver) runs a mechanism that interprets the message within an unmodified ZigBee (transmitter) packet. In other words, cross-decoding is pushing the complexity to the receiver side (conversely to the signal emulation) which is the higher-end. Along with the previous designs, this is the key to accomplishing PHY-layer CTC in all directions; thus bringing true ubiquitous connectivity among heterogeneous wireless systems and further, enabling cross-technology channel negotiations and advanced collaborations in practice.

To make our description specific, in the chapter, we focus on the cross-decoding of ZigBee packets at a BLE receiver, which is a missing piece left by existing transmitter-side signal emulation [55]. The CTC between ZigBee and Bluetooth, the two most popular technologies in IoT will trigger a lot of interesting applications as illustrated in Fig. 6.1, such as (a) In the gym, workout equipment attached with ZigBee radio can communicate with the wearable Bluetooth devices to make customized workout plan; (b) In the smart home, smart devices with ZigBee radio are able to associate to the Bluetooth speaker to play essential messages; (c) In the factory, ZigBee sensors can notify a Bluetooth camera to monitor the pipeline when abnormal events are detected; (d) In the indoor navigation, the ZigBee landmarks can help smartphones achieve fine-grained navigation.

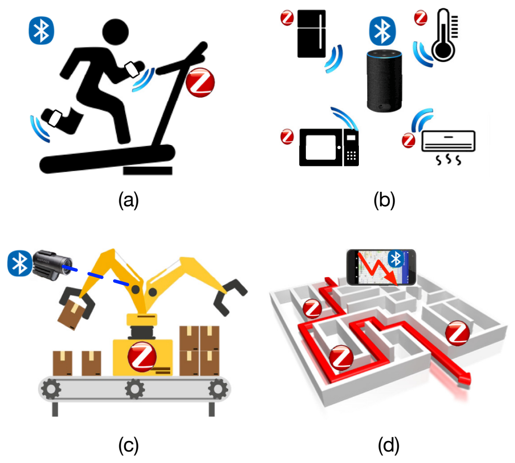


Figure 6.1: IoT applications with the cooperation of heterogeneous wireless technologies.

6.3 XBee In a Nutshell

• **Overview**. XBee is a PHY-layer CTC supporting CTC message from ZigBee to BLE with receiver-side cross-decoding. By pushing the complexity to the receiver side, i.e., the BLE side, XBee supports transmitting CTC messages in native ZigBee symbols. The BLE receiver simply demodulates all the input ZigBee signal into BLE bits. Then a cross-decoding module will interpret the demodulated bits into original ZigBee symbols to recover the CTC message. To trigger the cross-decoding as well as specify the receiver, a specific ZigBee symbol sequence is chosen to work as a BLE receiver ID, as illustrated in Fig. 6.2, which will match a manipulated BLE access address at the receiver side.

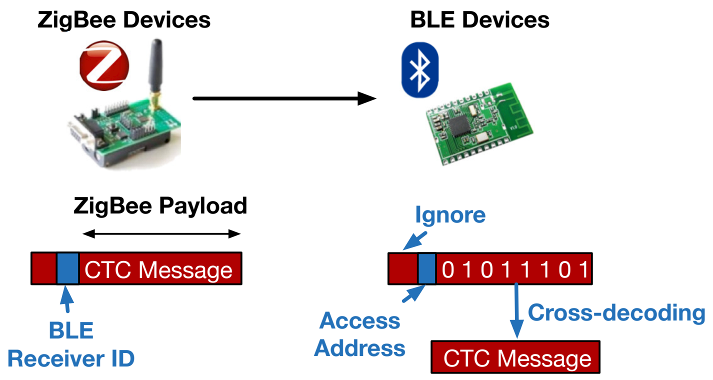


Figure 6.2: The system architecture of XBee.

• **Unique features** XBee is the first PHY-layer CTC based on receiver-side cross-decoding. By pushing the complexity to the receiver, it covers the essential but missing piece left by the state-of-the-art PHY-layer CTC based on transmitter-side signal emulation, paving the way to accomplish PHY-layer CTC in all directions. In addition, XBee is friendly to the transmitter, for the messages are in native ZigBee symbols. Finally, XBee needs no hardware or firmware modification at either the transmitter or receiver, making it easy to deploy onto millions of existing ZigBee and BLE devices.

6.4 XBee Design

6.4.1 Background

We first introduce the background of ZigBee transmitter and the BLE receiver, followed by the details of XBee design.

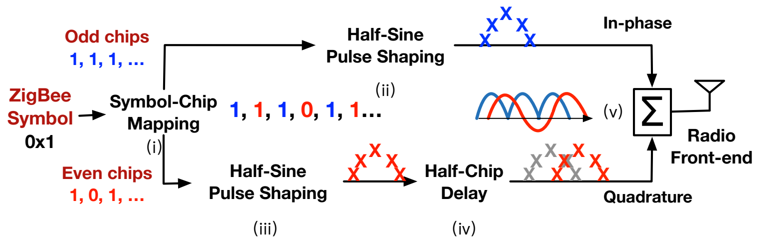


Figure 6.3: ZigBee as the transmitter.

Symbol (4 bits)	Chip Sequence (32 bits)
0x0	11011001110000110101001000101110
0x1	11101101100111000011010100100010
...	...
0xF	11001001011000000111011110111000

Table 6.1: Symbol-to-chip mapping in ZigBee (802.15.4)

• **ZigBee Transmitter** ZigBee adopts direct sequence spread spectrum (DSSS) and offset quadrature phase-shift keying (OQPSK) in its modulation. In Fig. 6.3, we illustrate the whole procedure from a ZigBee symbol to the transmitted I/Q signal in the air from step (i) to (v). A ZigBee symbol is the minimum unit of a ZigBee frame. Each ZigBee symbol contains 4-bit information, i.e., from symbol '0x0' to symbol '0xF'. In the PHY layer, a ZigBee symbol first goes through DSSS, where each ZigBee symbol will be extended to 32 chips according to the symbol-chip mapping table, i.e., Table 6.1, in the IEEE 802.15.4 standard, as illustrated in step (i). Then the 32 chips will go through the OQPSK modulation, where the odd chips are allocated to the in-phase and the even chips are allocated to the quadrature. Both the in-phase and quadrature chip sequences will go through a half-sine pulse shaping module, as illustrated in step (ii) and (iii), to shape the chips to a sinusoidal wave. What unique in OQPSK is that the quadrature chip sequence will further have a half-chip delay, as illustrated in step (iv). Finally, the in-phase and quadrature signals are merged and transmitted to the air.

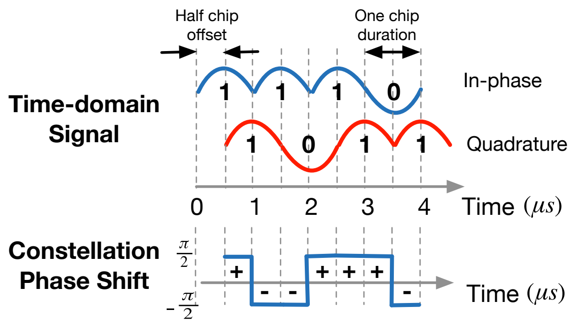


Figure 6.4: Time-domain signal and phase shift of ZigBee symbol ‘1’.

The transmitted ZigBee signal shows particular property in the constellation. In Fig. 6.4, the time-domain signal of the first 8 chips of ZigBee symbol ‘1’ is plotted. Each ZigBee chip lasts $1\mu s$. In the constellation, the change in phase between consecutive samples, referred to as the *phase shift*, is calculated. Due to the half-chip offset in OQPSK, the phase shifts will only take two values, $\frac{\pi}{2}$ or $-\frac{\pi}{2}$, representing positive or negative phase shifts.

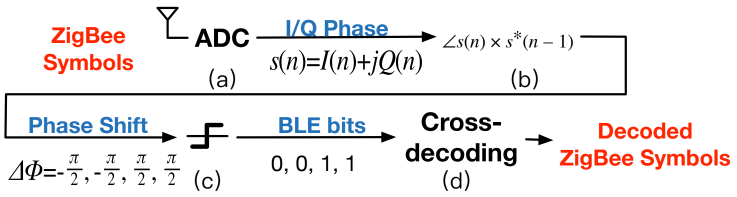


Figure 6.5: BLE as the receiver.

• **BLE receiver** In the PHY layer, BLE adopts the Gaussian Frequency Shift Keying (GFSK), where each BLE bit indicates either a positive or a negative frequency shift. As illustrated in Fig. 6.5, at the demodulator, BLE adopts the quadrature demodulator to detect frequency shifts through the phase shifts [1] [56]. More specifically, a BLE receiver first samples the channel, i.e., gets the complex I/Q samples $s(n) = I(n) + jQ(n)$, in step (a) and feeds to its demodulator. The demodulator calculates the change in phase, i.e., the *phase shifts*, between consecutive I/Q samples to figure out the signal frequency shift. More specifically, the BLE demodulator uses the formula $arctan(s(n) \times s^*(n-1))$, where $s^*(n-1)$ is the conjugate of $s(n-1)$ in step (b). In step (c), the phase shifts are quantized to be BLE bit 0 or 1 according to the *sign* of phase shifts to be negative or positive. Finally, the cross-decoding block, illustrated in step (d), interprets the BLE bits yielding from the native demodulator to ZigBee symbols. Since each BLE bit lasts $1\mu s$, while each ZigBee symbol lasts $16\mu s$, 16 BLE bits are interpreted as one ZigBee symbol.

6.4.2 Opportunities and Challenges

Cross-decoding is feasible due to the following two technical insights. First, the phase shift is the intrinsic feature of phase modulated signal such as ZigBee signal and is also used at the BLE receiver to figure out signal frequency shift. Second, at the BLE receiver, the phase shifts are quantized so that only the *sign* (+/-) of the phase shifts matters, which brings a lot of freedom in cross-decoding.

However, the challenges come from the fact that the BLE bandwidth is 1MHz, only half that of ZigBee. The low bandwidth is corresponding to the low sample rate, or equivalently the larger sample interval according to the Nyquist theorem. As a result, BLE receiver is not able to get full ZigBee symbol information from sampling. Later on, we will see this makes one ZigBee symbol corresponds to multiple possible BLE bit sequences at the BLE receiver. How to figure out and deal with the uncertainty are challenging issues in cross-decoding.

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

6.4.3 Cross-decoding

XBee's core technique of cross-decoding interprets ZigBee packet only from the bit patterns obtained at the BLE receiver, making the design fully compatible with commercial devices. To achieve this, we first offer insights on BLE output bits when it is fed with different ZigBee signals, which can be inversely applied to derive ZigBee chips (and thus symbols) – i.e., cross-decoding. The limited bandwidth of BLE (1MHz) compared to ZigBee (2MHz) makes BLE bits only partially reflect ZigBee signal.

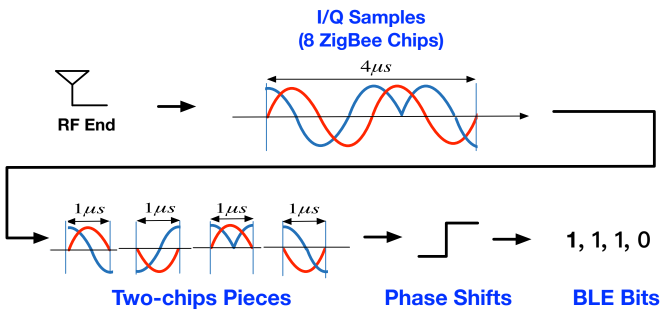


Figure 6.6: BLE receiver yields a bit for every $1us$ corresponding to two ZigBee chips.

We illustrate cross-decoding with a walk-through example in Fig. 6.6 with 8 ZigBee chips lasting $4\mu s$. At the BLE receiver, ZigBee chips are first cut into four $1\mu s$ pieces. We refer to each piece as the *two-chip piece* for its containing the phase shift information of two ZigBee chips. A two-chip piece will finally be demodulated as a single BLE bit '1' or '0' according to the accumulated phase shift.

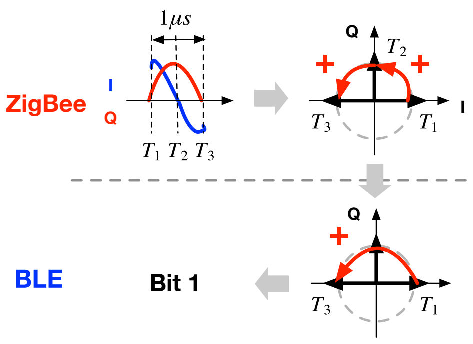


Figure 6.7: BLE bit when phase shifts in two chips are consistent. Phase shift of '++' yields bit '1' at the BLE receiver. Likewise, '−−' yields '0'.

We now take a closer look at the demodulation of the two-chip pieces. We first study the case when the two chips incur consecutive two positive (++) phase shifts, as shown in Fig. 6.7. In the figure, ZigBee has a $\frac{\pi}{2}$ phase shift every $0.5us$, from T_1 to T_2 and from T_2 to T_3. The BLE receiver, however, with its bandwidth limited to 1MHz, samples at only T_1 and T_3. The accumulated phase shift from T_1 to T_3 is positive (i.e., π), so BLE outputs bit '1'. Similarly, BLE conveys bit '0' upon two chips with both negative ($--$) phase shifts. This demonstrates the clear relationship between consistent two-chip pieces and the BLE bits, which, however, does not hold when phase shifts due to the two chips are inconsistent (i.e., '+-' or '-+').

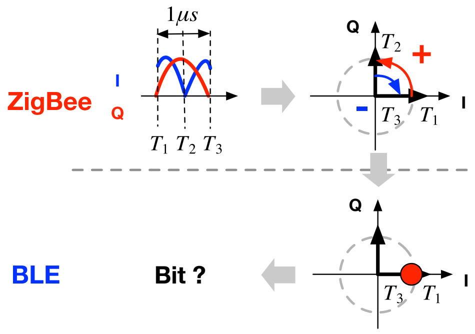


Figure 6.8: Inconsistent phase shifts leave BLE bit undetermined.

• **The impact of inconsistent phase shift:** We illustrate the inconsistent phase shift case, i.e., one positive and one negative, in Fig. 6.8. In the figure, the phase shifts from T_1 to T_2 and T_2 to T_3 are different in the constellation, which makes the overall phase shift 0 theoretically. In this case, we are not able to determine the final BLE bit, which we refer to as the *undetermined bits*. However, in a practical system, the phase shift will not stay at 0, factors such as the unsynchronized transmitter and receiver or the signal distortion in the air, will break the balance and make the final phase shift prone to either the positive or the negative side.

As a proof of concept, in Fig. 6.9, we use USRP BLE receiver to demodulate over 1000 random ZigBee symbols, and outputs the phase shift of each two-chip piece before interpreting them into BLE bits. Then we plot the distribution of the phase shift of the four possible groups of the ZigBee two-chip pieces, according to the combinations of '+' and '-', i.e., '++', '+-', '-+', and '- -'. From the figure we can see, almost all the phase shifts of the '++' and '- -' groups are positive and negative respectively, which will

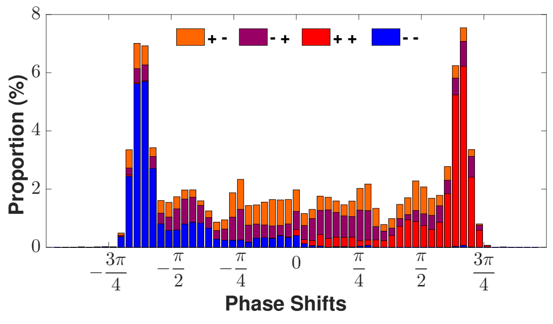


Figure 6.9: Phase shift distribution of all four groups of two-chips pieces.

be uniquely demodulated as BLE bits ‘1’ and ‘0’ respectively. The other two groups cannot be uniquely demodulated, because they distribute on both the positive and negative sides. In addition, the phase shifts of the group ‘++’ and ‘- -’ are mostly over $\pi/2$ or less than $-\pi/2$, while that of the other two groups accumulate near $\pm\pi/4$. That is because consistent phase shifts (i.e., ‘++’ and ‘- -’) will sum up, while inconsistent phase shifts (i.e., ’+-’ and ’-+’) will cancel out each other.

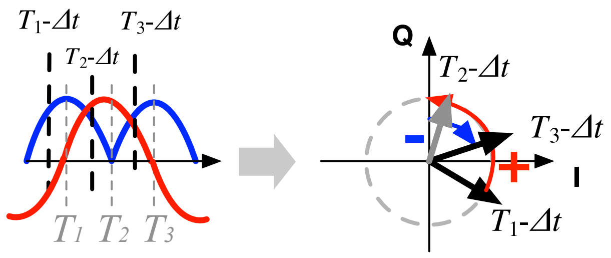
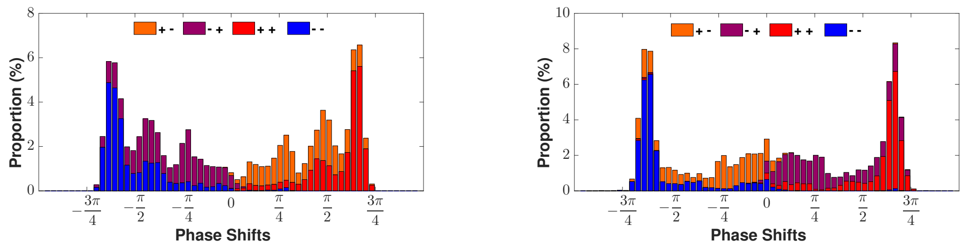


Figure 6.10: Sample offset affects BLE bit sequence.

• **Exploring the sample offset:** The actual output of undetermined bits largely depends on the sample offset between the transmitter and receiver, when they are not well synchronized. The sample offset may become prone to either the left or the right of the two-chips piece boundary. Without loss of generality, in Fig. 6.10, we illustrate the case when the samples have a left offset from the boundary T_1, T_2, and T_3 by Δt. In the constellation, the unsynchronized samples extend the duration of positive phase shifts, i.e., the red arrow from $T_1 - \Delta t$ to T_2, and shorten that of the negative phase shift, i.e., the blue arrow from T_2 to $T_3 - \Delta t$. So the positive phase shift dominates and makes the two-chip piece demodulated as BLE bit ‘1’. How about the other undetermined two-chip pieces? We do not need to analyze them one by one. Due to the relatively stable sample interval during a short time, i.e., within a ZigBee packet, all the samples share the same sample offset and can be derived accordingly. The stable sample interval also greatly reduces the computation cost. For n undetermined bits, we do not need to calculate 2^n variations, but only 2, either all the samples have a left offset or all have a right offset.



(a) Left sample offset

(b) Right sample offset

Figure 6.11: Distribution of the four groups of two-chips pieces with different sample offset

As a proof of concept, in Fig. 6.11(a) and 6.11(b), we illustrate the impact on phase shift distribution given the knowledge of sample offset. Here use the same data as in Fig. 6.9, but we first separate the data based on the sample offset of a packet to be left (Fig. 6.11(a)) or right (Fig. 6.11(b)). We can see the ZigBee two-chip pieces groups '+ -' and '- +' can now be uniquely demodulated. More specifically, the overall phase shift is positive if a two-chip piece '+ -' offset to the left and negative if it offsets to the right. Vice versa for the '- +' case. The results back up our finding that it is the sample offset that affects the actual output of undetermined two-chip pieces.

• **Cross-decoding at Commodity BLE Receiver:** Above discussion showed how a ZigBee symbol may yield two different bit sequences depending on (left or right) sample offset at the BLE receiver. In this part of the section, we illustrate how the original ZigBee symbols are recovered (i.e., cross-decoded) from the two-bit sequences. We initiate the step by step description with an example in Fig. 6.12; the determined bits (from consistent phase shift) are marked as grays boxes, where they remain the same regardless of the left/right offset. Conversely, undetermined bits (from inconsistent phase shift), indicated as white boxes, depends on sample offset. In other words, the

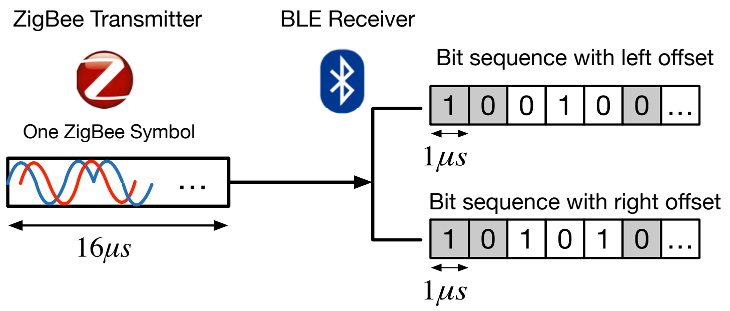


Figure 6.12: BLE receiver yields a two different bit sequences for any ZigBee symbol due to right/left sample offset. The two sequences share bits determined bits (in gray boxes), where the bits are flipped for undetermined bits (in white boxes).

two-bit sequences yield from left/right sample offset share the same bits in gray boxes, where those in white boxes are flipped (0↔1). Recall that there are 16 ZigBee symbols; since each symbol may yield two-bit sequences at the BLE receiver, cross-decoding turns out to be simply mapping each of the 32 possible bit sequences to a most likely ZigBee symbol. This is formulated as two matrices that map bit sequences to the corresponding ZigBee symbols, where one matrix holds the mapping for 16-bit sequences under left sample offset while the other holds those under right sample offset.

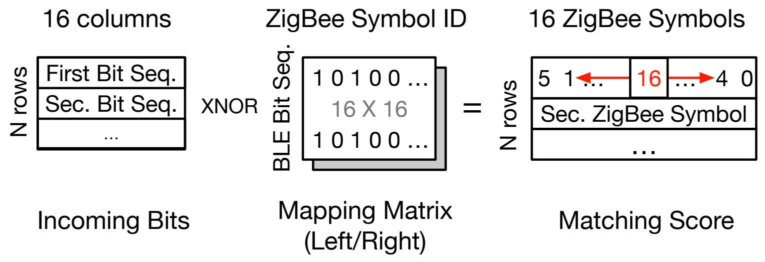


Figure 6.13: Cross-decode ZigBee symbols through BLE bit mapping.

Fig. 6.13 illustrates the process of cross-decoding via matrix mapping. The bit sequence output at the commercial BLE receiver is cut into small chunks of length 16 to match the duration of a ZigBee symbol (i.e., 16us). As the received bits are naturally prone to error in practice, each 16 bit sequence is compared with the ideal bit sequence corresponding to the ZigBee symbols under left/right sample offset – in other words, each column of the left/right mapping matrix. Specifically, this is computed by XNOR between the input bit sequence and the ideal bit sequence, which yields the number of bit match between the two. The ZigBee symbol that corresponds to the ideal bit sequence with the maximum bit match is the result of cross-decoding. From cross-decoded ZigBee symbols, XBee extracts the message including frame header, payload, and FCS. We note that mapping matrices are constant as therefore XBee computationally lightweight, enabling non-disruptive operation under low-end BLE devices. Specifically, the computational cost is only $O(n)$, where n is the number of input bits. We measure the XBee energy consumption in Sec. 6.7.5.

To summarize, XBee cross-decodes ZigBee symbols directly from the bits retrieved from a commercial BLE device. By smartly leveraging sampling offset, an unavoidable phenomenon in practice, XBee effectively recovers original ZigBee symbols under very light-weight computation. That is, surprisingly, XBee successfully receive 2MHz-wide ZigBee signal using BLE hardware constrained to 1MHz bandwidth.

6.5 How XBee works

In this section, we will discuss how to apply the cross-decoding technique at the BLE receiver along with all the BLE protocol constraints such as packet detection, data whitening, and scheduling.

6.5.1 XBee Packet Detection

According to the BLE frame format, a 32-bit access address is attached ahead of any BLE frame, which is used to identify the BLE frame in a specific BLE connection. Its value can be determined by the user and shared between the transmitter and receiver in the association process.

In XBee, for each ZigBee-to-BLE connection, we assign a unique receiver ID at the ZigBee transmitter before any CTC message, as illustrated in Fig. 6.2. This receiver ID is in native ZigBee symbol, and lasts 2-symbol long, corresponding to 32 BLE bits. At the BLE receiver side, we use the 32 BLE bit sequence corresponding to the 'receiver ID' as the access address. Then the ZigBee packet can be recognized by the BLE receiver as a valid BLE packet and go further into the demodulation and cross-decoding.

In addition, as we mentioned earlier, a receiver ID in ZigBee symbols will correspond to two BLE bit sequences depending on the sample offset. In the access address detection, we allow bit sequences matching either access addresses due to different sample offsets to pass. By looking at which access address matches the receiver ID, we are able to figure out whether there exists a left offset or right offset between the transmitter and receiver and choose the corresponding mapping matrix for cross-decoding.

6.5.2 Reverse Data Whitening

Till now, we have assumed that we have direct access to the BLE bit stream from the native demodulator for the simplicity of description. However, in real BLE platform, we only have access to the payload bytes. Between the raw BLE bytes and the payload bytes, for the security issue, there is a scrambler layer, known as the data whitening. The data whitening process in BLE is through a linear-feedback shift register (LFSR) shown in Fig. 6.14. More specifically, the LFSR circuit will output a scramble seed which is used 'whiten' the received bytes by doing the XOR operation with them. The scramble seed is initialized as the channel number (i.e., from 0 to 39) and change iteratively after each byte through the formula x^7+x^4+1 as shown in Fig. 6.14. To recover the raw BLE bytes from the payload bytes, we need to generate the same sequence of BLE scramble seeds and XOR them byte by byte with the BLE payload bytes.

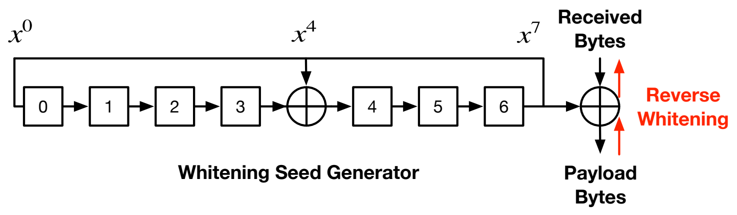


Figure 6.14: Reverse BLE data whitening

6.5.3 XBee Scheduling

Two specific difference between ZigBee and BLE network are that BLE devices need association and will do frequency hopping. To break the barrier between ZigBee and BLE, we need to design the MAC layer association and channel scheduling methods.

We start with a brief introduction of how BLE association and frequency hopping works. The 40 BLE channels, i.e., channel 0 to 39, are classified as the 3 advertising channels, i.e., 37, 38, 39, and 37 data channels. Before communication, two BLE devices must first associate at one of the three advertising channels. More specifically, one BLE device, referred to as the BLE slave device, will keep broadcasting its availability for connection through an advertising packet. Another device, referred to as the BLE master device, will choose to connect to the BLE slave device by replying 'request connection' message and necessary connection parameters, such as the channel hopping increment in frequency and the hopping interval in time. Then they will exchange packets following the associated channel hopping schedule as illustrated in Fig. 6.15.

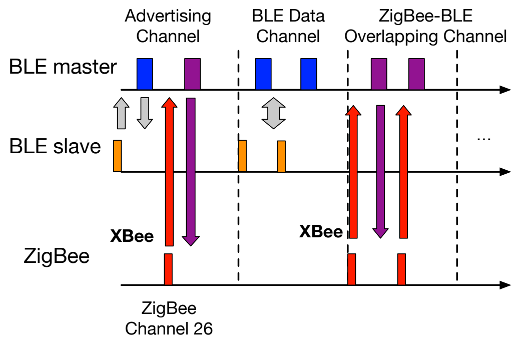


Figure 6.15: ZigBee devices in BLE network.

Similarly, in XBee, the association protocol between a ZigBee device and the BLE network is designed so that the ZigBee device can connect to existing BLE network. A ZigBee device will start by broadcasting its availability for connection with a specific receiver ID at ZigBee channel 26, which is also BLE broadcasting channel 39. If the BLE master device is willing to connect to a ZigBee device, it will listen for a specific access address corresponding to the ZigBee receiver ID, besides the normal access address for BLE advertising packets. After receiving the request from the ZigBee device, the BLE master device will use signal emulation technique [55] to reply necessary connection information, such as the hopping increment and hopping interval in a ZigBee compliant frame. The ZigBee device can either choose to follow BLE's hopping schedule on the ZigBee-BLE overlapping channels if it supports frequency hopping, e.g., the WirelessHart ZigBee protocol, or stay on one overlapping channel and wait for the arrival of the BLE device at most after 37 hops [57].

The join of a ZigBee device in the BLE network will not disrupt normal BLE connection. That is because the schedule of BLE communication is in a slotted manner. In other words, all associated devices to a BLE master devices will be given separate time slots, so that the BLE master device can communicate with normal BLE devices while keeping the connection with the ZigBee device through CTC. Thus we have achieved a hybrid network with both ZigBee and BLE devices working harmoniously.

6.6 Discussion

6.6.1 Receiver ID Protection

Recall that the ZigBee transmitter attaches a receiver ID ahead of any CTC message, which will be recognized as the BLE access address. The successful detection of the receiver ID is critical to BLE cross-decoding, because otherwise the whole packet will be regarded as noise and discarded. To protect the receiver ID, we simply repeat it multiple times. For example, in Fig. 6.27, we repeat the receiver ID two times, and the BLE receiver is able to identify the ZigBee packet if it successfully detects any of the receiver IDs. The cross-decoding will start after the last repeat of receiver ID.

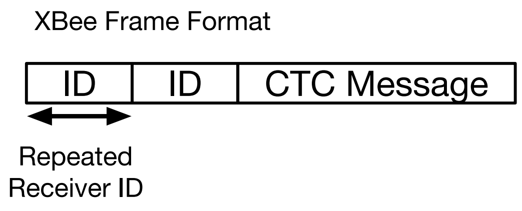


Figure 6.16: Reliable XBee with repeated receiver ID

6.6.2 Receiver Oversampling

In XBee, one challenge is the low bandwidth and the corresponding low sample rate at the BLE receiver side. However, we note that in commodity devices the receiver may oversample the channel, e.g., sample at $2MHz$ instead of $1MHz$, to make the system more robust. Even in that case, our arguments still satisfy, because the BLE demodulator will only yield bit stream at $1MHz$ no matter how fast it samples at the PHY layer, which can not fully represent the phase shift information in the ZigBee two-chips pieces.

6.7 Evaluation

In this section, we compare the performance of XBee with the state of the art and evaluate its performance under various settings.

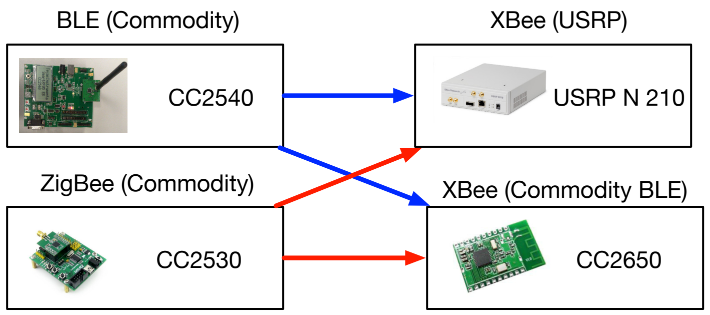


Figure 6.17: Experiment setting for XBee

6.7.1 Platform Setting

Fig. 6.17 demonstrates platforms for XBee evaluation. We have implemented XBee on USRP N210 with BLE PHY, as well as on commercial off-the-shelf (COTS) BLE CC2650 evaluation board. We also use COTS ZigBee and BLE with CC2530 and CC2540 boards, respectively, as transmitters. All experiments are repeated multiple times for statistical results.

6.7.2 Data Rate

We first evaluate the data rate of XBee in comparison to the standard ZigBee and state-of-the-art CTC techniques. The study spans both packet-leve CTC designs [9, 52, 41] and more recent PHY-layer CTCs [24, 55].

• **Versus packet-level CTC:** We first compare XBee's data rate with three packet level CTCs, i.e., FreeBee[9], DCTC [52], and C-Morse [41]. As shown in Fig. 6.18, the data rate of FreeBee with 14bps where DCTC and C-Morse have rates of 190bps and 215bps. Such designs use coarse-grained packet-level information (e.g., packet timing, patterns, etc.) and thus are intrinsically limited in performance. XBee, by directly utilizing the physical layer information, reaches significantly higher rate. XBee on USRP and commodity chip can achieve a data rate of 212$kbps$ and 217$kbps$ respectively, which

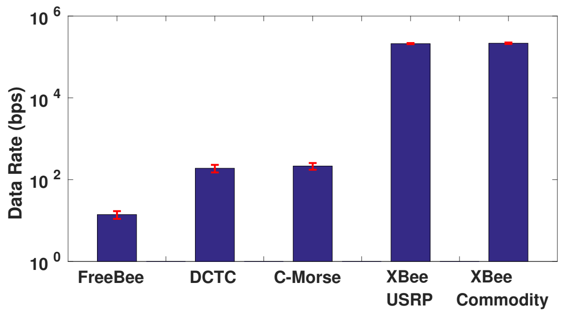


Figure 6.18: XBee data rate compared with three state-of-the-art packet level CTCs

outperforms FreeBee by over 15,000×, and DCTC and C-Morse by over three orders of magnitude. This demonstrates the practicality of XBee over packet-level competitors.

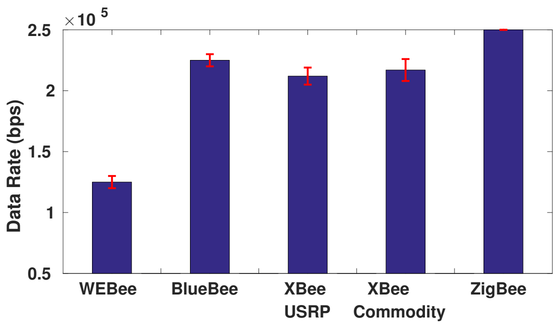


Figure 6.19: XBee data rate VS. state-of-the-art PHY-layer CTCs

• **Versus state-of-the-art PHY-layer CTC:** The recent advances in CTC introduced PHY-layer designs based on signal emulation, namely WEBee and BlueBee [24, 55]. Work along this line commonly leverage the transmitter's high degree of freedom in signal manipulation, to generate waveform closely follows that of the receiver technology. XBee is a new PHY-layer CTC taking the unique approach of cross-decoding, which, by transferring the complexity to the receiver side, enables CTC under transmitter with a limited degree of freedom (i.e., low-end RF). We compare the performance of XBee to the state-of-the-art PHY-layer CTC designs by measuring how closely they approach the ZigBee standard data rate of 250Kbps. Fig. 6.19 shows that WEBee and BlueBee achieve 125$kbps$ and 225$kbps$, respectively. XBee, by reaching 217$kbps$, outperforms WEBee by 1.7× and is comparable to BlueBee. In contrast to XBee, BlueBee's receiver's bandwidth (2MHz ZigBee) is wider than that of the sender

(1MHz BLE). BlueBee benefits from this to retrieve more fine-grained phase information, thus reaching slightly higher performance. The result validates that XBee, by only utilizing receiver-side technique (i.e., cross-decoding), achieves performance similar to the state-of-the-art PHY-layer CTCs that rely on sophisticated transmitter-side signal processing. This is an indication that XBee successfully fills in the gap towards CTC bidirectionality accompanied with known PHY-layer CTCs, as experimentally evaluated in Sec. 6.7.6.

6.7.3 Symbol Error Rate

Here we evaluate the ZigBee symbol error rate (SER) via cross-decoding.

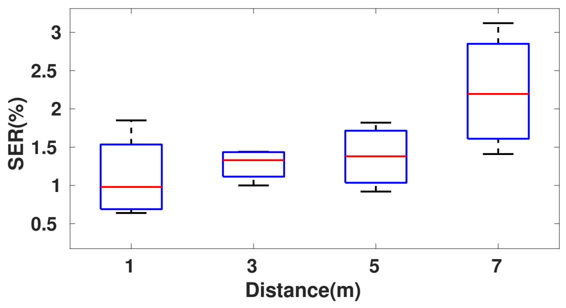


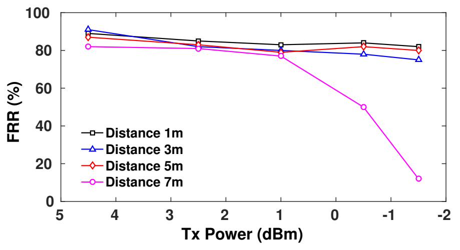
Figure 6.20: SER with different distance.

In Fig. 6.20, we study the ZigBee SER when a 4.3dBm commodity ZigBee Tx is put $[1m, 7m]$ away from a USRP BLE receiver. We find the average SER is about 1% at $1m$ and gradually increases to about 2.2% at 7m. That's because BLE signal is usually low in Tx power, and will quickly attenuate in the air.

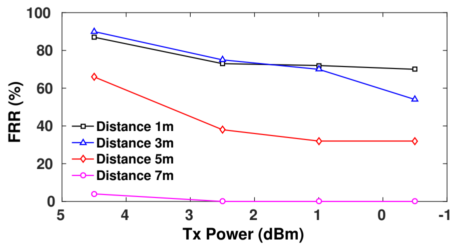
6.7.4 Frame Reception Ratio

In this part of the section, we study the impacts of the factors that affect the frame reception ratio (FRR) of XBee.

• **Impact of Distance & Tx Power** Like any other communication systems, XBee's performance is affected by distance and Tx power. The show their impact, FRR at the distance range of $[1m,\ 7m]$, and Tx power of $[4.5dBm,\ -1.5dBm]$. Both XBee implementation on USRP and commodity BLE devices are tested for completeness.



(a) USRP



(b) Commodity device

Figure 6.21: FRR with different Tx power and distance.

Fig. 6.21(a) reveals that the FRR of XBee on USRP gradually decreases from 90% to 80% with an increase in distance and the decrease in Tx power when the distance is within $5m$. A sharp decrease occurs at the distance of $7m$, at a low Tx power of $1dBm$ or lower. In Fig. 6.21(b), XBee on the commodity device reaches FRR over 60% when the distance is shorter than $3m$. The performance on commodity devices is worse than that of the USRP. While the details of commodity hardware are hidden, we believe that the performance gap is mainly due to cheap RF components in commodity BLE with low antenna gains and inaccurate phase detection, which is subject to change for different hardware and vendors. With the aim to provide stable and reproducible results, and to offer thorough analysis and deep understanding, we use USRP in the following parts of the section (unless otherwise stated).

Tx Power (dBm)	-22	-16	-10	-3	1	4.5
Rx RSS (dBm)	-75	-71	-64	-57	-53	-50

Table 6.2: Rx RSS changes linearly with the Tx power.

To understand the relationship between the FRR and the Rx RSS, we studied the relationship between the Tx power and Rx RSS. In the experiment, we use the CC2530 ZigBee device with the CC2540 BLE device. We connect the Tx and Rx with a cable with a $30dBm$ attenuator. In Table. 6.2, we show the relationship between the Tx power and the Rx RSS. We find that throughout the $30dB$ Tx power range, the Tx power almost changes linearly with the Rx RSS. Combined with the results in Fig. 6.21, we can tell the relationship between the FRR and the Rx RSS.

• **LoS Locations** We evaluate the FRR in various locations within a university building, including the lobby (F1), a meeting room with minimal obstacles (F2), a lab (F3), and

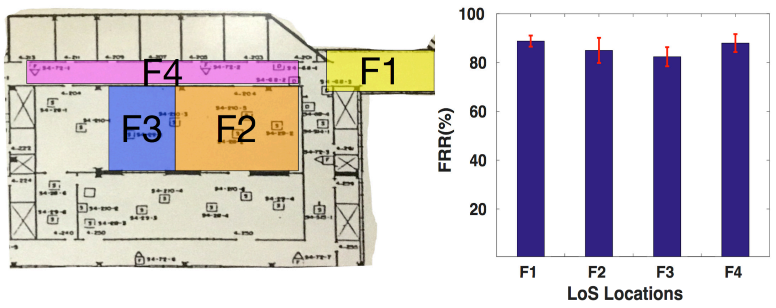


Figure 6.22: LoS scenarios.

the hallway (F4). The distance between the ZigBee transmitter and the XBee receiver was kept at $1m$ while maintaining the line of sight (LoS). As illustrated in Fig. 6.22, the average FRR in the lobby and the hallway are both over 85%, while falling as low as 80% in the lab environment. This is because the lab is crowded with many WiFi (e.g., laptops) and Bluetooth (e.g., wireless mouse) devices causing strong interference.

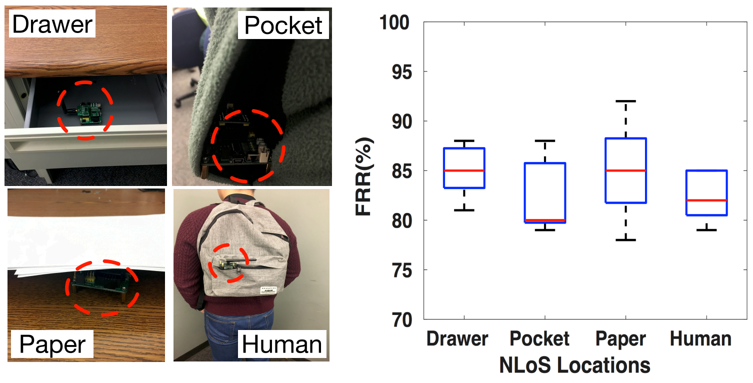


Figure 6.23: NLoS scenarios.

• **NLoS Locations** We also study the performance of XBee under various realistic NLoS scenarios, where the ZigBee transmitter is sitting in a drawer, in a pocket, on the desk covered by some paper, or obstructed by the human body. And a USRP BLE receiver is put $1m$ away. We note that this experiment was conducted in a large meeting room with mostly open space (F2 in Fig. 6.22) to factor out other channel effects (e.g., multipath). Fig. 6.23 depicts the results – While the pocket scenario shows the highest impact, FRR is kept at a reasonable level of above 80%, which demonstrates XBee's reliability under various NLoS scenarios in practice.

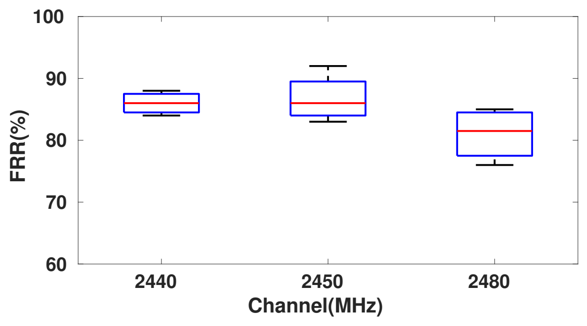


Figure 6.24: Interference on different channels.

• **Impact of Channel** XBee operates on all the overlapping channels between ZigBee and BLE, ranging from [2410MHz, 2420MHz, ..., 2480MHz]. Among them we choose three representative channels: one is overlapping with the WiFi channel, i.e., 2440MHz, another is the BLE advertising channel, i.e., 2480MHz, and the third is a relative clean channel, i.e., 2450MHz, as illustrated in Fig. 6.24. We compare the impact of different sources of wireless interference, i.e., WiFi and BLE advertising packets. From Fig. 6.24, we find the average FRR of XBee on channels 2440MHz and 2450MHz are both over 85%, while the FRR on channel 2480MHz is slightly lower, around 80%. We believe that is due to the huge amount of BLE advertising packets on the channel 2480MHz interfering with the cross-decoding.

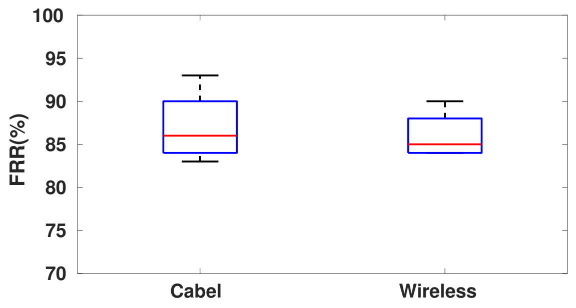


Figure 6.25: Low noise scenarios: cable and over the air

• **Cross-decoding Under Low Noise** In this experiment, we factor out the effect of the wireless channel to provide insights into the performance limits of cross-decoding. To do completely remove wireless noise, we connect the ZigBee transmitter and the USRP XBee receiver through a cable. We also measure low noise scenario over the air, by putting the transmitter and the receiver pair side by side. Fig. 6.25 demonstrates that, in both cases, FRR was kept similar at 85%, which is effectively the maximum performance of XBee under the ideal channel.

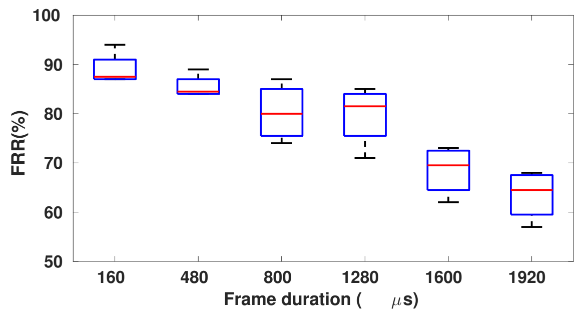


Figure 6.26: FRR under varying frame durations.

• **Impact of Packet Duration** Compared to BLE 4.0 standard which restricts payload size to be 33 bytes at maximum, the newly introduced BLE 4.2 allows a much longer payload of up to 251 bytes. This provides the opportunity to cross-decode long ZigBee frames. This naturally leads to a question of the impact of the frame length on XBee's performance. To investigate this, we first place ZigBee (CC2530) transmitter nearby (i.e., 1m) the USRP XBee receiver to minimize channel effects. The antenna gains are $30dB$ at the Tx and $10dB$ at the Rx. Under this setting, we increase the frame duration from $160\mu s$ to $1920\mu s$, corresponding to 20 and 240 BLE payload bytes, which closely approaches the BLE 4.2 limit. As shown in Fig. 6.26, the FRR of XBee decreases with the frame duration, from 90% to 66% due to a higher chance of corruption, as in any wireless communication designs. It shows XBee's FRR is kept a reasonable level of 65% under a long frame size of $1920\mu s$ defined by the latest BLE 4.2 standard – indicating XBee is able to support long and bursty data delivery.

• **Repeated Receiver ID** Recall that receiver ID corresponds to the preamble in BLE. Therefore, successful detection of the receiver ID is critical for XBee to successfully receive frames (i.e., cross-decode). To this end, we evaluate how the repetition of the receiver ID affects FRR. Fig. 6.27 demonstrates the increase in FRR with more repetition of receiver ID. A single receiver ID (i.e., repeat = 1) yields FRR=50%, however, appending two receiver IDs (i.e., repeat = 2) quickly increases the FRR to 85%. Increasing the repetition over two does not offer notable gain while increasing the overhead.

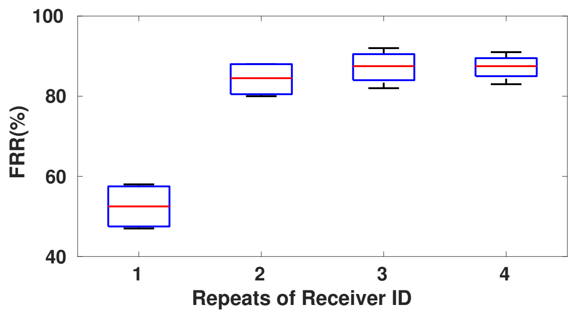


Figure 6.27: Repeats of receiver ID

Therefore XBee uses two receiver IDs by default.

6.7.5 Energy Consumption

Here we study the energy consumption of XBee on the commodity CC2650 BLE board. Recall that the additional energy consumption brought by XBee is the cross-decoding through matrix multiplication in Fig. 6.13. To measure the energy cost, we measure the average computation time the BLE board takes to cross-decode each ZigBee symbol, then multiplied by the average power of the CC2650 BLE board. Our measurement finds that it takes $174\mu s$ on average to cross-decode a ZigBee symbol. The average power consumption of the board is $4.35mW$ when active [58]. So it takes $0.7\mu J$ to cross-decode a ZigBee symbol.

6.7.6 Application: Bidirectional CTC

XBee is a receiver-side design, which can be applied alongside existing transmitter-side PHY-CTC to achieve bidirectionality – an essential function for multiple critical aspects of networking. We demonstrate the feasibility of bi-directional CTC, with a case study of ZigBee and BLE. This is achieved by utilizing [55], a state-of-the-art BLE to ZigBee CTC technique, with XBee. Specifically, we implement an acknowledge mechanism via the following two steps. First, a commodity ZigBee device sends native ZigBee frames to a BLE receiver. Upon correct cross-decoding at the BLE receiver via XBee, the BLE device sends back cross-technology ACKs through signal emulation in [55]. The packet interval at the ZigBee transmitter is set between [$50ms$, $500ms$]. The result depicted in Fig. 6.28 shows that the frame reception and ACK rates are both low when under short transmission interval. They improve as the frame interval grows larger. This is because the USRP-based receiver has a delay in receiving, processing, and sending back ACK, ranging between [50ms, 100ms]. The long delay causes ACKs to collide with the coming ZigBee to BLE packets. When the frame interval is large enough, e.g., 500ms, XBee correctly cross-decodes 85% of the frames, and acknowledges 95% of the decoded frames, thereby successfully demonstrating bidirectional CTC in practice. We believe this will be a key enabler to sophisticated cross-technology protocol designs at the upper layers.

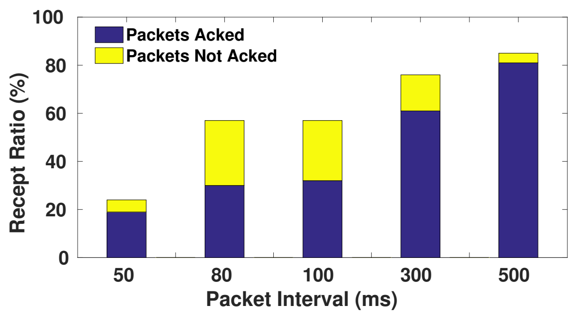


Figure 6.28: Acknowledged ZigBee packets

6.8 Conclusion

In summary, XBee is the first receiver-side PHY-layer CTC with cross-decoding. By moving the complexity to the receiver side, it covers the fundamental but missing piece to enable PHY-layer CTC from lower-end transmitted to higher-end receiver to accomplish PHY-layer CTC in all directions, paving the way for advanced cross-technology channel negotiations and collaborations in practice.

Chapter 7

Discussion

In this section, we discuss several critical concerns in CTC, including the roles CTC played in the IoT security, cutting-edge research directions, costs in deploying CTC, and its counterpart in the real world.

7.1 CTC in IoT Security

CTC enables direct communication between heterogeneous wireless devices. It inevitably brings security concerns if the techniques are used in wrong places. Here we discuss CTC threat models, the vulnerability of existing ZigBee/BLE protocols, and possible ways to detect and protect from CTC transmitters.

7.1.1 CTC thread model

Since the packet-level and PHY-layer CTC have distinct features, we discuss their threat models respectively.

Packet-level CTC. Packet-level CTC solutions are commonly achieved through building covert channels between heterogeneous devices upon packet-level information such as energy level [54, 18], packet length [6], and packet reorder [52, 41]. However, such covert channels are not under the control of existing security mechanism in IoT protocols. That is because existing security mechanisms are mainly built on the contents of packets, which are orthogonal to the packet characteristics. So it is essential to build efficient detection methods to avoid stealthy data leakage.

PHY-layer CTC. PHY-layer CTC solutions, on the other hand, are built upon the payloads of valid wireless packets. They inherit the native (de)modulation procedure so that they are capable of packet sniffering and injection. Potential attacks include the man-in-the-middle (MITM) attack, Denial-of-Service (DoS), and packet replay attack.

7.1.2 Vulnerability in IoT protocols

At the meantime, IoT protocols are inherently vulnerable, because they are usually designed to be simple to satisfy the limited power supply and computation capability on the board. The typical vulnerabilities of Bluetooth Low Energy (BLE) and ZigBee are as follows.

Bluetooth Low Energy Pairing process is essential in BLE. Most existing BLE devices work in the *Just Works* mode. In this mode, the packets exchange message in plain text, which provides no security against MITM attacks. To counter MITM, enhanced mechanisms like *Passkey Display*, *Out of Band (OOB)*, and the latest *Numeric Comparison (LE Secure Connections Pairing)* must be adopted [59].

ZigBee The ZigBee protocol also suffers from some protocol vulnerabilities. Among them are the *default link key values* used by manufactures, *unauthenticated acknowledgement packets (ACK)* generated by a malicious attacker, and *unencrypted keys* when a non-preconfigured device joins a network [60]. All these protocol drawbacks make ZigBee devices vulnerable to MITM, DoS, and replay attacks.

7.1.3 Detection of CTC Attack

According to the [61], packet level CTC can be effectively classified via a decision tree with features like energy, packet duration, and packet interval. The classification accuracy is claimed to be 94.7%. To protect from packet level CTC, a jamming based solution is adopted to successfully disrupt the covert channel.

On the other hand, attacks from PHY-CTC are no more than the known cyber attacks in the homogeneous technology. Existing security mechanism is still applicable to protect from the attacks. However, existing security protocols can not tell whether the attacker is a normal device or a heterogeneous one, making it hard to distinguish the attacker. Here we provide several methods in the PHY layer that can help us identify

CTC attackers.

Energy sensing In PHY-CTC, a CTC packet is embedded within another wireless packet. This is abnormal because the channel is usually idle before and after a non-collided packet in real-world wireless communication. As a result, the high energy levels before and after a legitimate packet are strong clues for a CTC transmitter.

Spectrum analysis Heterogeneous wireless devices occupy different width of spectrum. In the frequency domain, it is illustrated as different envelopes of the signal. If the envelope at the receiver side is much wider/narrower than a homogeneous transmitter. It is possible to be a CTC transmitter.

7.2 Cutting-edge Research Directions

There are a lot of CTC solutions proposed since it was first proposed in 2009[6]. Despite the engineering efforts to make CTC possible, novel ideas are also proposed and deeply affect the followers. The cutting-edge research directions in CTC includes but not limited to:

Covert channels transparent to the upper layers Packet-level CTC solutions are essentially building covert wireless channels upon exiting wireless traffics. Such a covert channel can be built upon energy level, packet length, and packet reorder. The main concern in designing such covert channels is how to build reliable communication channels while being transparent to the upper layer data traffic. This line of research also inspires studies in the security of CTC itself [61] to avoid the leakage of sensitive information in such covert channels.

Signal emulation under constraints. Since the idea of signal emulation first proposed in WEBee [24], a lot of follow-up works study better emulation techniques under constrained wireless modulation schemes. Research topics include: 1) How to choose the best modulation schemes for emulation? Emulation in the ZigBee phase shifts is proved to achieve better performance than the direct signal emulation under the same QAM resources [62, 55]; 2) How to instruct CTC emulation with the hints from the receiver side? In other words, the emulated signal is not necessarily the same as that at the receiver side, only if it can be demodulated. Based on this idea, BlueBee[55] and XBee [63] successfully build CTC for low-end IoT devices; 3) How to design proper coding schemes to overcome the emulation errors? TwinBee [64] takes advantage of the cyclic shift feature of ZigBee DSSS modulation to successfully overcome the emulation errors due to WiFi cyclic prefix; 4) What are the theoretical boundary of signal emulation in CTC? Theoretical analysis of CTC performance boundary given transmitter's coding schemes is still an open question to explore.

Tradeoff between heterogeneous wireless technologies Another branch of PHY-CTC focuses on constructing some mixed signal that conveys messages to heterogeneous wireless receivers. In other words, a transmitter emits some special signal mixing a legitimate WiFi and a ZigBee packet. At the receiver sides, both WiFi and ZigBee can demodulate intended signal [53]. There is a tradeoff whether the signal is more like WiFi or ZigBee. Some machine learning techniques are adopted to do such adjustment [65]. Of course, according to the information theory, the mixed signal sacrifices reliability, i.e., SNR at both receivers.

7.3 Costs in CTC

Of course the benefits of CTC come along with certain costs. These costs can be categorized as the configuration cost, bandwidth cost, management cost, as well as the hardware cost. Comparison results are shown in Table 7.1.

	Config. Cost	Band. Cost	Manage. Cost	Hardware Cost	Data Rate
Gateway	Low	Medium	Medium	Medium	High
Packet-CTC	High	None	High	None	Low
PHY-CTC	Medium	Low	Medium	None	High

Table 7.1: Comparison of the costs of the gateway and existing CTC solutions

Configuration Cost: Configuration cost includes all the necessary hardware/software configuration and computation. The gateway solution has low cost in the configuration. The packet-level CTC, however, requires complex configuration, since it builds covert channels with customized (de)modulation schemes. Constructing PHY-CTC is no more than transmitting/receiving a normal wireless packet, but it requires some computational cost in generating CTC bytes. It includes complex matrix multiplication and inversion. However, there are some techniques to reduce such costs like i) combining multiple matrix operations; ii) precomputing the bytes for all possible symbols and stored them on the board; and iii) connecting to a remote server to do the CTC conversion in real-time [66].

Bandwidth Cost: The bandwidth cost includes the impact on the other wireless communication in the air. The gateway solution will increase the network overhead due to the packets flowing into and outside the gateway. The packet-CTC solutions are free in bandwidth cost if they piggyback messages along with packet-level information in existing wireless packets [9, 52, 41, 67]. The PHY-CTC is no more than transmitting/receiving a normal wireless packet, so its impact on the spectrum without the duplication of wireless packets in the air.

Management Cost: Management costs include the MAC layer coordination as well as the network layer scheduling. The gateway solution requires the conversion of wireless packets in the PHY/link/network layers so that it still needs some medium operation cost inside the gateway. The packet-CTC requires high costs for the network supply of its customized modulation. The PHY-CTC requires medium operation cost to overcome the heterogeneity link and network compatibility. However, till now, studies of CTC network protocols are still in the early ages and might be a promising topic in the future.

Hardware Cost: Hardware costs include the purchase of new hardware. It costs about $2,000 to purchase new commercial hardware [68]. In contrast, there is no cost in CTC solutions because they are built upon existing wireless hardware.

7.4 CTC Counterparts

In the industry, a counterpart of CTC is the software-defined radio (SDR). In contract to the several thousand dollars commercial SDR [68], deploying CTC techniques on an existing wireless device is free. As a result, CTC satisfies the requirement for the flexible and large-scale deployment of IoT devices in the real-world IoT application scenarios. In addition, more and more combo chips appear in the market, such as the ZigBee-Bluetooth combo chip [69] and some WiFi-Bluetooth combo chips [70]. In the combo chips, multiple wireless devices are not simply adhered on the same board, instead, they share the same radio, amplifier, and other signal processing parts in common.

Wireless combo chips can be viewed as some ”intermediate” form between SDR and a specialized wireless chip. The studies of CTC will help the industry understand what are the bottlenecks in combining multiple wireless technologies on a single chip, and eventually push the birth of some universal CTC platforms.

Chapter 8

Future Work

8.1 Short-term Plan

Enhanced cross-technology communication. Existing CTC technologies are commonly based on mature and popular wireless standards, such as the WiFi 802.11a/b/g standards. However, recent advances in WiFi technology have come out with latest standards such as 802.11n, 802.11ac, and 802.11ax, which support multiple-input and multiple-output (MIMO), wider bandwidth up to $160MHz$, and orthogonal frequency-division multiple access (OFDMA). These technologies bring opportunities to dramatically increase CTC data rate, along with the challenges of complex physical layer designs and fine-grained spectrum use. An interesting question is how to develop CTC technologies to take advantage of the latest standards.

Ubiquitous connectivity. Existing CTC technologies are largely designed for wireless technologies on the 2.4GHz ISM band, such as WiFi (2.4GHz), ZigBee, and Bluetooth. Few studies are done with wireless technologies on other ISM bands, such as the LTE on 5GHz band and the LoRa, SigFox, and 802.11ah on the sub-1GHz band. In addition, the wireless communication and backscatter communication can also be bridged by using the wireless device, such as the WiFi device as the RFID reader. I believe bringing wireless technologies in all bands into a unified system will be a milestone towards ubiquitous connection.

Enhanced spectrum efficiency and fairness. The heterogeneous MAC layer is also a big challenge besides the physical layer. There are several problems to solve in order to let multiple technologies share the spectrum: (i) some wireless technologies, such as WiFi and ZigBee, adopt carrier sense to share the spectrum while others, such as Bluetooth, uses frequency hopping. It is a problem how to design a new MAC protocol compatible with different schemes. (2) When different wireless technologies are sharing the spectrum, the goal of MAC protocol design is not to optimize the throughput of a certain technology, but instead the throughput of the heterogeneous network. For example, a low-rate device might need to ask if any high-rate devices want to use the spectrum before it occupies the channel. This will finally lead to harmonious network coexistence.

Seamless network handover. In the mobile scenario, a user may be frequently changing among several networks. Network handover is slow, especially among heterogeneous wireless networks. CTC breaks the boundary between wireless technologies, which enables the share of information, such as the signal strength and authentication information, among heterogeneous wireless technologies, so that the devices can intelligently decide when to do a handover to achieve seamless network handover.

8.2 Long-term Plan

Collaborative network model. The introduction of CTC technology breaks the assumption that wireless technologies are identical, thus we need to remodel the wireless network embracing all wireless devices. A specific feature of the new model is that each device can intelligently choose which wireless standard it should use based on the current network environment.

Abstractive user interface. To better serve the upper layer user, user interface abstraction is an emergent and necessary task I plan to work on to hide the complex network details. Due to the new network topology, it should be redesigned how to hide the sophisticated network architecture to guarantee the quality of service for the user. For example, data traffic may arrive a smartphone user through both the WiFi and Bluetooth link. The user interface should hide the details of heterogeneous wireless technologies and provide a unified interface.

Collaborative data fusion. CTC enables collaborative data fusion. The technique of data fusion blends diverse information acquired from various sources to gain a precise view of the physical world. This is critical especially in detecting emergent events in IoT. CTC brings two distinct changes compared with traditional data fusion: (1) Universal data fusion is available on each wireless device, no matter its technology, which is more responsive and energy efficient; (2) Localized IoT devices can have access to the Internet with existing infrastructure, such as WiFi and LTE base stations, so that they can easily upload and store data at the cloud. I believe the collaborative data fusion will greatly change the Big Data collection and intelligent analysis.

Collaborative privacy preserving. The fusion of wireless networks will also bring privacy and security problems, which is also what I plan to work on. CTC might be used as a source of attack by adversaries. Sophisticated privacy and security algorithms have been introduced within a certain technology, such as the wired equivalent privacy (WEP) and Wi-Fi protected access (WPA) algorithms in the WiFi network. But it is not the case for low-power standards such as the Bluetooth Low Energy, which deliberately avoids complex encryption methods for energy concern. As a result, adversaries might use CTC technology as a back door to attack existing systems.

References

[1] Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions). `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`.

[2] Fadel Adib, Swarun Kumar, Omid Aryan, Shyamnath Gollakota, and Dina Katabi. Interference alignment by motion. In *Proceedings of ACM MobiCom, 2013*, 2013.

[3] Tian Hao, Ruogu Zhou, Guoliang Xing, Matt W Mutka, and Jiming Chen. Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks. *IEEE Transactions on mobile computing*, 13(6):1379–1392, 2014.

[4] Tao Jin, Guevara Noubir, and Bo Sheng. Wizi-cloud: Application-transparent dual zigbee-wifi radios for low power internet access. In *INFOCOM*, pages 1593–1601, 2011.

[5] Saravana Rathinakumar, Bozidar Radunovic, and Mahesh K Marina. Cprecycle: Recycling cyclic prefix for versatile interference mitigation in ofdm based wireless systems. In *Proceedings of ACM CoNEXT 2016*, 2016.

[6] Kameswari Chebrolu and Ashutosh Dhekne. Esense: communication through energy sensing. In *Proceedings of ACM MOBICOM*, pages 85–96. ACM, 2009.

[7] Yifan Zhang and Qun Li. Howies: A holistic approach to zigbee assisted wifi energy savings in mobile devices. In *Proceedings of IEEE INFOCOM*, pages 1366–1374. IEEE, 2013.

[8] Xinyu Zhang and K.G. Shin. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *Proceedings of IEEE INFOCOM*, pages 3094–3101, April 2013.

[9] Song Min Kim and Tian He. Freebee: Crosstechnology communication via free sidechannel. In *Proceedings of ACM MOBICOM*, 2015.

[10] Abusayeed Saifullah, Mahbubur Rahman, Dali Ismail, Chenyang Lu, Ranveer Chandra, and Jie Liu. Snow: Sensor network over white spaces. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 2016.

[11] Souvik Sen, Romit Roy Choudhury, and Srihari Nelakuditi. No time to countdown: Migrating backoff to the frequency domain. In *MobiCom '11*, pages 241–252. ACM, 2011.

[12] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. Successive interference cancellation: Carving out mac layer opportunities. *IEEE Transactions on Mobile Computing*, 12(2):346–357, 2013.

[13] Karthikeyan Sundaresan, Srikanth V Krishnamurthy, Xinyu Zhang, Amir Khojastepour, Sampath Rangarajan, et al. Trinity: A practical transmitter cooperation framework to handle heterogeneous user profiles in wireless networks. In *MobiHoc '15*, pages 297–306. ACM, 2015.

[14] Sangki Yun and Lili Qiu. Supporting wifi and lte co-existence. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 810–818. IEEE, 2015.

[15] Xinyu Zhang and Kang G Shin. Enabling coexistence of heterogeneous wireless systems: case for zigbee and wifi. In *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, page 6. ACM, 2011.

[16] Xinyu Zhang and Kang G Shin. Cooperative carrier signaling: Harmonizing coexisting wpan and wlan devices. *Networking, IEEE/ACM Transactions on*, 21(2):426–439, 2013.

[17] Ruogu Zhou, Yongping Xiong, Guoliang Xing, Limin Sun, and Jian Ma. Zifi: wireless lan disc overy via zigbee interference signatures. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 49–60. ACM, 2010.

[18] Zicheng Chi, Yan Li, Hongyu Sun, Yao Yao, Zheng Lu, and Ting Zhu. B2w2: N-way concurrent communication for iot devices. In *Proceedings of ACM Sensys 2016*, 2016.

[19] Eugene Chai, Karthik Sundaresan, Mohammad A Khojastepour, and Sampath Rangarajan. Lte in unlicensed spectrum: are we there yet? In *MobiCom '16*, pages 135–148. ACM, 2016.

[20] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 356–369. ACM, 2016.

[21] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. Passive wi-fi: Bringing low power to wi-fi transmissions. In *NSDI '16*, pages 151–164. USENIX Association, 2016.

[22] Zhenjiang Li, Yaxiong Xie, Mo Li, and Kyle Jamieson. Recitation: Rehearsing wireless packet reception in software. In *MobiCom '15*, pages 291–303. ACM, 2015.

[23] Jiajue Ou, Yuanqing Zheng, and Mo Li. Misc: Merging incorrect symbols using constellation diversity for 802.11 retransmission. In *INFOCOM 2014*, pages 2472–2480. IEEE, 2014.

[24] Zhijun Li and Tian He. Webee: Physical-layer cross-technology communication via emulation. In *Proceedings of ACM MobiCom 2017*, 2017.

[25] Wireless open-access research platform. `http://warpproject.org/trac/`.

[26] Micaz datasheet. `http://www.memsic.com`.

[27] Iperf. `http://iperf.fr/`.

[28] Jose Gutierrez, Marco Naeve, Ed Callaway, Monique Bourgeois, Vinay Mitter, Bob Heile, et al. Ieee 802.15. 4: a developing standard for low-power low-cost wireless personal area networks. *network, IEEE*, 15(5):12–19, 2001.

[29] ERVE LOVELACE, JM Sutton, and E Salpeter. Digital search methods for pulsars. *Nature*, 222:231–233, 1969.

[30] A Glass, B Ali, and E Bastaki. Design and modeling of h-ternary line encoder for digital data transmission. In *Proceedings of IEEE ICII 2001*, volume 2, pages 503–507. IEEE, 2001.

[31] vsftpd - secure, fast ftp server for unix-like systems. `https://security.appspot.com/vsftpd.html`.

[32] Plex media server - your media on all your devices. `https://www.plex.tv`.

[33] Jun Huang, Guoliang Xing, Gang Zhou, and Ruogu Zhou. Beyond co-existence: Exploiting wifi white space for zigbee performance assurance. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 305–314, Oct 2010.

[34] Chieh-Jan Mike Liang, Nissanka Bodhi Priyantha, Jie Liu, and Andreas Terzis. Surviving wi-fi interference in low power zigbee networks. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, 2010.

[35] Xinyu Zhang and Kang G. Shin. Cooperative carrier signaling: Harmonizing coexisting wpan and wlan devices. *IEEE/ACM Trans. Netw.*, 21(2), April 2013.

[36] Minkeun Ha, Seong Hoon Kim, Hyungseok Kim, Kiwoong Kwon, Nam Giang, and Daeyoung Kim. 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*, 2012.

[37] Minkeun Ha, Kiwoong Kwon, Daeyoung Kim, and Peng-Yong Kong. 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*, 2014.

[38] Stefan Nastic, Hong Linh Truong, and Schahram Dustdar. Sdg-pro: a programming framework for software-defined iot cloud gateways. *J. Internet Services and Applications*, 6(1):21:1–21:17, 2015.

[39] Soheil Qanbari, Negar Behinaein, Rabee Rahimzadeh, and Schahram Dustdar. 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*, pages 38–43, 2015.

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

[41] Zhimeng Yin, Wenchao Jiang, Song Min Kim, and Tian He. C-morse: Cross-technology communication with transparent morse coding. In *Proceedings of IEEE INFOCOM 2017*, 2017.

[42] Dominic Spill and Andrea Bittau. Bluesniff: Eve meets alice and bluetooth. *WOOT*, 7:1–10, 2007.

[43] Bluetooth technology website.

[44] Ieee802.org. Ieee 802.15.4. 2012.

[45] Scapy radio. `https://github.com/BastilleResearch/scapy-radio`.

[46] Ti cc2540 development kit. `http://www.ti.com/tool/cc2540dk/`.

[47] T. H. Laine, C. Lee, and H. Suk. Mobile gateway for ubiquitous health care system using zigbee and bluetooth. In *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 139–145, July 2014.

[48] G. Ballou. *Handbook for Sound Engineers*. Focal Press. Focal, 2005.

[49] FCC. *Fact Sheet: Spectrum Frontiers Order To Identify, Open Up Vast Amounts Of New High-Band Spectrum For Next Generation (5g) Wireless Broadband*. Available at https://apps.fcc.gov/edocs_public/attachmatch/DOC-340310A1.pdf.

[50] FCC. Increased availability of spectrum for unlicensed uses in the 5 ghz band, 2014. Available at https://www.fcc.gov/document/fcc-increases-5ghz-spectrum-wi-fi-other-unlicensed-uses.

[51] FCC. Rules for unlicensed operations in the tv and the 600 mhz band, 2015. Available at https://www.fcc.gov/document/fcc-adopts-rules-unlicensed-services-tv-and-600-mhz-bands.

[52] Wenchao Jiang, Zhimeng Yin, Song Min Kim, and Tian He. Transparent cross-technology communication over data traffic. In *Proceedings of IEEE INFOCOM, 2017*, 2017.

[53] Zicheng Chi, Zhichuan Huang, Yao Yao, Tiantian Xie, Hongyu Sun, and Ting Zhu. Emf: Embedding multiple flows of information in existing traffic for concurrent communication among heterogeneous iot devices. In *Proceedings of IEEE INFOCOM 2017*.

[54] Xiuzhen Guo, Xiaolong Zheng, and Yuan He. Wizig: Cross-technology energy communication over a noisy channel. In *Proceedings of IEEE INFOCOM 2017*.

[55] Wenchao Jiang, Zhimeng Yin, Ruofeng Liu, Song Min Kim, Zhijun Li, and Tian He. Bluebee: a 10,000x faster cross-technology communication via phy emulation. In *Proceedings of ACM Sensys 2017*, 2017.

[56] cc2420 data sheet. `http://www.ti.com/lit/ds/symlink/cc2420.pdf`.

[57] Robin Heydon. *Bluetooth low energy: the developer's handbook*, volume 1.

[58] Christin Lee Joakim Lindh and Marie Hernes. *Measuring Bluetooth Low Energy Power Consumption.* Texus Instruments. 2017.

[59] Angela Lonzetta, Peter Cope, Joseph Campbell, Bassam Mohd, and Thaier Hayajneh. Security vulnerabilities in bluetooth technology as used in iot. *Journal of Sensor and Actuator Networks*, 7(3):28, 2018.

[60] `https://research.kudelskisecurity.com/2017/11/21/zigbee-security-basics-part-3/`.

[61] G. Chen and W. Dong. Jamcloak: Reactive jamming attack over cross-technology communication links. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 34–43, Sep. 2018.

[62] Xiuzhen Guo, Yuan He, Jia Zhang, and Haotian Jiang. Wide: Physical-level ctc via digital emulation. In *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, IPSN '19, pages 49–60, New York, NY, USA, 2019. ACM.

[63] Wenchao Jiang, Song Min Kim, Zhijun Li, and Tian He. Achieving receiver-side cross-technology communication with cross-decoding. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, pages 639–652, New York, NY, USA, 2018. ACM.

[64] Yongrui Chen, Zhijun Li, and Tian He. Twinbee: Reliable physical-layer cross-technology communication with symbol-level coding. In *Proceedings of IEEE INFOCOM 2018*, 2018.

[65] Anatolij Zubow, Piotr Gawłowicz, and Suzan Bayhan. Deep learning for cross-technology communication design. *arXiv preprint arXiv:1904.05401*, 2019.

[66]

[67] Zicheng Chi, Yan Li, Hongyu Sun, Yao Yao, Zheng Lu, and Ting Zhu. B2w2: N-way concurrent communication for iot devices. In *Proceedings of ACM Sensys*, 2016.

[68] `https://www.ettus.com/all-products/un210-kit/`.

[69] `https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840`.

[70] `https://www.cypress.com/products/wi-fi-bluetooth-combos`.