LEGO-Fi: Transmitter-Transparent CTC with Cross-Demapping

Xiuzhen Guo[1], Yuan He[1], Xiaolong Zheng[2], Zihao Yu[1], Yunhao Liu[1,3]
[1]School of Software & BNRist, Tsinghua University, China
[2]School of Computer Science, Beijing University of Posts and Telecommunications, China
[3]Department of Computer Science and Engineering, Michigan State University, USA
{guoxz16, zh-yu17}@mails.tsinghua.edu.cn, heyuan@mail.tsinghua.edu.cn,
zhengxiaolong@bupt.edu.cn, yunhao@cse.msu.edu

***Abstract*—Cross-Technology Communication (CTC) is an emerging technique that enables direct communication across different wireless technologies. The state-of-the-art works in this area propose physical-level CTC, in which the transmitters emulate signals that follow the receiver's standard. Physical-level CTC means considerable processing complexity at the transmitter, which doesn't apply to the communication from a low-end transmitter to a high-end receiver, e.g. from ZigBee to WiFi. This paper presents transmitter-transparent cross-technology communication, which leaves the processing complexity solely at the receiver side and therefore makes a critical advance toward bidirectional high-throughput CTC. We implement our proposal as LEGO-Fi, the communication from ZigBee to WiFi. The key technique inside is cross-demapping, which stems from two key technical insights: (1) A ZigBee packet leaves distinguishable features when passing the WiFi modules. (2) Compared to ZigBee's simple encoding and modulation schemes, the rich processing capacity of WiFi offers extra flexibility to process a ZigBee packet. The evaluation results show that LEGO-Fi achieves a throughput of 213.6Kbps, which is respectively $13000\times$ and $1200\times$ faster than FreeBee and ZigFi, the two existing ZigBee-to-WiFi CTC approaches.**

I. Introduction

The proliferation of Internet of Things (IoT) applications brings about the increasingly dense deployments of various wireless devices. Coexistence of heterogeneous wireless technologies becomes a common situation, where the interplay across technologies draws a lot of research attention [17], [21], [22], [27]–[29]. Under such circumstances, cross-technology communication (CTC) emerges to enable direct communication among devices that follow different communication standards [1], [5], [10], [26].

With CTC, the heterogeneous wireless devices can build a new communication channel to coordinate with each other, so that wireless interference and collisions will be appropriately handled [18], [24], [35]. The ZigBee sensors installed in a smart home can directly report the sensory data to a WiFi device, which may further share the data via high-speed WiFi network. The front-end sensors and controllers in a smart factory can closely collaborate with each other by directly transmitting commands and data, so as to the meet the requirement of real-time applications [9], [19]. CTC not only provides a new way to manage wireless networks, but also enhances the ability of interoperation and in-situ data exchange among heterogeneous devices.

To realize CTC, one will meet two major challenges: the incompatibility between technologies and the asymmetry of device capacity. Early works in this area propose packet-level CTC, which manipulate transmitted packets and use the packet length [31], the received signal strength [3], [7], [25], [34], the transmission timings [11], [13], and other features [4], [6] as the information carrier. Recent works propose physical-level CTC. WEBee [14] proposes the technique of physical-level emulation. It uses the high-speed WiFi radio to emulate the desired signals of the low-speed ZigBee radio. BlueBee [15] emulates legitimate ZigBee frames with a Bluetooth radio by using the phase differences between samples. Physical-level CTC significantly enhances the throughput, compared with packet-level CTC.

In spite of the promising progress, we still face a critical gap towards the bidirectional high-throughput CTC. The reason is that the state-of-the-art physical-level emulation technique incurs considerable processing cost at the transmitter, preventing it from being applied to the reverse direction, i.e. from a low-end device to a high-end device. Recently, [23] proposes a symbol-level CTC via payload encoding and recycles the idle listening result of WiFi to decode ZigBee signals. Other proposals ([6], [11], [13]) for CTC from ZigBee to WiFi all stay at the packet level.

Taking CTC from ZigBee to WiFi as an example, we list the critical challenges as follows. First, the bandwidth of ZigBee (2MHz) is much narrower than that of WiFi (20MHz). Given the fixed sampling period at a WiFi receiver, only a small portion of the ZigBee signals can be received, which means inevitable information loss. Second, ZigBee and WiFi have totally different packet formats. Passing a ZigBee preamble into the packet detection module of WiFi will always result in failure. Third, ZigBee doesn't comply with the symbol synchronization mechanism of WiFi. Even if the ZigBee packet entered the symbol synchronization process of WiFi, the synchronization will be erroneous. Last but not least, ZigBee and WiFi adopt different modulation schemes. For the quadrature demodulator of WiFi, a signal from ZigBee (modulated by DSSS and OQPSK) contains undesirable time

delay and phase offset between its I-phase and Q-phase. Directly decoding a ZigBee signal on WiFi appears to be infeasible.

In order to address the above challenges, we in this paper propose transmitter-transparent cross-technology communication with cross-demapping, and implement it as LEGO-Fi. LEGO-Fi conceptually achieves a throughput of 250Kbps, the ceiling speed of standard ZigBee communication. The design of LEGO-Fi stems from two key technical insights: (1) A ZigBee packet leaves distinguishable features when passing the WiFi modules. (2) Compared to ZigBee's simple encoding and modulation schemes, the rich processing capacity of WiFi offers extra flexibility to process a ZigBee packet.

LEGO-Fi reuses standard WiFi modules to achieve ZigBee reception. Specifically, we devise a light-weight downsampling technique to bridge the bandwidth gap between ZigBee and WiFi. The periodicity of ZigBee preambles is preserved after passing WiFi's short preamble detection, which in turn acts as the important basis for WiFi to identify ZigBee packets. LEGO-Fi reuses the WiFi long preamble detection module to locate the start of frame delimiter (SFD) in a ZigBee packet and realizes symbol synchronization. The WiFi quadrature demodulator is reused to obtain the phase shift sequences of ZigBee signals. The key technique in LEGO-Fi is cross-demapping, which translates the CTC decoding task into a pattern identification problem. Our contributions are summarized as follows.

- We propose LEGO-Fi, a transmitter-transparent CTC for ZigBee reception at WiFi. LEGO-Fi leaves the processing complexity at the receiver and conceptually achieve a throughput up to 250Kbps, the ceiling speed of standard ZigBee communication. To the best of our knowledge, LEGO-Fi is the first work on physical-level CTC that unleashes the full communication capacity from a low-end device to a high-end device (e.g. ZigBee to WiFi).
- In LEGO-Fi, we propose a key concept called cross-demapping, which leverages the mapping between the distinguishable patterns of ZigBee symbols to decode ZigBee signals. Highlights of our design mainly include: (1) the light-weight downsampling technique to bridge the bandwidth gap between ZigBee and WiFi, (2) exploiting the periodicity of ZigBee preambles to detect ZigBee packet at WiFi, (3) the symbol synchronization scheme built on top of the SFD structure, and (4) the matching filter based cross-demapping scheme.
- We implement LEGO-Fi on the USRP platform, and evaluate its performance under a wide range of settings. The results demonstrate that LEGO-Fi achieves a throughput of 213.6Kbps in practice, which is respectively $13000\times$ and $1200\times$ faster than FreeBee and ZigFi, the two existing ZigBee-to-WiFi CTC approaches.

The rest of this paper is organized as follows. Section II discusses related works. Section III introduces the background and motivation of this work. We elaborate on our design in Section IV. Section V presents the evaluation results. We conclude this work in Section VI.

II. Related Works

As we mentioned in the first section, the incompatibility between technologies and the asymmetry of device capacity are the two major challenges of CTC. According to the way to cope with the challenges, we can classify the existing works into two categories: packet-level CTC and physical-level CTC.

Packet-level CTC. Packet related features are relatively easy to access, even when the packets come from a different communication technology. By manipulating the packets as information carrier, packet-level CTC builds a mutually accessible side channel. Depending on the communication contexts, the existing works utilize various packet related features, such as the received signal strength [3], [7], [25], [34], the packet length [31], the transmission timings [11], [13], and the channel state information [4], [6]. FreeBee [13] shifts the transmission timings of beacons to embed symbols. DCTC [11] proposes a similar technique with FreeBee, while the target to be shifted changes to the application-layer data packets. Esense [3] encodes data bits by packet presence and absence. HoWiEs [31] controls the WiFi packet length and encode bits by the length of packet on-air time. Gap Sense [30] leverages the gap between energy pulses caused by special WiFi preambles to delivery data. WiZig [7] adjusts the transmission power and grouping size of packets to encode multiple bits simultaneously. StripComm [34] proposes an interference-resilient CTC scheme that exploits Manchester Coding and interference cancellation technique. B2W2 [4] exploits the feature of overlapped channels to realize communication from BLE to WiFi. ZigFi [6] deliberately piggyback the ZigBee transmission over a WiFi transmission. It leverages the channel state information in the overlapped WiFi packet to build a side channel from ZigBee.

Generally speaking, manipulation of packet transmissions can convey data while masking the aforementioned incompatibility and asymmetry. The throughput of packet-level CTC, however, is bounded by the granularity of packet manipulation, which is at the magnitude of millisecond.

Physical-level CTC. Differing from the packet-level schemes, physical-level CTC aims at creating compliance across technologies and building the CTC channel right at the physical layer. WEBee [14] proposes physical-level emulation, which uses the high-speed WiFi radio to emulate the desired signals of the low-speed ZigBee radio. Specifically, WEBee chooses the payload of a WiFi frame so that a portion of this WiFi frame is recognized by commodity ZigBee devices transparently as a legitimate ZigBee frame. Such emulation usually contains errors. Hence, techniques like retransmission and link-layer FEC are adopted to enhance the communication reliability. BlueBee [15] presents the fact that both Bluetooth and ZigBee use the phase differences between samples, referred to as phase shifts, to indicate symbols, which makes emulation possible. Then it proposes emulation of ZigBee frames with a Bluetooth radio.

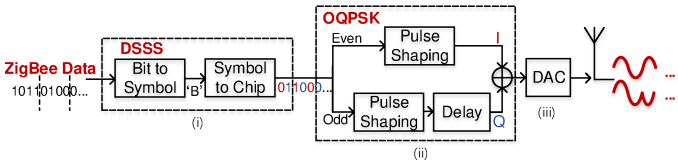


Fig. 1: The workflow of a ZigBee transmitter

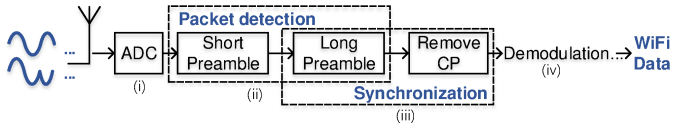


Fig. 2: The workflow of a WiFi receiver

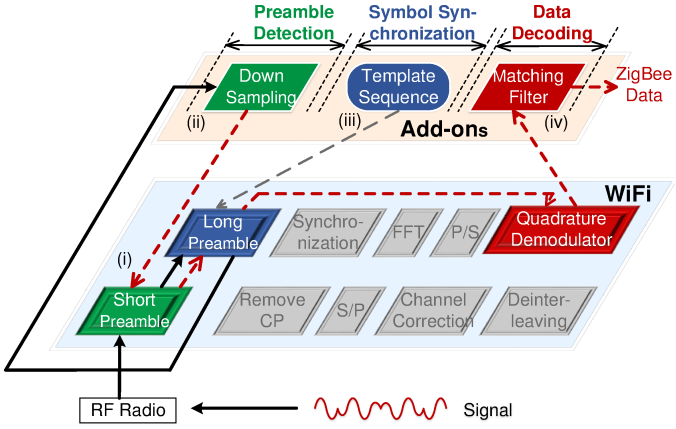


Fig. 3: The framework of LEGO-Fi

Cross-technology interference(CTI) detection. In the past decade, Cross-Technology Interference(CTI) detection attracts a lot of research interests [2], [8], [12], [16], [20], [32]. The CTI detection methods check the existence of specific interference. For example, ZiFi [35] detects the existence of nearby WiFi hotspots by ZigBee radio instead of WiFi radio to save energy of WiFi devices. ZiSense [33] identifies the ongoing wireless signal by RSSI signatures to avoid the false wake-up of ZigBee devices. Different from CTI detection, CTC not only needs to detect the existence of cross-technology signal to trigger further processing, but also needs to decode the signal and obtain the bit-level message to achieve communication.

III. Background

This section starts with a brief introduction to the workflows of a ZigBee transmitter and a WiFi receiver. This introduction helps us to better understand the underlying challenges of physical-level CTC between ZigBee and WiFi.

A. ZigBee Transmitter

Figure 1 shows how a ZigBee transmitter (IEEE 802.15.4) works. (i) Every four data bits are mapped into a symbol and each symbol is then mapped into a 32-chip sequence. This process is the so-called Direct Sequence Spread Spectrum (DSSS). The bit rate and chip rate are 250kbit/s and 2Mchip/s respectively, following the IEEE 802.15.4 standard. (ii) The chip sequences are modulated by OQPSK with half sin pulse shaping. The time offset between I-phase and Q-phase is $0.5\mu s$, which is the reciprocal of the chip rate. (iii) The modulated ZigBee signals are converted by Digital Analog Conversion (DAC) and sent via the RF radio.

B. WiFi Receiver

Taking IEEE 802.11 g/n as an example, the workflow of a WiFi receiver is shown in Figure 2. (i) The WiFi Rx radio digitalizes the received signals by Analogy Digital Conversion (ADC). (ii) The discrete samples of signals are sent to the packet detection module, which sequentially executes short preamble detection and long preamble detection. The detection result tells whether the received frame is a legitimate WiFi frame. (iii) If a legitimate WiFi frame is identified, the receiver conducts symbol synchronization and decodes the WiFi data bits. Otherwise, the WiFi receiver discards the signals.

C. Challenges

The above introduction indicates the incompatibility across technologies and the asymmetry of device capacity, which are embodied in three major challenges of CTC between ZigBee and WiFi. We will respectively discuss these challenges in the following subsections.

IV. Design

LEGO-Fi is a transmitter-transparent CTC for ZigBee reception at WiFi. The design of LEGO-Fi stems from two key technical insights: (1) A ZigBee packet leaves distinguishable features when passing the WiFi modules. (2) Compared to ZigBee's simple encoding and modulation schemes, the rich processing capacity of WiFi offers extra flexibility to process a ZigBee packet. In this section, we will first present an overview of LEGO-Fi and then introduce the design details.

A. Overview

LEGO-Fi leaves the processing complexity at the receiver and reuses the standard WiFi modules for the ZigBee reception. In LEGO-Fi, we propose a key concept called cross-demapping, which leverages the mapping between the phase shift sequences and ZigBee symbols to decode ZigBee signals. The framework of LEGO-Fi is shown in Figure 3. The received signals will be down-converted and digitalized into discrete samples by the WiFi RF radio. The samples are then sent to the processing modules to decode specific data bits as the following workflow.

(i) The received samples are first sent to the WiFi preamble detection module to decide whether the received samples are WiFi signals. If the WiFi preamble is detected, LEGO-Fi will not process them and let the WiFi modules continue decoding the packets as the traditional WiFi receiver does.

(ii) Otherwise, LEGO-Fi takes over the samples and checks whether they are from ZigBee. The received samples are first processed by downsampling to obtain complete ZigBee preambles. Then LEGO-Fi reuses the WiFi short preamble

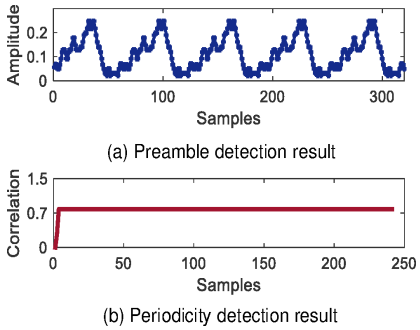


Fig. 4: The detection result of ZigBee preamble and the periodicity detection result

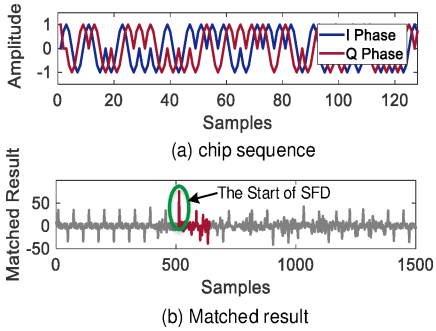


Fig. 5: The chip sequence of SFD and the correlation result with received signals

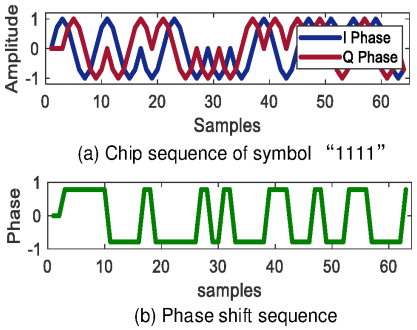


Fig. 6: Carrier frequency shift sequence (take symbol "1111" as an example)

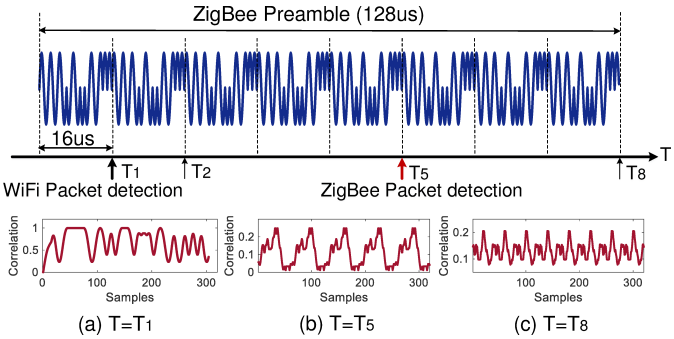


Fig. 7: Diagram of ZigBee preamble detection

detection module to detect the periodic ZigBee preamble. If not detected, LEGO-Fi discards the samples.

(iii) If the periodic ZigBee preamble is detected, LEGO-Fi conducts the symbol synchronization to segment each ZigBee symbol. LEGO-Fi feeds the Start of Frame Delimiter (SFD) template of ZigBee packets into the WiFi long preamble detection module to locate the SFD and accomplish the symbol synchronization.

(iv) Then the samples are forwarded to the WiFi quadrature demodulator to obtain the phase shift of OQPSK signals. LEGO-Fi uses the matching filter to identify the phase shift sequences. The ZigBee signal can be decoded by the mapping between phase shift sequences and ZigBee symbols, ZigBee symbols and data bits.

B. ZigBee Preamble Detection

ZigBee preamble detection is a prerequisite of receiving ZigBee packets at WiFi. In Section III, we have shown that even though the ZigBee packets cannot be directly detected, the periodicity of the ZigBee preamble is still reserved after passing the WiFi preamble detection module. But due to the asymmetry bandwidths, given the fixed sampling period, the WiFi receiver can only process a portion of the ZigBee signals, leading to the inevitable information loss.

LEGO-Fi devises a light-weight downsampling technique to bridge the bandwidth gap between ZigBee and WiFi. Then LEGO-Fi can obtain more samples in the fixed sampling window, without distorting the ZigBee signals. Then LEGO-Fi reuses the WiFi short preamble detection module to check the existence of periodicity in the information-recovered samples. If periodicity is found, a ZigBee preamble is detected. Otherwise, LEGO-Fi discards the samples and processes the next received signals.

1) DownSampling: The duration time of the WiFi preamble is 16μs and the sampling rate is 20MHz at a WiFi receiver, which means that the WiFi packet preamble detection module processes only 320 samples at once. But the duration of a complete ZigBee preamble is 128μs. Hence, only 1/8 of the ZigBee preamble can be processed in the WiFi packet detection window T_1, which causes information loss. The processing result of a ZigBee preamble in T_1 is shown in Figure 7(a). We cannot find any periodicity.

To detect the periodicity, more information needs to be included into the WiFi detection window. Therefore, we adopt downsampling and reduce the sampling rate to 2.5MHz to capture the whole ZigBee preamble. The periodicity is easy to find in the processing results, as shown in Figure 7(c).

Whereas, such a low rate will result in the distortion of ZigBee signals. According to the Nyquist Theorem, the sampling rate must be at least twice of the highest analog frequency component to fully recover the signal. A sampling rate of 2.5MHz cannot recover the ZigBee signals whose highest rate is 2MHz. Hence, we adjust the sampling rate to 4MHz. During the WiFi detection window, 5/8 of the ZigBee preamble can be captured. The processing results when downsampling to 4MHz are shown in Figure 7(b). We can find even though the periodic preamble is not completely included, it is feasible to identify the periodicity.

2) Periodicity Detection: The downsampled samples are sent to the WiFi short preamble detection module to examine the existence of periodicity. The delay correlation algorithm adopted by the WiFi short preamble detection module can be reused. If there is a ZigBee packet, the 320-sample detection window consists of five repeated symbol "0000" and each symbol has 64 samples. The delay correlation result of ZigBee

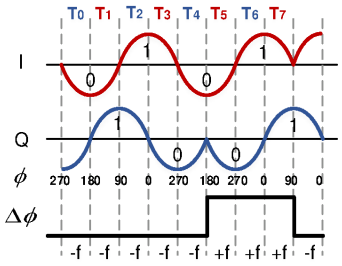


Fig. 8: The phase shift of OQPSK signals

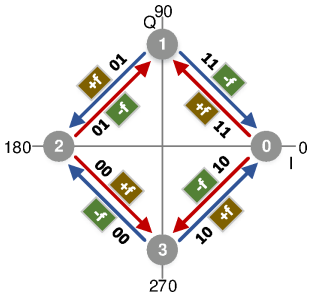


Fig. 9: OQPSK decoding

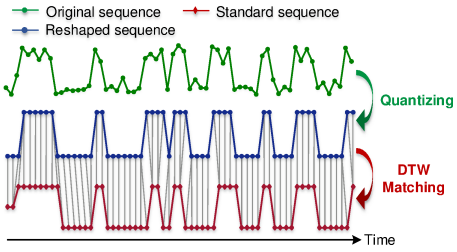


Fig. 10: Matching filter based decoding

preamble is:

$$c_n = \sum_{k=0}^{l} r_{n+k} r^*_{n+k+16} = \sum_{k=0}^{l} r_{n+k+64} r^*_{n+k+16+64} = c_{n+64} \quad (1)$$

where r_n is the received signal and r_{n+16} is the signal delayed by 16 samples. l is the length of the moving window, which is 64 for common WiFi receivers. Since the repeated period of ZigBee preamble is also 64, c_n, the result of performing the WiFi short preamble detection module on ZigBee preamble should also be periodic, as shown in Figure 4(a). We then feed the sequence of c_n into the WiFi short preamble detection module again to examine the existence of periodicity. The processing result should be:

$$s_n = \sum_{k=0}^{l} c_{n+k} c^*_{n+k+16} \quad (2)$$

Figure4(b) presents the processing result of the c_n sequence in Figure 4(a).

Specifically, LEGO-Fi adopts uniformly-spaced sampling to implement downsampling and extracts one sample from every five received samples. When the number of extracted samples cumulates to 320, the WiFi detection window size, LEGO-Fi reuses the WiFi short preamble detection twice to get s_n. If there is a subsequence of s_n and each value in the subsequence is larger than the threshold, LEGO-Fi determines there are ZigBee signals. The threshold is empirically set at 0.5.

C. Symbol Synchronization

Symbol synchronization is essential for a communication system to get the start and the end of a symbol. WiFi mainly relies on the long training sequence for symbol synchronization. Due to the difference in packet formats of ZigBee and WiFi, the synchronization mechanism at WiFi receiver doesn't work for ZigBee packets. Even though ZigBee packets do not have the long training sequence, it has the SFD to indicate the start of the packet. The SFD is defined as '0xA7". Because of the DSSS mechanism, the symbol "0xA7" is mapped to a specific chip sequence, as shown in Figure 5(a). LEGO-Fi leverages the certainty of SFD to locate it. LEGO-Fi calculates the moving correlation between the known SFD chip sequence and the received signal, as follows.

$$corr_i = \sum_{k=0}^{127} r_{i+k} x^*_k, i = 1, 2, ..., n - 127 \quad (3)$$

where x_k is the template SFD chip sequence, and r_n is the received signal. The length of moving correlation window is 128, equal to the size of SFD. As shown in Figure 5(a), the start of SFD is the position with the maximum value of the correlation results. If the maximum value, $corr_{max}$, is larger than a threshold, the SFD can be accurately located to the max-th position. If no $corr_i$ is larger than the threshold, the SFD localization fails and LEGO-Fi will drop the packet. The threshold is empirically set at 20.

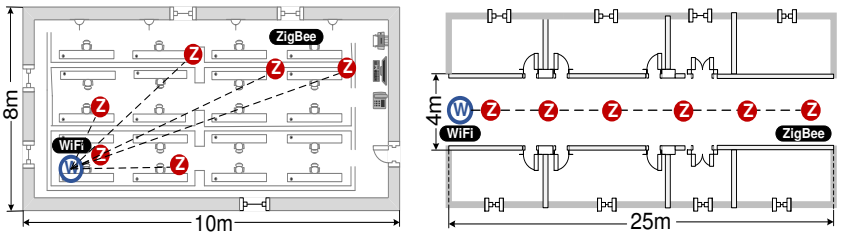
The above process can be realized by the WiFi long preamble detection module. The SFD sequence needs to be added as the template sequence.

D. ZigBee Data Decoding

Existing demodulation methods supported by WiFi cannot decode OQPSK signals due to the undesirable time delay and phase offset between the I-phase and Q-phase. Even though directly decoding ZigBee packets is infeasible, a ZigBee packet leaves distinguishable features when passing the WiFi modules. Hence, we translate the ZigBee decoding problem into the pattern identification problem.

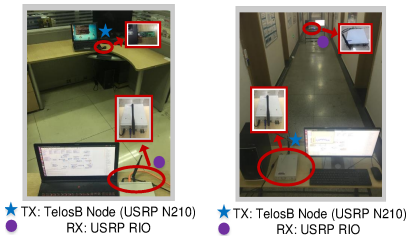
LEGO-Fi leverages the featured phase shift sequences to identify the ZigBee symbols, by a matching filter based on Dynamic Time Wrapping (DTW). The phase shift sequences can be obtained from the WiFi quadrature demodulator. Then LEGO-Fi decodes the ZigBee signals by demapping from the symbols to the bits.

1) Phase Shift Sequence Acquisition: We use an concrete example to show how LEGO-Fi leverages the WiFi quadrature demodulator to obtain the phase shift sequence of OQPSK signals. Suppose the I-Q chip set is $\{01, 10, 00, 11\}$ with the half-sine shaping, as shown in Figure 8. There are eight time slots and the duration of each slot is $0.5\mu s$, including four even time slots T_0, T_2, T_4, T_6 and four odd time slots T_1, T_3, T_5, T_7. The time offset between I-phase and Q-phase is also $0.5\mu s$. k is the sign of phase shift sequence $\Delta\phi$, which is determined



(a) Lab (b) Hallway

Fig. 11: The floor plans of the lab and the hallway



(a) Lab (b) Hallway

Fig. 12: Experiment settings in the lab and the hallway

by the value of I-phase and Q-phase. k can be calculated by:

$$k = \begin{cases} I \oplus Q & even\ time\ slot \\ not \quad (I \oplus Q) & odd\ time\ slot \end{cases} \tag{4}$$

Suppose the carrier frequency is f_c, the sampling frequency is f_s, and the initial phase is ϕ_0, then the input complex signal is $r(n) = Ae^{j2\pi(\frac{f_c}{f_s}n+\phi_0)}$. We can use WiFi quadrature demodulator to obtain the phase shift sequence according to the following equation.

$$\begin{aligned} \Delta\phi &= angle(r(n) * r(n-1)^*) \\ &= angle(Ae^{j2\pi(\frac{f_c}{f_s}n+\phi_0)}\overline{Ae^{j2\pi(\frac{f_c}{f_s}(n-1)+\phi_0)}}) \\ &= \frac{f_c}{f_s} \end{aligned} \tag{5}$$

2) Phase Shift Sequence Identification : To decode the OQPSK symbols, both the phase shift $\Delta\phi$ and the phase state ϕ are required. As shown in Figure 9, there are four phase states ϕ, which are $\{0(0^o), 1(90^o), 2(180^o), 3(270^o)\}$. The transition of phase state and phase shift represents data bits of I-phase and Q-phase. The outer blue arrows and the inner red arrows apply to the transitions of even and odd time slots, respectively. For example, in the time slot T_1, the phase state is "2" ($\phi = 180°$) and the phase shift $\Delta\phi$ is $-f$, as shown in Figure 8. Since T_1 is the odd time slot, we leverage the inner arrows to find the IQ tuple set is 01. From the above example, we can learn that only obtaining the phase shift $\Delta\phi$ is not enough and the phase state ϕ is necessary. Unfortunately, the existing WiFi modules cannot obtain the phase state and therefore fail to decode OQPSK signals.

We find the phase shift sequences of different symbols are different. Therefore, we identify the featured phase shift sequences to infer the corresponding symbols. We propose the matching filter based decoding to identify the phase shift sequence and then decode the ZigBee data bits according to the mapping between the phase shift sequences and symbols.

First, LEGO-Fi reuses WiFi quadrature demodulator to calculate the product of the signal and the conjugate delayed signal. There is no need to separate I/Q signals. Then LEGO-Fi divides the phase shift sequence into segments with a length of 64 samples. Each segment corresponds to a ZigBee symbol to be decoded. To remove the influence of dynamic channel condition, LEGO-Fi quantizes the received ZigBee samples of each segment based on a threshold filter. When the value of phase shift is greater than or equal to 0, the phase shift is quantized to 1. Otherwise, it is quantized to -1.

LEGO-Fi then leverages the matching filter to find which standard phase shift sequence is the most similar one with the current quantized phase shift segment. Due to the distortion caused by channel noise, the received phase shift can contain errors. Therefore, we use DTW to measure the similarity between the standard phase shift sequence and the received phase shift sequence. Figure 10 shows an example. Suppose that the 16 standard phase shift sequences are $\{Y_1(n), Y_2(n), ..., Y_{16}(n)\}$. For the j-th received phase shift segment, we obtain a cost set $\{cost_1^j, cost_2^j, ..., cost_{16}^j\}$ after DTW, where $cost_i^j$ is the similarity between the i-th standard phase shift sequence and the j-th received phase shift segment. The sequence with the minimal matching cost is the most similar sequence. Therefore we can find the corresponding $symbol$ according to the following equation:

$$symbol = Index(min(cost_i), i = 1, 2, ..., 16) \tag{6}$$

After learning the symbol id, LEGO-Fi can infer the four data bits based on the mapping relationship between symbols and data bits. In this way, LEGO-Fi achieves the ZigBee reception at WiFi successfully at the physical level.

In order to achieve bi-directional communication between WiFi and ZigBee, the transmission from WiFi to ZigBee also need to be implemented. WEBee is a representative physical-level CTC work from WiFi to ZigBee. Therefore, WEBee and LEGO-Fi can collaborate to achieve the bi-directional communications.

E. Discussion

With the proliferation of IoT devices in all kinds of scenarios, we may foresee the ever increasing needs of CTC in wireless systems. Supporting CTC with commercial wireless devices is very likely to be a default practice in the near future. As a software implementation on USRP, LEGO-Fi maximizes the reuse of standard WiFi modules and sheds lights on future hardware implementation of CTC-compatible WiFi devices. Note that in the design of LEGO-Fi, standard WiFi modules are reused as much as possible. As long as the

basic root privilege of each module (i.e. the interoperability of the interface and the register) is provided, LEGO-Fi can decode ZigBee signals by rewiring the WiFi modules and adjusting the control logic of the data flow. According to the scheme of LEGO-Fi, the current design of WiFi devices can obtain the new ability of CTC, with limited modification to the circuit, while all the functions and properties of a WiFi device are completely preserved.

V. Evaluation

In this section, we conduct extensive experiments to evaluate the performance of LEGO-Fi. The LEGO-Fi transmitter is the commercial TelosB node with CC2420 and the prototype of LEGO-Fi is implemented on a USRP-RIO platform.

A. Experiment Setup

For comparison, we implement two state-of-the-art works, FreeBee [13] and ZigFi [6]. FreeBee embeds CTC symbols by the shifting transmission timing of a beacon. FreeBee detects the packet by received signal strength (RSS) and uses folding to determine the shifts of the transmission timings. In the default setting of FreeBee, a 6-bit symbol can be detected when folding five beacons. The beacon rate is 20 beacon/second, as same as the default setting of FreeBee in [13]. ZigFi piggybacks ZigBee packets over WiFi data packets and leverages the changes of Channel State Information (CSI) to convey CTC symbols. The CSI variations of eight packets can encode 1-bit symbol. Since the implementation of ZigFi relies on an existing WiFi link, a commercial WiFi device is used to transmit WiFi packets with a packet length of 145 bytes. The data packet rate is set to 2000 packets/second, as same as the default setting in [6]. Another WiFi device installed Intel 5300 NIC and CSITool software platform is used as the ZigFi receiver with the sampling rate of 2KHz.

Unless otherwise specified, the transmission power of TelosB node is set to 0 dBm. The channel is set to ZigBee channel 23 and the receiving channel is set to WiFi channel 11. In LEGO-Fi, the packet size is 24 bytes. The payload covers all the 16 ZigBee symbols and the number of each symbol is equal. The experiments are conducted in two environments, our lab on campus and the hallway of a building. The floor plans are shown in Figure 11(a) and Figure 11(b), and the corresponding settings are shown in Figure 12(a) and Figure 12(b).

B. Overall Performance Comparison

First, we analyze and compare the theoretical capacity of LEGO-Fi and two the-state-of-art works, FreeBee [13] and ZigFi [6]. The capacity of FreeBee depends on the beacon rate. In the default setting of FreeBee, the beacon rate is 20 beacons/second and five beacons can modulate six bits. Therefore, the theoretical capacity of FreeBee = 24bps = 20 beacons/second $\times$ 6/5 bits/beacon. For ZigFi, the capacity is decided by the packet transmission rate. In the default setting of ZigFi, the packet transmission rate is 2000 packets/second and eight packets can modulate one bit. Hence, the theoretical capacity of ZigFi = 250bps = 2000 packets/second $\times$ 1/8 bits/beacon. Since LEGO-Fi is a physical-level CTC that allows the ZigBee transmitter sends standard ZigBee packet, the theoretical capacity of LEGO-Fi is same with the capacity of ZigBee, which is 250Kbps.

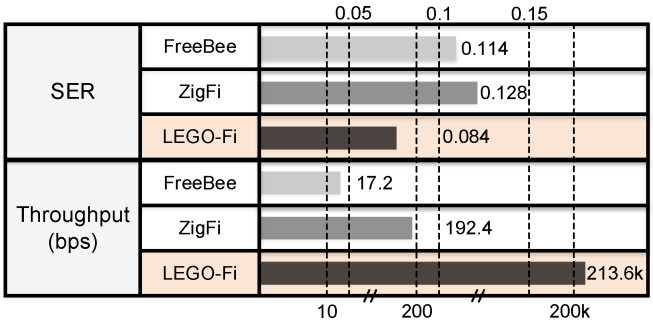


Fig. 13: Overall performance comparison

We then conduct experiments to compare the performance of three works in practice. The distance between the ZigBee sender and the WiFi receiver is 5m. All settings are same with the default settings. The experiments of three works are conducted in our lab, with a consistent ambient environment and a similar network interference condition.

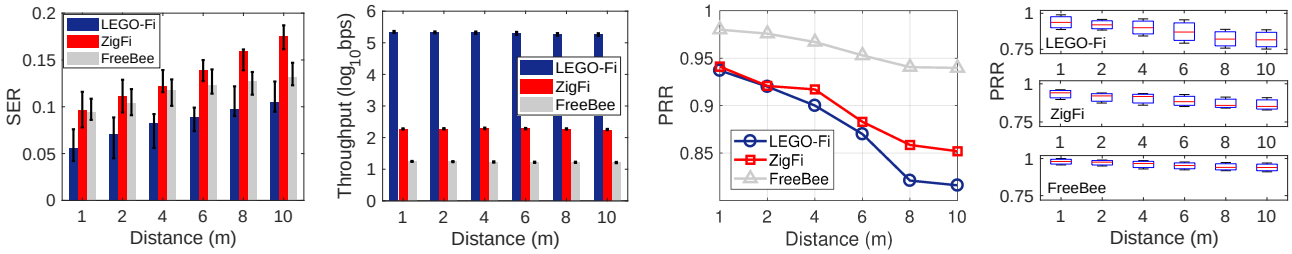
The comparison results are shown in Figure 13. FreeBee achieves a throughput of 17.2bps with a Symbol Error Rate (SER) of 0.114. ZigFi achieves a throughput of 192.4bps with a SER of 0.128. The measured performance of both FreeBee and ZigFi are close to the reported. The throughput of LEGO-Fi is up to 213.6Kbps, which is 13000$\times$ and 1200$\times$ higher than that of FreeBee and ZigFi, respectively. The SER of LEGO-Fi is 0.084, which is lower than FreeBee and ZigFi, revealing the physical-level manipulation is more reliable than packet-level controls. The results demonstrate the effectiveness of LEGO-Fi and the efficiency of implementing CTC in the physical level.

C. Performance under Different Settings

In this subsection, we vary the distance between the sender and the receiver, the transmission power, the payload length, and operating environments to study their impacts on LEGO-Fi in terms of Packet Reception Ratio (PRR), Symbol Error Ratio (SER), and throughput. We also evaluate LEGO-Fi in mobile scenarios.

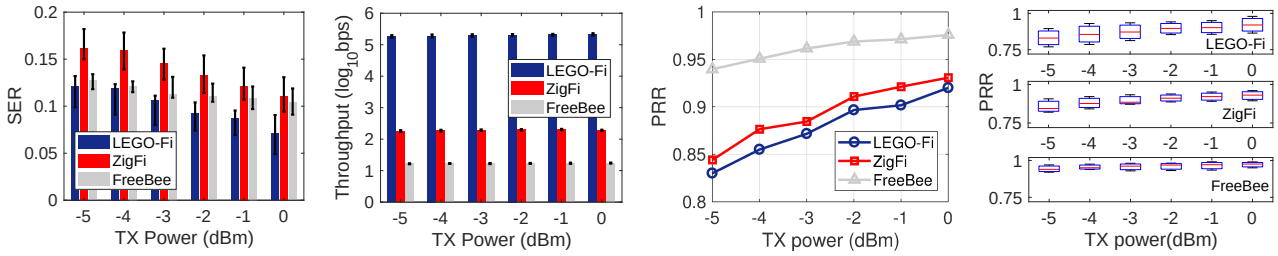
1) Impact of Distance: We study the impact of distance between the sense and the receiver on the overall performance of LEGO-Fi, taking all the components into consideration. We conduct this experiment in the lab and vary the distance between the sender and the receiver from 1m to 10m, as shown in Figure 11(a).

Figure 14(a) and Figure 14(b) show the SER and throughput, respectively. We can find that LEGO-Fi achieves a SER lower than the other two methods. This is because manipulating signals at the physical level is more reliable than controlling the packet behavior at the packet level. To present



(a) SER (b) Throughput (c) PRR (d) PRR variation

Fig. 14: Performance comparison with different distances between the ZigBee sender and the WiFi receiver



(a) SER (b) Throughput (c) PRR (d) PRR variation

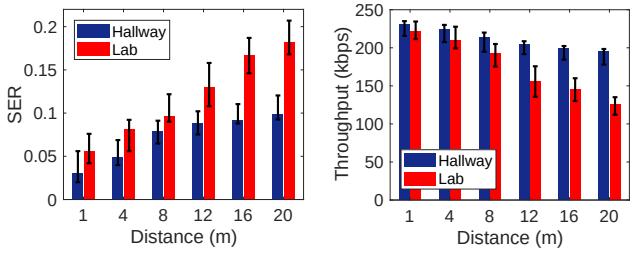
Fig. 15: Performance comparison with different transmission power of the ZigBee sender

the results clearly, we show the logarithm of throughput in Figure 14(b). LEGO-Fi's ability of decoding standard ZigBee packets enables a high throughput. As expected, LEGO-Fi achieves a high throughput of 201.2Kbps, which is much higher than that of the other two methods.

Figure 14(c) and Figure 14(d) present the PRR and PRR variations respectively. The PRR of all there works decreases with the increase of the distance. Besides, we can find that the PRR of LEGO-Fi is a little lower than the other two methods. This is because detecting the ZigBee preamble with existing WiFi modules in LEGO-Fi is harder than detect the energy of transmissions in FreeBee or the CSI variations in ZigFi at the the packet level. Besides, both FreeBee and ZigFi have reliability enhancement mechanisms. If a higher PRR is desired, users can enhance LEGO-Fi with the retransmission. Based on our empirical results, two retransmissions are good enough to provide a 0.99 reliability and the throughput is still much higher than the-state-of-the-arts.

2) Impact of Transmission Power: We then study the impact of transmission power. We conduct the experiments in the lab shown in Figure 12(a) and we change the transmission power of the ZigBee transmitter from -5dBm to 0dBm. The distance between the ZigBee sender and WiFi receiver is 2m.

The transmission power of ZigBee sender affects the received signal strength. Hence, we observe results similar to the impact of the distance. From the results in Figure 15(a), we can find the average SER of LEGO-Fi is 0.099, which is still the lowest among the three methods. When the transmission power is -5dBm, the throughput of LEGO-Fi, ZigFi, and FreeBee are 182.3Kbps, 176.9bps, and 17.5bps respectively.



(a) Lab (b) Hallway

Fig. 16: LEGO-Fi performance in different environments

The throughput of LEGO-Fi increases from 182.3Kbps to 212.1Kbps when the transmission power varies from -5dBm to 0dBm, as shown in Figure 15(b). As expected, the PRR of all three methods increases with the increase of the transmission power, as shown in Figure 15(c). The results of PRR and PRR variation in Figure 15(c) and Figure 15(d) are consistent to the results in Figure 14(c) and Figure 15(d) because the transmission power also affects the received signal strength.

The received signal strength depends on both the transmission power and the distance between the transmitter and receiver. We evaluate the PRR of LEGO-Fi under different transmission powers and distances. We can find the PRR varies in the range of 0.778 and 0.937. Even when the transmission is low (-5dBm) and the distance is far (10m), the PRR of LEGO-Fi is still above 0.77.

3) Impact of Environment: We evaluate LEGO-Fi in different environments. We conduct the experiments in another

larger lab and the hallway Figure 16(a) and Figure 16(b) present the SER and the throughput of LEGO-Fi in two environments, respectively. We can find that when the distance increases, the SER increases and the throughput degrades in both environments. When the distance is 20m, the SER in the hallway and the lab is 0.098 and 0.182 respectively. When the distance is 20m, the throughput of LEGO-Fi in the hallway is 194.8Kbps and only 125.1Kbps in the lab. The increasing speed of the SER and dropping speed of the throughput in the lab are much faster than the ones in the hallway. This is because the environment in the lab is more complicated than that in the hallway, leading to more serious multipath influences on the received signals.

VI. Conclusion

CTC has great potential in IoT applications. Our work focuses on "the shortest plank" of CTC, i.e. communication from ZigBee to WiFi. Our proposal called LEGO-Fi unleashes the full communication capacity from ZigBee to WiFi. LEGO-Fi is a transmitter-transparent CTC, which leaves the processing complexity at the receiver. LEGO-Fi reuses the standard WiFi modules and presents cross-demapping to decode ZigBee signals. The main take-away from this paper is two-folded: (1) the asymmetry in devices' capacity offers extra flexibility to process CTC signals; (2) besides packet-level metrics, the unique protocol features also provide important basis in building the side channel for CTC. In the future, we will also try to deploy LEGO-Fi in real-world applications.

Acknowledgment

This work was supported by National Key R&D Program of China No. 2017YFB1003000, National Nature Science Foundation of China No. 61772306 and No. 61672320.

References

[1] Z. An, L. Yang, and Q. Lin. Cross-frequency communication: Near-field identification of uhd rfids with wifi! In *MobiCom*. ACM, 2018.

[2] B. Bloessl, S. Joerer, F. Mauroner, and F. Dressler. Low-cost interferer detection and classification using telosb sensor motes. In *MobiCom*. ACM, 2012.

[3] K. Chebrolu and A. Dhekne. Esense: communication through energy sensing. In *MobiCom*. ACM, 2009.

[4] Z. Chi, Y. Li, H. Sun, Y. Yao, Z. Lu, and T. Zhu. B2w2: N-way concurrent communication for iot devices. In *SenSys*. ACM, 2016.

[5] Z. Chi, Y. Li, Y. Yao, and T. Zhu. Pmc: Parallel multi-protocol communication to heterogeneous iot radios within a single wifi channel. In *ICNP*. IEEE, 2017.

[6] X. Guo, Y. He, X. Zheng, L. Yu, and O. Gnawali. Zigfi: Harnessing channel state information for cross-technology communication. In *INFOCOM*. IEEE, 2018.

[7] X. Guo, X. Zheng, and Y. He. Wizig: Cross-technology energy communication over a noisy channel. In *INFOCOM*. IEEE, 2017.

[8] A. Hithnawi, H. Shafagh, and S. Duquennoy. Tiim: technology-independent interference mitigation for low-power wireless networks. In *IPSN*. ACM, 2015.

[9] S. Hu, S. Yao, H. Jin, Y. Zhao, Y. Hu, X. Liu, N. Naghibolhosseini, S. Li, A. Kapoor, W. D. andLu Su, A. Bar-Noy, P. Szekely, R. Govindan, R. Hobbs, and T. F. Abdelzaher. Data acquisition for real-time decision-making under freshness constraints. In *RTSS*. IEEE, 2015.

[10] W. Jiang, S. M. Kim, Z. Li, and T. He. Achieving receiver-side cross-technology communication with cross-decoding. In *MobiCom*. ACM, 2018.

[11] W. Jiang, Z. Yin, S. M. Kim, and T. He. Transparent cross-technology communication over data traffic. In *INFOCOM*. IEEE, 2017.

[12] M. Jin, Y. He, X. Zheng, Y. He, D. Fang, D. Xu, T. Xing, and X. Chen. Smoggy-link: Fingerprinting interference for predictable wireless concurrency. In *ICNP*. IEEE, 2016.

[13] S. M. Kim and T. He. Freebee: Cross-technology communication via free side-channel. In *MobiCom*. ACM, 2015.

[14] Z. Li and T. He. Webee: Physical-layer cross-technology communication via emulation. In *MobiCom*. ACM, 2017.

[15] Z. Li, W. Jiang, and T. He. Bluebee: Physical-layer cross-technology communication via emulation. In *SenSys*. ACM, 2017.

[16] Z. Li, Y. Xie, L. Mo, and K. Jamieson. Recitation:rehearsing wireless packet reception in software. In *MobiCom*. ACM, 2015.

[17] C. J. M. Liang, K. Chen, N. B. Priyantha, J. Liu, and F. Zhao. Rushnet: practical traffic prioritization for saturated wireless sensor networks. In *SenSys*. ACM, 2014.

[18] C. Ma, J. Liu, X. Tian, H. Yu, Y. Cui, and X. Wang. Interference exploitation in d2d-enabled cellular networks: A secrecy perspective. In *TCOM*. IEEE, 2015.

[19] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, V. Bahl, and I. Stoica. Low latency geo-distributed data analytics. In *SIGCOMM*. ACM, 2015.

[20] S. Rayanchu, A. Patro, and S. Banerjee. Airshark: detecting non-wifi rf devices using commodity wifi hardware. In *IMC*. ACM, 2011.

[21] A. Saifullah, M. Rahman, D. Ismail, C. Lu, R. Chandra, and J. Liu. Snow: Sensor network over white spaces. In *SenSys*. ACM, 2016.

[22] K. Sundaresan, S. V. Krishnamurthy, X. Zhang, A. Khojastepour, and S. Rangarajan. Trinity: A practical transmitter cooperation framework to handle heterogeneous user profiles in wireless networks. In *MobiHoc*. ACM, 2015.

[23] S. Wang, S. M. Kim, and T. He. Symbol-level cross-technology communication via payload encoding. In *ICDCS*. IEEE, 2018.

[24] Y. Yan, P. Yang, X. Li, T. Yue, L. Zhang, and L. You. Zimo: building cross-technology mimo to harmonize zigbee smog with wifi flash without intervention. In *MobiCom*. ACM, 2013.

[25] Z. Yin, W. Jiang, S. M. Kim, and T. He. C-morse: Cross-technology communication with transparent morse coding. In *INFOCOM*. IEEE, 2017.

[26] Z. Yin, Z. Li, S. M. Kim, and T. He. Explicit channel coordination via cross-technology communication. In *MobiSys*. ACM, 2018.

[27] T. Zachariah, B. Campbell, J. Adkins, N. Jackson, and P. Dutta. The internet of things has a gateway problem. In *HotMobile*. ACM, 2015.

[28] W. Zeng, A. Arora, and K. Srinivasan. Low power counting via collaborative wireless communications. In *IPSN*. ACM/IEEE, 2013.

[29] X. Zhang and G. S. Kang. Enabling coexistence of heterogeneous wireless systems:case for zigbee and wifi. In *MobiHoc*. ACM, 2011.

[30] X. Zhang and K. G. Shin. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *INFOCOM*. IEEE, 2013.

[31] Y. Zhang and Q. Li. Howies: A holistic approach to zigbee assisted wifi energy savings in mobile devices. In *INFOCOM*. IEEE, 2013.

[32] Z. Zhao, W. Dong, G. Chen, G. Min, T. Gu, and J. Bu. Embracing corruption burstiness: Fast error recovery for zigbee under wi-fi interference. *TMC*, 16(9):2518–2530, 2017.

[33] X. Zheng, Z. Cao, J. Wang, Y. He, and Y. Liu. Zisense: towards interference resilient duty cycling in wireless sensor networks. In *SenSys*. ACM, 2014.

[34] X. Zheng, Y. He, and X. Guo. Stripcomm: Interference-resilient cross-technology communication in coexisting environments. In *INFOCOM*. IEEE, 2018.

[35] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma. Zifi: wireless lan discovery via zigbee interference signatures. In *MobiCom*. ACM, 2010.