General and Offset-Resistant Physical-Layer Acknowledgement Approach for Cross-Technology Communication

Shumin Yao, Qinglin Zhao, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Li Feng,
Peiyun Zhang, *Senior Member, IEEE*, and Aiiad Albeshri

***Abstract*— Cross-technology communication (CTC) enables direct communications among devices with heterogeneous wireless technologies (e.g., Bluetooth, WiFi, ZigBee, etc.), thereby reducing the cost and complexity of their interconnections. However, CTC is essentially unreliable due to the technology heterogeneity, and most existing CTC designs do not provide acknowledgment (ACK) feedback to ensure reliable data transmission. Few ACK designs are only applicable to feedback for ZigBee-WiFi pair and are vulnerable to sampling offsets that inherently exist in CTC. In this work, we propose a General and Offset-resistant Physical-layer ACK approach, called GOP-ACK, to support reliable feedback with practical considerations. Its core idea lies in encoding ACK messages with offset-resistant signal that has two benefits: 1) it can be adapted to a wide range of CTC scenarios with minimal adjustment, and 2) it can be effortlessly and robustly detected even in the presence of sampling offsets. With GOP-ACK, we provide a general framework with easy-to-follow design steps allowing it to be applied to specific CTC cases. We demonstrate its generality by presenting two specific design cases: ZigBee-to-BLE and ZigBee-to-WiFi feedback and propose a theoretical model to analyze its performance. We then conduct real-world experiments and extensive simulations to verify its feasibility and superiority over the state of the art, thereby enhancing the practicality of CTC greatly.**

***Index Terms*— ACK, Cross-Technology Communication, Offset-Resistant.**

I. Introduction

A. Background

Cross-technology communication (CTC) is a technique that aims to enable direct communication among devices with heterogeneous wireless technologies (e.g., Bluetooth, WiFi, ZigBee, etc.). It significantly simplifies the complex heterogeneous connections and triggers a wide range of novel Internet of Things (IoT) applications. Among them, point-to-point (P2P) applications are particularly noteworthy due to their integral role in various smart systems [1]–[4]. For example, a ZigBee smart speaker may broadcast the temperature detected by a Bluetooth Low Energy (BLE) sensor, or a smart air conditioner equipped with a ZigBee radio could adjust room temperatures based on the instructions sent from a WiFi or Bluetooth device (e.g., a mobile phone).

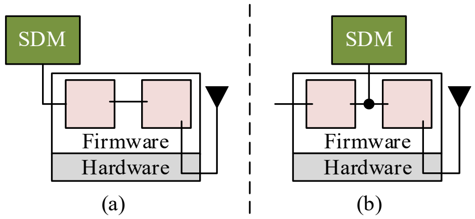


Fig. 1. CTC schemes can be classified as (a) no-firmware-modifying and (b) firmware-modifying schemes.

The implementation method of CTC has evolved with the progress of the physical-layer firmware, an embedded program controlling the operations of radio hardware (e.g., the processing of bits and signals [5], [6]). Historically, around 2009-2018, the pioneering studies focused on enabling CTC without modifying the firmware (e.g., [7]–[14]), as the firmware was unmodifiable during that period. Since 2017, with the emergence of modifiable firmware, such as that supporting software-defined radios (SDRs), the mainstream method of achieving CTC has shifted towards firmware modification (e.g., [15]–[25]). To reflect this evolution, we classify existing schemes into two types:

- *No-firmware-modifying scheme*. It connects software-defined modules (SDMs) running specific algorithms to the external interfaces (i.e., input or output interfaces) of firmware to achieve CTC, as shown in Fig. 1(a). Despite its low computational efficiency and limited functionality [26]–[28], it has laid the foundation for advanced CTC.
- *Firmware-modifying scheme*. It connects SDMs to the internal interfaces of the firmware to achieve CTC, as shown in Fig. 1(b). It provides computationally efficient and flexible CTC services that the no-firmware-modifying scheme wishes to offer but cannot, thereby dominating recently.

B. Motivations

Unfortunately, both types of CTC schemes are inherently unreliable, leading to sometimes malfunctions in CTC-based IoT applications. The root cause of this unreliability lies in the

S. Yao is with the School of Computer Science and Engineering, Macau University of Science and Technology, Avenida Wei Long, Taipa, Macau, China, and also with the Department of Broadband Communication, Peng Cheng Laboratory, Shenzhen 518066, China (email: zhysm@outlook.com).

Q. Zhao and L. Feng (corresponding author) are with the School of Computer Science and Engineering, Macau University of Science and Technology, Avenida Wei Long, Taipa, Macau, China (qlzhao@must.edu.mo and lfeng@must.edu.mo).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (email: zhou@njit.edu).

P. Zhang is with the Engineering Research Center of Digital Forensics of Ministry of Education, and the School of Computer Science, Nanjing University of Information Science & Technology, Nanjing 210044, China (email: zpy@nuist.edu.cn; 20211220029@nuist.edu.cn).

Aiiad Albeshri is with the Department of Computer Science, King Abdulaziz University, Jeddah 21481, Saudi Arabia (aaalbeshri@kau.edu.sa).

inherent hardware and standard mismatch between a sender and receiver [13], [29], [30].

Given the inherent unreliability, it is crucial for CTC to have an acknowledgment (ACK) feedback mechanism to allow a sender to be aware of transmission failures and perform timely retransmissions. However, there is a lack of sufficient ACK studies for both types of CTC schemes [31]. Few existing ACK studies, such as [18], [19], [26], [32], [33], have made certain progress in achieving lightweight CTC feedback by proposing physical-layer ACK designs. Nonetheless, these designs have two major limitations:

1. *Lack of generality.* The existing ACK designs are only applicable to feedback for the ZigBee-WiFi pair and lack a general framework to extend them to cases having different heterogeneous technology pairs.
2. *Vulnerable to sampling offsets.* Sampling offsets are inherent in CTC due to the poor synchronization between sender and receiver caused by their hardware and standard mismatch [9], [20]. Existing ACK designs do not account for the adverse impact of these offsets on ACK detection and therefore are often unreliable.

In summary, there is an urgent need for a general and offset-resistant physical-layer ACK method for both types of CTC schemes, which motivates this study. To contribute to the advancement of CTC, we primarily focus on the more promising firmware-modifying scheme, while briefly presenting our design for the no-firmware-modifying scheme.

C. Our contributions

This work proposes a general and offset-resistant ACK (GOP-ACK) approach for CTC feedback. While the primary focus is on a P2P scenario, the approach can be extended to the scenarios of multiple device coexistence (see the Supplementary File of this paper [34]). This study tends to make the following novel contributions to the era of CTC.

1. It proposes offset-resistant signal (ORS) and its use for robustly conveying ACK messages across heterogeneous devices. An ORS comprises two identical waves with lengths equal to the sampling interval of an ACK receiver. This design allows the receiver to always sample each wave at the same position and obtain a phase shift of 0 even if the sampling is offset, thus achieving robust detection. Importantly, the ORS design is general because its generation and detection require mainly a modulator and a phase-shift calculator, respectively, which are modules commonly found in the firmware of a wide range of wireless technologies, e.g., BLE [9], ZigBee [10], and WiFi [17].
2. It proposes a general framework with clear and easy-to-follow guidelines to achieve GOP-ACK feedback with a wide application scope of CTC. The guidelines include constructing practical ORS when the sender's default wavelength does not match with the receiver's default sampling interval, achieving robust and low-cost ACK transmission under non-ideal channel conditions, and reusing modulators and phase-shift calculators embedded in the firmware. The feasibility of the guidelines is then examined by ZigBee-to-BLE and ZigBee-to-WiFi feedback case studies.
3. It proposes a theoretical model to analyze one successful ACK decoding probability and complete transmission time of GOP-ACK. This model takes various design parameters into account and, hence, can be used to provide theoretically-guided parameter settings.
4. It conducts real-world experiments and extensive simulations to verify that GOP-ACK is feasible and outperforms the state of the art significantly and validates the accuracy of the proposed theoretical model.

The outcome of this study is expected to put CTC into industrial use.

The rest of the paper is organized as follows. We outline existing CTC designs and ACK designs for CTC in Section II and propose GOP-ACK in Section III. We then apply it to design ZigBee-to-BLE and ZigBee-to-WiFi feedback in Sections IV and Supplementary File of this paper [34]. We model the performance of GOP-ACK in Section V. We evaluate the proposed design and make a performance comparison with the state of the art in Section VI. We conclude this paper in Section VII.

II. Related work

This section sequentially introduces existing CTC designs for data transmissions and ACK designs for CTC.

A. Existing CTC designs

Most existing CTC designs primarily focus on sender-to-receiver communications, without ACK feedback. They fall into two classes.

1) *No-firmware-modifying CTC scheme.* This scheme achieves CTC by deploying SDMs to process the input of transmitting (TX) firmware or output of receiving (RX) firmware while keeping firmware intact. There are three types of SDM deployment.
   a) TX-side: It deploys an SDM to input a special bit sequence to TX firmware to emulate packets that are compatible with and can be received by other technologies. This deployment enables WiFi to emulate ZigBee packets [10], [27], [29], [30], [35]–[38], BLE packets [12], [39], [40], LoRa packets [41], or signal pulses [42], enables BLE to emulate LoRa packets [43] or ZigBee packets [11], and enables cellular devices to emulate BLE packets [13].
   b) RX-side: It deploys an SDM to further process the output of RX firmware such that a receiver can receive and decode other technologies' packets. This deployment enables ZigBee to decode WiFi packets [14], enables BLE to decode WiFi packets [44], [45] and ZigBee packets [9], and enables WiFi to decode ZigBee packets [46].
   c) Both-side: Both the TX-side and RX-side SDMs are deployed. For example, Yao et al. [28] deploy an SDM to the TX firmware of a WiFi-based gateway and another SDM to the RX firmware of a WiFi node, enabling the gateway to transmit WiFi packets while sending signal pulses to ultra-low power devices.

2) *Firmware-modifying CTC scheme.* This scheme achieves CTC by making slight modifications to existing firmware.

It connects SDMs to the internal interfaces of firmware without affecting the functionality of the original firmware modules. It provides computationally efficient and flexible CTC services that the no-firmware-modifying scheme aspired to offer but could not, thereby becoming mainstream recently. Two types of firmware modification have emerged. The first one is TX firmware modification. For example, a WiFi TX firmware with SDM inserted can perform simultaneous WiFi-to-ZigBee and WiFi-to-WiFi packet transmissions [15], [21]–[23]. Cho et al. [47] designed SDM to directly inject phase shift keying signals to the existing mixer on BLE TX firmware, enabling BLE to generate WiFi packets. The second one is RX firmware modification. For example, WiFi RX firmware with SDM inserted behind the phase-shift calculator can decode special BLE symbol combinations [40], special ZigBee symbol combinations [25], and even arbitrary ZigBee symbols [24], [48]. A BLE RX firmware with two SDMs inserted can achieve lightweight ZigBee packet decoding [20]. A LoRa RX firmware with an SDM inserted can decode BLE-emulated and ZigBee-emulated single-tone sine waves [16].

B. Existing ACK designs for CTC

To enhance the reliability of CTC, some suggested designing medium-access-control-layer (MAC-layer) ACK, while others proposed physical-layer ACK.

1) MAC-layer ACK

After a reception, a data receiver adopts an existing CTC technique to transmit back a sender- or receiver-compliant MAC-layer ACK packet to a data sender. For example, XBee [9] suggested that a BLE node could adopt the BlueBee [11], a BLE-to-ZigBee CTC technique, to transmit a ZigBee MAC-layer ACK packet to a ZigBee node. We note that a MAC-layer packet has a long header that consists of many fields such as "frame control" and "duration." However, for P2P applications, which are the primary focus of CTC [9], and the CTC implementations with Fountain codes [32], a short ACK message mainly comprising a destination address or 0/1 feedback is sufficient. Hence, employing a MAC-layer packet to convey an ACK message is impractical and rarely used in CTC.

2) Physical-layer ACK

The existing ACK designs for CTC focus on the physical layer and achieve lightweight feedback by encoding an ACK message into a small set of symbols. However, they lack generality and are vulnerable to sampling offsets. These designs fall into three categories based on ACK-encoding methods.

- *Energy-level ACK*. An ACK message is encoded into the specified set of symbols detectable via energy detection. For example, in [32], a ZigBee node replies a one-bit (0/1) ACK to a WiFi node, by sending out a short/long ZigBee symbol. The WiFi node identifies the ACK by energy detection. Xia et al. [33] adopt the same method to achieve WiFi-to-ZigBee feedback, they use the absence/presence of a fixed-length WiFi symbol to represent an ACK bit 0/1. These designs are not general because they are only limited to the WiFi-ZigBee pair. Moreover, they are vulnerable to the sampling offset because the detected energy values might vary with the offsets, causing errors in energy detection.
- *Symbol-level ACK*. An ACK message is encoded into special symbol combinations. For example, in [19], [26], a ZigBee receiver replies a one-bit (0/1) ACK to a WiFi node, by sending out a special symbol combination, "EF"/"67", whose waveform contains a single-tone sine-wave segment. The WiFi node first samples the sine-wave, then adopts a phase-shift calculator (usually used for detecting WiFi preambles) to detect the phase shift value between sampling instances, and finally decodes the ACK as 0/1 according to the sign (negative/positive) of the phase shift value. These designs are not general because they only utilize the unique property of the ZigBee symbols obtained by WiFi. Also, they are vulnerable to sampling offsets for two reasons. First, the sinewave is too short (i.e., $5\mu s$, which is $1/3$ of the waveform length of a ZigBee symbol) to detect in the presence of sampling offsets. Second, the signs of the detected phase shifts might vary with the sampling offsets, causing detection errors.
- *Bit-level ACK*. An ACK message is encoded into a bit sequence that will be transformed into a special set of symbols by a modulator. For example, in [18], a ZigBee receiver replies a one-bit (0/1) ACK to a WiFi node, by sending out a carefully constructed bit sequence, which is transformed into a $32\mu s$ long version of sinewave. The WiFi node decodes the ACK as in [19], [26]. This design, compared with [19], [26], extends the length of the sinewave and thereby improves the reliability of detection to some extent. However, this reliability improvement is limited since WiFi still detects ACK based on the signs of phase shifts that might vary with the sampling offsets. Moreover, like [19], [26], it still relies on the distinct features of the ZigBee-WiFi pair and therefore is not general.

In contrast to these existing designs, GOP-ACK is general because it provides a general framework with simple design steps to achieve direct feedback between many different heterogeneous devices. It combats the adverse impact of the sampling offsets in ACK detection by encoding ACK messages into specially designed offset-resistant signals.

III. GOP-ACK APPROACH

In this section, we focus on a heterogeneous ACK sender-receiver pair to present our GOP-ACK approach. The core idea of the approach is to encode ACK messages into specially designed offset-resistant signals (ORSs). We first present the ideal design of the general and reliable ORS. Then, considering non-ideal factors, we introduce an ORS-based framework for practically applying ORS-based ACK in specific CTC cases.

A. Ideal design of offset-resistant signal

In our design, an ACK sender employs its modulator to generate an ORS as an ACK message, which consists of two identical waves, as shown in Fig. 2. An ACK receiver is required to have a phase-shift calculator. By sampling the same position of the two waves of one ORS, the calculator obtains a zero phase-shift even in the presence of sampling offsets. Hence, the receiver can robustly detect an ACK message.

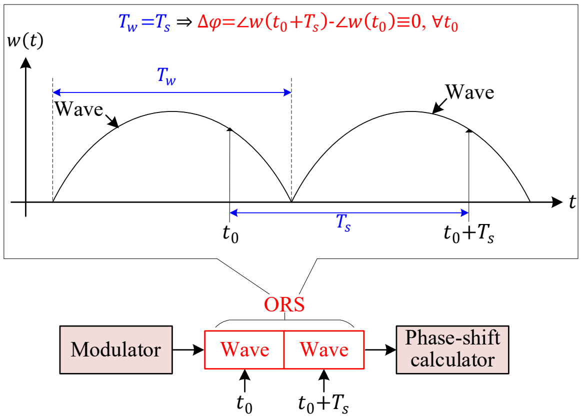


Fig. 2. The core idea of offset-resistant signal.

Specifically, let T_w denote the length of one wave generated by the sender. Let T_s denote the interval of two sampling instances that the receiver uses to calculate one phase shift. Our approach requires:

$$T_w = T_s. \tag{1}$$

Let $w(t)$ denote one ORS. Let $w(t_0)$ and $w(t_0+T_s)$ respectively denote the first and second sampling instances of $w(t)$ that the receiver uses to calculate one phase shift. Under the ideal channel conditions, the two sample instances, $w(t_0)$ and $w(t_0+T_s)$, are the same for $t_0 \in [0,\ T_s]$ according to (1) and then their phases, $\angle w(t_0)$ and $\angle w(t_0+T_s)$, are equal. As a result, the receiver can obtain a phase-shift $\Delta\varphi = \angle w(t_0+T_s) - \angle w(t_0) = 0$.

Our ORS design is general and reliable. Its generality stems from the core requirement of having a modulator and a phase-shift calculator only, which are modules commonly found in the firmware of a wide range of wireless technologies (e.g., BLE [9], ZigBee [10], WiFi [17] etc.). Its reliability stems from our ORS construction which can resist the negative impact of sampling offsets due to the zero phase-shift property.

B. ORS-based framework for practical considerations

With ORS, we aim to achieve general and offset-resistant physical-layer ACK (GOP-ACK) feedback with a wide applicable scope of CTC. To this end, we need to consider the divergence between parameter specifications of specific CTC cases and our requirement (1), non-ideal conditions, and module reusing nature of CTC. Here, we present a general framework that provides step-by-step guidelines for addressing three practical questions:

Q1: How to meet requirement (1) when the sender's default T_w does not match the receiver's default T_s?

Q2: How to achieve robust and low-cost ACK transmission under non-ideal conditions?

Q3: How to reuse the modulator and phase-shift calculator that are usually embedded in the firmware for transceiving the ORS-based ACK signals?

Below, we first present a practical construction of ORS to address *Q1*, then propose a reliable and lightweight ORS-based ACK design to address *Q2*, and finally present methods that reuse existing modules to address *Q3*.

1) Practical construction of ORS

To meet (1), we should consider the technology characteristics of both ACK sender and receiver. Let T_b be a positive integer representing the length of one basic signal unit (BSU) of an ACK sender. Assume that the ACK sender constructs one ORS wave which is comprised of α BSUs, where α is a positive integer. Then $T_w = \alpha T_b$, as shown in Fig. 3. Then consider the ACK receiver. Let T_p be a positive integer representing the default sampling interval that the phase-shift calculator of the ACK receiver computes one phase shift. Assume that $T_s = \beta T_p$, where β is a positive integer. Let $LCM(T_b, T_p)$ be the least common multiple of T_b and T_p. To ensure that $T_w = T_s$, we have

$$\alpha \cdot T_b = \beta \cdot T_p \tag{2}$$

where

$$\begin{aligned} \alpha &= LCM(T_b, T_p)/T_b \\ \beta &= LCM(T_b, T_p)/T_p. \end{aligned} \tag{3}$$

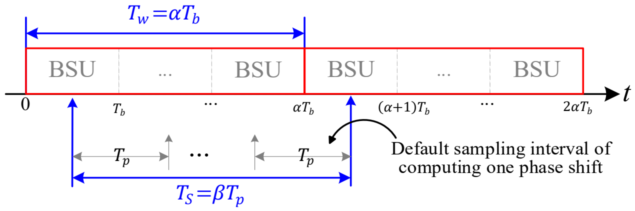


Fig. 3. Practical design of ORS considers technology characteristics, where $T_w = \alpha T_b$ and $T_s = \beta T_p$.

2) Reliable and lightweight design of ORS-based ACK

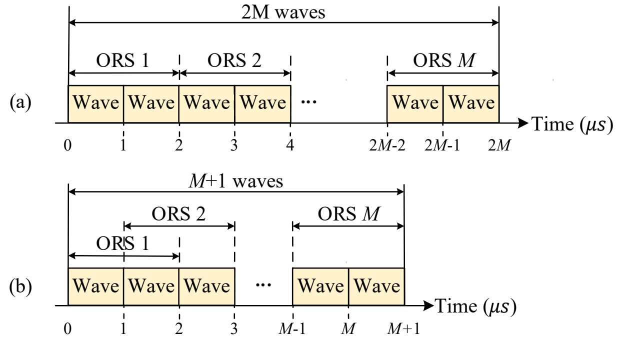


Fig. 4 An ACK signal with M ORSs constructed by (a) 2M nonoverlapping waves and (b) M+1 overlapping waves.

Encoding an ACK message into an ORS effectively combats the adverse impact of sampling offsets on ACK detection. However, it remains susceptible to non-ideal conditions, such as the presence of noise and carrier offset, where the receiver obtain two different sampling instances with a non-zero phase shift when detecting an ORS. To address this issue, we propose encoding a single ACK message into M identical ORSs and collectively defining these ORSs as an ACK signal. Additionally, we set detection thresholds. Specifically, let $\Delta\varphi_i$ denote the phase shift between the two sampling instances of the ORS i. The receiver detects ORS i if $|\Delta\varphi_i|$ is smaller than a threshold $\Delta\hat{\varphi}$ ($\Delta\hat{\varphi} > 0$) and detects an ACK signal if the number of detected ORS, N_{ORS}, is larger than another threshold, $\hat{N}_{ORS}$ ($\hat{N}_{ORS} < M$).

Logically, since each ORS consists of two same waves, the ACK signal should consist of 2M identical waves, as shown in Fig. 4(a). However, this would introduce additional transmission overhead. To maintain reliability while reducing

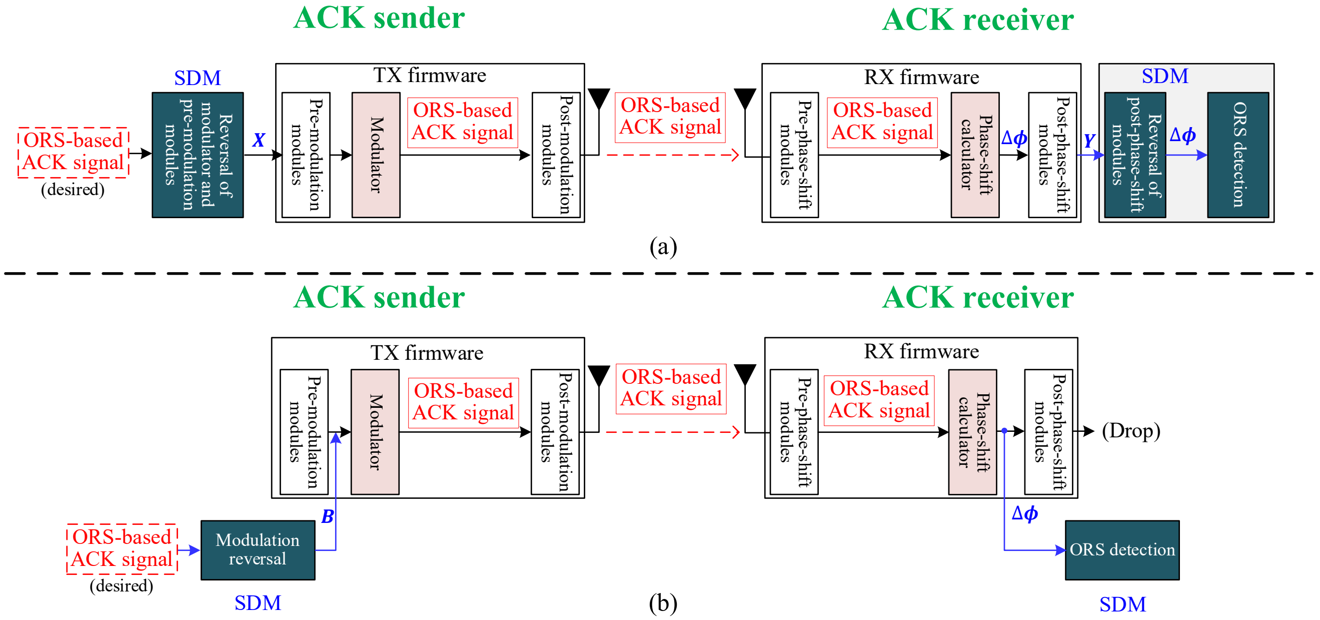


Fig. 5. GOP-ACK can be implemented via (a) no-firmware-modifying and (b) firmware-modifying methods.

overhead, we let ORS i's second wave serves as ORS i+1's first wave for $i = 1, \ldots, M$-1, as shown in Fig. 4(b). As a result, the ACK signal only comprises M+1 identical waves. Experimental results in Section VI demonstrate that ACK signals in Fig. 4(a) and Fig. 4(b) exhibit almost the same detection performance.

3) Methods of reusing existing modules

As a CTC design, GOP-ACK reuses existing communication modules to convey ACKs. Particularly, it mandates the reuse of a modulator to generate desired ORS-based ACK signals on the sender side and the reuse of a phase-shift calculator on the receiver side to detect these signals, as shown in Fig. 2.

In practice, the reused modules are usually embedded in the physical-layer firmware. For a sender-receiver pair, we refer to the firmware installed on the sender side as TX firmware and the firmware on the receiver side as RX firmware. GOP-ACK only requires that the TX firmware contains a modulator, and the RX firmware contains a phase-shift calculator. Without loss of generality, as shown in Fig. 5, we assume that in TX firmware, each input message passes successively through pre-modulation modules, a modulator, and post-modulation modules, resulting in its transformation into a signal for transmission. On the other hand, in RX firmware, each received signal passes successively through pre-phase-shift modules, a phase-shift calculator, and post-phase-shift modules before being restored to a message.

As a general approach, GOP-ACK provides two methods for reusing firmware modules to achieve ORS-based ACK for different CTC schemes: 1) no-firmware-modifying method, which accesses the input and output interfaces of the firmware only, designed for the firmaware-nonmodifying CTC scheme, and 2) firmware-modifying method, which can access the internal interface of the firmware, intended for the firmware-modifying CTC scheme.

No-firmware-modifying method. This method reuses all modules in firmware, as shown in Fig. 5(a). On the ACK sender side, we input a special message $\boldsymbol{X}$ to TX firmware to reuse all its modules for transmitting ORS-based ACK signals. The special message can be constructed by reversing the operations of the modulator and the pre-modulation modules in TX firmware. On the ACK receiver side, we first reuse all modules in RX firmware to obtain a message $\boldsymbol{Y}$. Then, we may create one SDM to process $\boldsymbol{Y}$ for obtaining the desired ACK signals. Roughly speaking, SDM may contain two submodules: one for transforming back $\boldsymbol{Y}$ to a phase shift $\Delta\boldsymbol{\phi}$ by reversing the operations of the post-phase-shift modules in RX firmware, and another for detecting the received ACK signals by $\Delta\boldsymbol{\phi}$. Further details are provided in Appendix D in the Supplementary File of this paper [34]. We note that the no-firmware-modifying method is cumbersome and time-consuming, as it reverses the operations of many modules. With the growing demand for flexible CTC services and with the proliferation of software-defined firmware, accessing the internal interfaces to reuse only the necessary modules for CTC, which makes CTC implementation redundancy-free and timesaving, has received much interest [18], [24].

Firmware-modifying method. This method reuses only the necessary modules in firmware. As shown in Fig. 5(b), if we can access the modulator in TX firmware via an internal interface, we may create an SDM that reverses the operation of the modulator in TX firmware. SDM transforms the desired ACK signal into a bit sequence, $\boldsymbol{B}$. Then $\boldsymbol{B}$ is sequentially passed to the modulator via the internal interface for restoring the desired ACK signal, which is next fed to the post-modulation modules in TX firmware for transmission. As a result, we only reuse the modulator and post-modulation modules in TX firmware. On the ACK receiver side, as shown in Fig. 5(b), if we can obtain the output of the phase-shift calculator in RX firmware, $\Delta\boldsymbol{\phi}$, via an internal interface, we may create SDM to detect the received ACK signal by $\Delta\boldsymbol{\phi}$. As a result, we only reuse the pre-phase-shift modules and phase-shift calculator in RX firmware. It is worth noting that with the emergence of more modifiable firmware [15]–[25], obtaining $\Delta\boldsymbol{\phi}$ from RX firmware via an internal interface is feasible, as

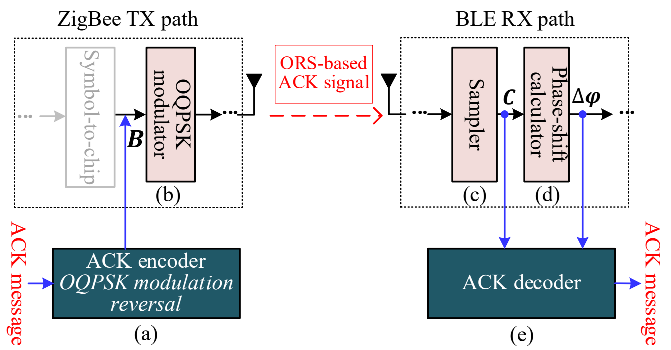


Fig. 6. In our ZigBee-to-BLE feedback, an ORS-supporting ZigBee TX chain shown in (a)-(b) and an ORS-supporting BLE RX chain shown in (c)-(e).

verified by many existing CTC designs via real-world experiments [17]–[20], [32]. Our real-world experiments in Section VI.A also verify this feasibility.

To contribute to the advancement of CTC, this paper primarily focuses on the more promising firmware-modifying method in reusing firmware modules.

The above framework provides clear and easy-to-follow guidelines for implementing GOP-ACK in specific CTC cases. Next, we examine the feasibility of the guidelines by applying them to design the ZigBee-to-BLE feedback in the next section and to design ZigBee-to-WiFi feedback in Appendix B of the Supplementary File of this paper [34].

IV. Case study: ZigBee-to-BLE feedback

In this section, we follow the guidelines of our framework presented in Section III.B to design ZigBee-to-BLE feedback, in which a ZigBee receiver transmits an ACK message (e.g., successful reception or reception error) to a BLE sender. In our design, we aim to answer the following critical questions:

Q1': How to design ORS for ZigBee-to-BLE feedback?

Q2': How to design reliable and lightweight ORS-based ACK for ZigBee-to-BLE fee dback?

Q3': How to reuse ZigBee's modulator and BLE's phase-shift calculator for transceiving ORS-based ACK signals?

We first follow the guidelines from Section III.B to design an ORS-supporting ZigBee-to-BLE chain, answering *Q3'*. Then, we follow the guidelines from Section III.B.1) and fully consider ZigBee modulation and BLE demodulation to design ORS, which answers *Q1'*. Finally, we follow the guidelines from Section III.B.2) to design the lightweight and reliable ORS-based ACK, thus answering *Q2'*.

A. ORS-supporting ZigBee-to-BLE chain

As shown in Fig. 6, our ORS-supporting ZigBee-to-BLE chain can be divided into an ORS-supporting ZigBee TX chain and an ORS-supporting BLE RX chain. The former reuses ZigBee's modulator by integrating a conventional ZigBee TX chain with an SDM (called ACK encoder). The latter reuses BLE's phase-shift calculator by integrating a conventional BLE RX chain with an SDM (called ACK decoder).

1) ORS-supporting ZigBee TX chain

Our ORS-supporting ZigBee TX chain transmits an ACK message by following the three steps below, as shown in Figs. 6(a)-(b).

1. The ACK encoder transforms an ACK message (from the upper layer) into a carefully constructed binary chip sequence $\boldsymbol{B}$. This message-to-sequence transformation is crucial to the ORS construction, to be detailed later.
2. ZigBee's offset quadrature phase-shift keying (OQPSK) modulator modulates the chip sequence $\boldsymbol{B}$ into a set of ORSs (instead of regular ZigBee signals), because of our special construction of $\boldsymbol{B}$, to be detailed later.
3. The rest of the ZigBee TX chain amplifies and sends out the ORSs.

2) ORS-supporting BLE RX chain

Upon receiving the ORSs, our ORS-supporting BLE RX chain transforms them to an ACK message via the following three steps, as shown in Figs. 6(c)-(e).

1. The BLE's sampler samples the received ORS signal to obtain a set of complex sampling instances $\boldsymbol{C}$.
2. The BLE's phase-shift calculator calculates the phase shifts $\Delta\boldsymbol{\varphi}$ between two consecutive sampling instances in $\boldsymbol{C}$.
3. The ACK decoder obtains the ACK message according to $\boldsymbol{C}$ and $\Delta\boldsymbol{\varphi}$, to be detailed later.

B. ORS design

We first present ZigBee modulation and BLE demodulation, and then design ORSs for ZigBee-to-BLE feedback.

1) ZigBee modulation and BLE demodulation

We next explain ZigBee modulation and then BLE demodulation.

ZigBee modulation. ZigBee adopts the OQPSK technique [9], [49]–[51] to transform a chip sequence to a signal in the following three steps, as shown in Figs. 7(a)-(d).

1. The modulator transforms the odd-indexed chips of the chip sequence into an in-phase signal $I(t)$ consisting of half-sine pulses (called I-pulses). In the chip-to-pulse transformation, chips "1" and "0" are transformed to a positive half-sine pulse of $1\,\mu s$ and a negative one, respectively.
2. The modulator transforms the even-indexed chips into a quadrature signal $Q(t)$, which also consists of half-sine pulses (called Q-pulses), by adopting the above same chip-to-pulse transformation.
3. The modulator delays $Q(t)$ for $0.5\mu s$ and superposes it with $I(t)$, and finally generates a ZigBee signal.

BLE demodulation. A phase-shift calculator receives a set of sampling instances with a sampling interval of $1\mu s$ as shown in Fig. 7(d), and then calculates the phase shifts of every two consecutive sampling instances. Finally, these phase shifts are decoded into bits.

2) ORS construction

Recall that an ORS should meet (1), i.e., it should consist of two identical waves and be of zero phase shift, as shown in Fig. 2. To achieve a ZigBee-to-BLE direct feedback, we aim to exploit ZigBee's OQPSK modulation technique to construct an ORS that can be detected by a BLE demodulator.

ORS construction requirements for BLE detection: We note that a BLE demodulator samples the received signals every $1\mu s$ and then calculates the phase shift of two consecutive sampling instances for decoding the received signals. Thus, to meet (1),

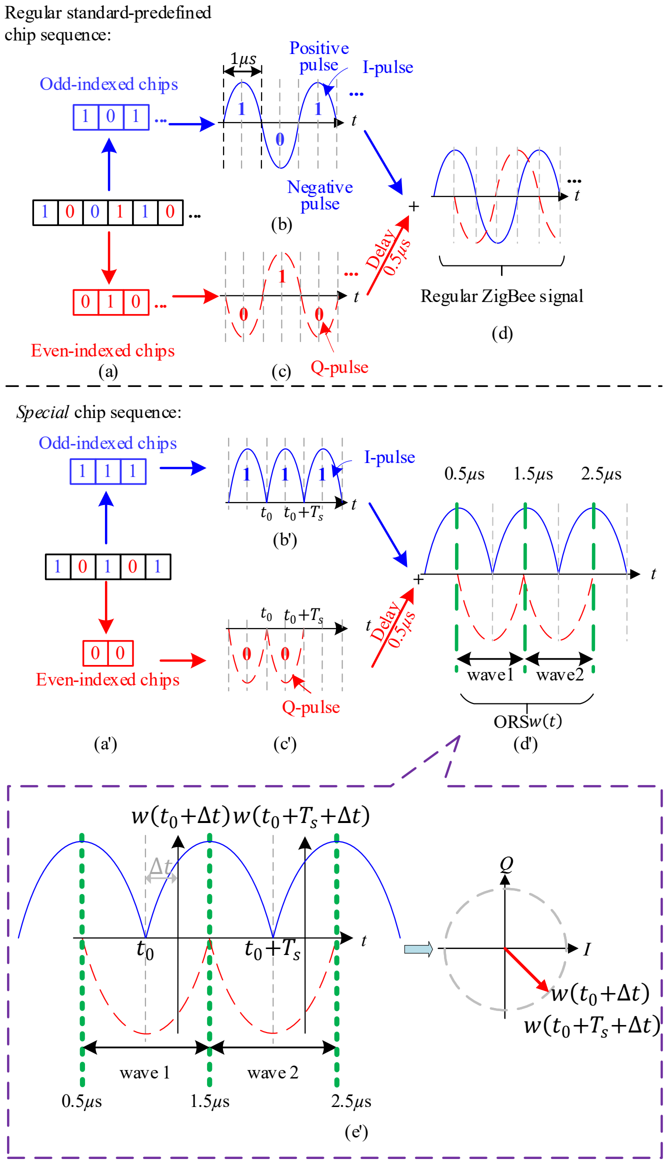


Fig. 7. In ZigBee, an OQPSK modulator modulates a regular chip sequence into a regular ZigBee signal as shown in (a)-(d), whereas it modulates a special chip sequence into an ORS as shown in (a')-(e').

our ORS construction should meet the following two requirements.

R1: each wave of ORS lasts for $1\mu s$.

R2: the two waves of ORS are identical.

ORS construction by ZigBee modulation: As shown in Figs. 7(a)-(d), a ZigBee modulator first creates an in-phase signal consisting of I-pulses and then creates a quadrature signal consisting of Q-pulses, next delays the quadrature signal for $0.5\mu s$ and superposes the delayed quadrature signal with the in-phase signal together, and finally transmits the superposed signal. Here each I- or Q-pulse lasts for $1\mu s$ and is either positive or negative. Keeping this in mind, we explain the ORS construction.

R1 dictates that a wave of an ORS is a superposition of two half I-pulses and a complete Q-pulses (e.g., wave 1 in Fig. 7(d')), according to the modulation process of a ZigBee signal.

R2 dictates that an ORS can be obtained by superposing three identical I-pulses and two identical Q-pulses into a ZigBee signal, and then trimming its first-$0.5\mu s$ and last-$0.5\mu s$ parts. This implies that we can feed a carefully constructed chip sequence to the ZigBee modulator for obtaining an ORS. For example, assume that the chip sequence is “10101”, as shown in Figs. 7(a')-(d'). After receiving the sequence, the ZigBee modulator first transforms the odd-indexed chips “111” into three identical I-pulses and the two even-indexed chips “00” to two identical Q-pulses, and then superpose them together to create a ZigBee signal of $3\mu s$, and finally generate an ORS, i.e., the portion of the signal from $0.5\mu s$ to $2.5\mu s$.

We note that an ORS is always a superposition of identical I-pulses and identical Q-pulses, where each pulse is either positive or negative. In total, we have 4 types of ORS, as shown in TABLE I. Further, each type of ORS is associated with one unique quadrant. For example, for the given sequence 10101, as shown in Figs. 7(d')-(e'), the ORS consists of positive I-pulses and negative Q-pulses, and hence its sampling instance is in quadrant 4 because the sampling value of its I-pulse is positive and that of its Q-pulse is negative. The illustration of ORS quadrants can be found in Appendix C in the Supplementary File of this paper [34].

TABLE I. MAPPING AMONG BIT SEQUENCES, QUADRANTS, ORS TYPES, AND ACK TYPES

Special bit sequence	Odd-indexed bits	I-pulse type	Quadrant of ORS sampling instances	ORS type	ACK type
	Even-indexed bits	Q-pulse type			
11111	111	positive	1	1	1
	11	positive			
01010	000	negative	2	2	2
	11	positive			
00000	000	negative	3	3	3
	00	negative			
10101	111	positive	4	4	4
	00	negative			

C. ORS-based ACK encoder and decoder

In our design, we define 4 types of ACK messages for ZigBee-to-BLE feedback: 1) successful reception, 2) reception error, 3) successful reception of retransmission, and 4) reception error of retransmission.

In our ZigBee-to-BLE feedback as shown in Fig. 7, on the ZigBee side, one type of ACK message is encoded into one unique ACK chip sequence via our newly introduced ACK encoder. The chip sequence is then transformed into one type of ORS-based ACK signal via the ZigBee modulator for transmission. On the BLE side, BLE samples the received ACK signal and then decodes the sample values into an ACK message via our newly introduced ACK decoder. To ensure a low-cost and robust ACK detection even under non-ideal channel conditions, we define one ACK signal to a sequence of overlapped ORSs.

In our ZigBee-to-BLE design, it is the ORS-based ACK signal that dictates an ACK chip sequence and ACK message.

We next present the ACK signal, encoder, and decoder sequentially.

1) ACK signal

Here, we define the ACK signal, ACK chip sequence, and the mapping between them.

ACK signal: We follow the guidance from Section III.B.2) to design the ACK signal. More specifically, same as Fig. 4(b), an ACK signal consists of M ORS, where ORS i's second wave serves as ORS i+1's first wave for $i = 1, \dots, M$-1, as shown in Fig. 8(a).

ACK chip sequence: As shown in Figs. 8(a)-(b), in an ACK signal, the M+1 waves can be constructed by superposing M+2 identical I-pulses and M+1 identical Q-pulses. Each pulse can be generated by one chip as shown in Figs. 7(a')-(c'). For an ACK signal, we define an ACK chip sequence as a $2M$+3-chip sequence $\boldsymbol{B} = [b_I, b_Q, \dots, b_I, b_Q, b_I]_{2M+3}$, where each odd-indexed chip b_I and even-indexed chip b_Q respectively correspond to an I-pulse and a Q-pulse of the ACK signal as shown in Figs. 8(b)-(c). Because b_I and b_Q are either 1 or 0, we have four types of chip sequences in total, each corresponding to one type of ACK signals. For example, the chip sequence of $b_I = 1$ and $b_Q = 1$ represents the type-1 ACK signal.

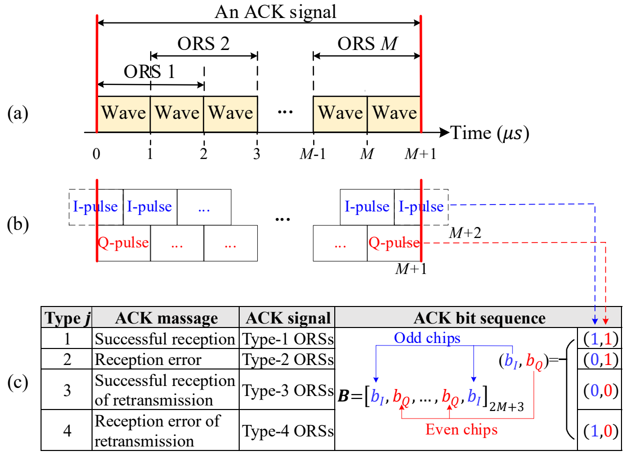
Mapping from ACK messages to ACK signals and chip sequences: We may map one ACK message to one ACK signal and one ACK chip sequence uniquely as shown in Fig. 8(c). For example, the ACK message "successful reception" is mapped to the type-1 ACK signal and the chip sequence of $b_I = 1$ and $b_Q = 1$.

2) ACK encoder

An ACK encoder transforms a given ACK message to the corresponding chip sequence, following the relationship listed in Fig. 8(c). For example, the ACK encoder transforms the message "successful reception" to the chip sequence of $b_I = 1$ and $b_Q = 1$.

3) ACK decoder

When receiving an ACK signal over the channel, an ACK receiver (i.e., BLE in this case) keeps sampling the signal, as shown in Figs. 6(c)-(e). For every $M + 1$ complex sampling instances $\{C_m\}_{m=1}^{M+1}$, it calculates their M phase shifts $\{\Delta\varphi_m\}_{m=1}^{M}$, where $\Delta\varphi_m$, $m = 1, \dots, M$, is the phase shift between C_m and C_{m+1}. Then, it feeds $\{C_m\}_{m=1}^{M+1}$ and $\{\Delta\varphi_m\}_{m=1}^{M}$ into an ACK decoder, which transforms them into an ACK message via the following two steps.



Type j	ACK massage	ACK signal	ACK bit sequence
1	Successful reception	Type-1 ORSs	(1,1)
2	Reception error	Type-2 ORSs	(0,1)
3	Successful reception of retransmission	Type-3 ORSs	(0,0)
4	Reception error of retransmission	Type-4 ORSs	(1,0)

Fig. 8. (a) An ACK signal constructed by M overlapping ORSs (M+1 waves). (b) M+1 identical waves can be constructed by superposing M+2 identical I-pulses and M+1 identical Q-pulses. (c) The one-to-one mapping among ACK messages, signals, and bit sequences.

1. ACK-signal detection. A decoder detects an ACK signal successfully if the following two conditions are met:

- *C1: Condition of a successful busy channel detection.* Let $\lambda = \frac{\sum_{m=1}^{M+1}|C_m|^2}{M+1}$ denote the power of $\{C_m\}_{m=1}^{M+1}$. C1 is satisfied if λ is greater than a threshold $\hat{\lambda}$, namely,
$$\lambda > \hat{\lambda}. \tag{4}$$
- *C2: Condition of the number of detected ORSs.* Recall that an ACK signal consists of M ORSs. In non-ideal conditions, a decoder cannot always detect all these ORSs successfully. Let N_{ORS} denote the number of detected ORSs. C2 is satisfied if N_{ORS} is greater than a threshold $\widehat{N}_{ORS}$, namely,
$$N_{ORS} > \widehat{N}_{ORS}. \tag{5}$$
Note that an ORS consists of two consecutive and identical waves. In ideal conditions, a decoder detects an ORS if the phase shift between the two wave sampling instances of the ORS, $\Delta\varphi$, is 0. In the presence of noise and carrier offset, however, $\Delta\varphi$ is not always 0. Hence, in our design, a decoder detects an ORS m if $|\Delta\varphi_m|$ is smaller than a threshold $\Delta\hat{\varphi}$ ($\Delta\hat{\varphi} > 0$), namely,
$$|\Delta\varphi_m| < \Delta\hat{\varphi}. \tag{6}$$

2. ACK-type detection. Recall that a type-j ACK signal consists of M type-j ORSs. When receiving an ORS, the decoder infers that the ORS type is j if the first sampling instance of the ORS appears in quadrant j of the constellation diagram (as shown in Appendix C in the Supplementary File [34]). However, given a type-j ORS, in non-ideal conditions, the decoder might identify its type as another one mistakenly. Let N_{ORS}^j be the number of detected type-j ORSs, $j = 1, \dots, 4$. In our design, the decoder first calculates $j_0 = \underset{j}{\operatorname{argmax}}\{N_{ORS}^j\}_{j=1}^4$ and then infers that the ACK type is j_0. Finally, the decoder decodes the ACK signal into an ACK message j_0.

C3: Condition of a successful ACK-type detection. C3 is satisfied if an ACK sender has transmitted a type-j_0 ACK signal and the decoder at the ACK receiver infers that the ACK-type is j_0.

In short, a decoder decodes an ACK message successfully if conditions C1-C3 are satisfied. Algorithm 1 realizes the above ACK decoding procedure. Given the sampling instances $\{C_m\}_{m=1}^{M+1}$ and their phase shifts $\{\Delta\varphi_m\}_{m=1}^{M}$, the decoder calculates $\widetilde{C1} = [\![\lambda > \hat{\lambda}]\!]$ and $\widetilde{C2} = [\![N_{ORS} > \widehat{N}_{ORS}]\!]$ to check C1 and C2 (lines 3-4), where $[\![s]\!] = 1$ if the statement s is true and 0 otherwise. If C1 and C2 are satisfied (line 5), the decoder detects an ACK signal and then infers and returns the ACK message j_0 (lines 6-7); otherwise, the decoder returns null (lines 8-9).

Remarks: A typical BLE-to-ZigBee transmission with feedback follows the pattern: packet/SIFS/ACK. That is, BLE

```
Algorithm 1: ACK decoding
Input: $\{C_m\}_{m=1}^{M+1}$ and $\{\Delta\varphi_m\}_{m=1}^{M}$
Output: ACK message $j_0$
$\widetilde{\mathrm{C1}} = [\![\lambda > \hat{\lambda}]\!]$                  // Check C1
$\widetilde{\mathrm{C2}} = [\![N_{ORS} > \hat{N}_{ORS}]\!]$      // Check C2
If $\widetilde{\mathrm{C1}}$ and $\widetilde{\mathrm{C2}}$                  // Step 1: ACK-signal detection
    $j_0 \leftarrow \underset{j}{\mathrm{argmax}}\{N_{ORS}^{j}\}_{j=1}^{4}$ // Step 2: ACK-type detection
    Return $j_0$                        // Success when C3 is satisfied
Else
    Return Null                     // No ACK is detected
```

sends a packet to ZigBee, then waits for a short interframe space (SIFS) time, and finally receives an ACK from ZigBee. Nevertheless, BLE retransmits the packet if one of the following cases occurs: 1) the ACK indicates that the packet is not successfully received by ZigBee, 2) BLE's ACK decoder fails to decode the ACK, and 3) BLE does not receive an ACK within a certain time (i.e., ACKTimeout time, which can be set to a SIFS time plus an ACK transmission time).

V. Performance model

In this section, we model the performance of GOP-ACK. In GOP-ACK, if node A transmits a packet to node B, it will receive an ACK message from node B. We model two performance metrics: one successful ACK decoding probability (i.e., the probability that A correctly decodes the ACK message from B) and one complete transmission time (i.e., the time between when A sends a packet to B and when it receives an ACK that acknowledges B's successful reception). An ACK message may be delivered via a long or short ACK signal, as shown in Figs. 8(a') and (a). Thus, we need to investigate the impact of the two types of ACK signals on the two performance metrics of our model. For summarizations of the used notations, please refer to Appendix C in the Supplementary File of the paper [34].

A. Successful ACK decoding probability

In GOP-ACK, upon receiving a packet from node A, node B sends back an ACK signal $x(t)$. For the sake of clarity, in the rest of this subsection, we call B and A an ACK encoder and decoder, respectively.

Without loss of generality, we assume that $x(t)$ is a superposition of an in-phase signal $x_I(t)$ and a quadrant signal $x_Q(t)$. As described in Section IIIV.C.1), $x(t)$ is a sequence of M' waves where

$$M' = \begin{cases} 2M & \text{a long ACK signal} \\ M+1 & \text{a short ACK signal.} \end{cases} \tag{7}$$

To detect an ACK signal, the decoder keeps sampling $x(t)$ with a sampling offset ΔT, which is assumed to follow a uniform distribution, namely, $\Delta T \sim \mathcal{U}\left[-\frac{\tau}{2}, \frac{\tau}{2}\right]$, where τ is the sampling interval. Let C_m^I and C_m^Q denote the in-phase and quadrature components of the m-th sampling instance C_m, respectively. We have

$$C_m = C_m^I + jC_m^Q, \tag{8}$$

where

$$C_m^I \sim \mathcal{N}\left(x_I(m\tau + \Delta t), \frac{\sigma^2}{2}\right),$$

$$C_m^Q \sim \mathcal{N}\left(x_Q(m\tau + \Delta t), \frac{\sigma^2}{2}\right).$$

In (8), we assume an additive white Gaussian noise (AWGN) channel. σ^2 is the noise power. Δt is the value of ΔT.

According to the ACK decoding procedure in Algorithm 1, an ACK is successfully decoded if C1 (i.e., condition of a successful busy channel detection), C2 (i.e., condition that the number of detected ORSs is larger than a threshold), and C3 (i.e., condition of a successful ACK type detection) are satisfied. Given a sampling offset $\Delta T = \Delta t$, let $P_{C1}(\Delta t)$, $P_{C2}(\Delta t)$, and $P_{C3}(\Delta t)$ be the probability of satisfying C1-C3, respectively. Then, the probability of one successful ACK decoding is $P_{C1}(\Delta t) \cdot P_{C2}(\Delta t) \cdot P_{C3}(\Delta t)$. Since ΔT follows a uniform distribution, the average probability of one successful ACK decoding is given as

$$P_{ACK}^{s} = \frac{1}{\tau}\int_{-\frac{\tau}{2}}^{\frac{\tau}{2}} P_{C1}(\Delta t) \cdot P_{C2}(\Delta t) \cdot P_{C3}(\Delta t) d\Delta t. \tag{9}$$

1) Calculation of $P_{C1}(\Delta t)$

In GOP-ACK, when receiving a long or short ACK signal consisting of M' waves, the decoder samples its each wave once. Recall that C1 is met when the average power of these M' sampling instances, λ, is higher than a threshold, $\hat{\lambda}$. According to (8) and [51], λ is a non-central chi-squared random variable with $2M'$ degrees of freedom and a non-centrality parameter $\sum_{m=1}^{M'}\left[x_I^2(m\tau+\Delta t)+x_Q^2(m\tau+\Delta t)\right]$. Thus, we have:

$$\begin{aligned} P_{C1}(\Delta t) &= P\left(\lambda > \hat{\lambda}\right) \\ &= Q_{M'}\left(\frac{\sqrt{\sum_{m=1}^{M'}\left(x_I^2(m\tau+\Delta t)+x_Q^2(m\tau+\Delta t)\right)}}{\sigma}, \frac{\sqrt{M'\hat{\lambda}}}{\sigma}\right), \end{aligned} \tag{10}$$

where $Q_{M'}(a,b) = \frac{1}{a^{M'-1}}\int_b^\infty \varepsilon^{M'} e^{-\frac{a^2+\varepsilon^2}{2}} I_{M'-1}(a\varepsilon) d\varepsilon$ is the generalized Marcum Q function of order M' and $I_{M'-1}$ is the modified Bessel function of the first kind. For the derivation of (10), please refer to Appendix E in the Supplementary File [34].

2) Calculation of $P_{C2}(\Delta t)$

In GOP-ACK, when receiving an ACK signal consisting of M ORS, the decoder performs M independent detections of ORSs. Under a sampling offset Δt, each detection is successful with a probability $P_o^s(\Delta t)$. Recall that C2 is satisfied if $N_{ORS} > \hat{N}_{ORS}$. Thus, the probability $P_{C2}(\Delta t)$ of detecting enough ORSs is given as

$$\begin{aligned} P_{C2}(\Delta t) &= P\left(\hat{N}_{ORS} < N_{ORS} \le M\right) \\ &= \sum_{N_{ORS}=\hat{N}_{ORS}+1}^{M}\left[\binom{M}{N_{ORS}}\left(P_o^s(\Delta t)\right)^{N_{ORS}} \right. \\ &\qquad \left. \times \left(1-P_o^s(\Delta t)\right)^{M-N_{ORS}}\right]. \end{aligned} \tag{11}$$

Note that the decoder decodes an ORS if the phase shift $\Delta\varphi$ of the ORS's sampling instances is smaller than a threshold $\Delta\hat{\varphi}$. According to [52], $P_o^s(\Delta t)$ is calculated as:

$$\begin{aligned} P_o^s(\Delta t) &= P(\Delta\varphi < \Delta\hat{\varphi}) \\ &= 1 - \frac{\sin\Delta\hat{\varphi}}{2\pi}\int_{-\pi/2}^{\pi/2}\frac{e^{-\gamma(1-\cos\Delta\hat{\varphi}\cos\rho)}}{1-\cos\Delta\hat{\varphi}\cos\rho}d\rho, \end{aligned} \tag{12}$$

in which $\gamma = \lambda/\sigma^2$ is the signal-to-noise ratio (SNR).

3) Calculation of $P_{C3}(\Delta t)$

Recall that N_{ORS}^{j} is the number of detected type-j ORSs. Let $P_{\Delta t}(n_1, \dots, n_{\mathcal{T}}) \triangleq P_{\Delta t}(N_{ORS}^{1}=n_1, \dots, N_{ORS}^{\mathcal{T}}=n_{\mathcal{T}} \text{ under } \Delta t)$

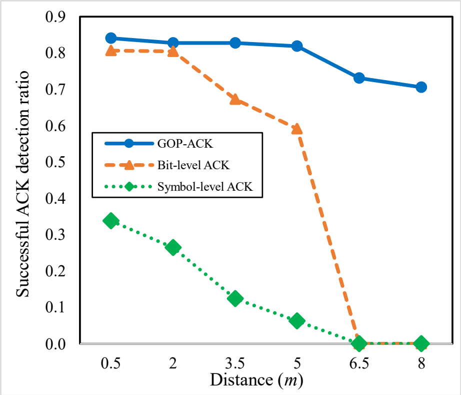


Fig. 9. Successful detection ratio of GOP-ACK, symbol-level ACK, and bit-level ACK for ZigBee-to-WiFi feedback under different distances between the ZigBee and WiFi nodes.

denote the joint probability mass function of $N_{ORS}^1, \dots, N_{ORS}^{\mathcal{T}}$ under Δt. According to C3, when the encoder sends a type-j^*ACK signal to the decoder, the decoder successfully detects type j^* if $j^* = j_0 \triangleq \underset{j}{\text{argmax}}\{n_j\}_{j=1}^{\mathcal{T}}$, where $j^* = 1, \dots, \mathcal{T}$. Then, the successful probability of one ACK-type detection, $P_{C3}(\Delta t)$, is calculated as:

$$P_{C3}(\Delta t) = \sum_{j^*=1}^{\mathcal{T}} \sum_{j=1}^{\mathcal{T}} \text{Prob}(j_0 = j^* | j^*) P(j^*)$$
$$= \sum_{j^*=1}^{\mathcal{T}} \sum_{j=1}^{\mathcal{T}} \text{Prob}\left(\{n_{j^*} > n_j\}_{j \neq j^*}\right) P(j^*) \qquad (13)$$
$$= \sum_{j^*=1}^{\mathcal{T}} \sum_{j=1}^{\mathcal{T}} \sum_{n_{j^*} > n_{j \neq j^*}} P_{\Delta t}(n_1, \dots, n_{\mathcal{T}})\, P(j^*)\,.$$

For the expression of $P_{\Delta t}(n_1, \dots, n_{\mathcal{T}})$, please refer to Appendix E in the Supplementary File [34].

B. Complete transmission time

As explained in Section IV.C.3), after node A sends a packet to node B, A should receive an ACK from B after a SIFS time. In case that A does not receive an ACK within an ACKTimeout time (which is set to a SIFS time plus an ACK transmission time), it retransmits the packet. We define one complete transmission time Ω as the time between when A sends a packet to B and when A receives an ACK that acknowledges B's successful reception.

In non-ideal conditions, A might successfully deliver a packet to B after multiple transmission rounds, each with the same duration T_r. Let N_r denote the number of transmission rounds for delivering a packet. We have $\Omega = N_r T_r$. The expected value of Ω is given as

$$E(\Omega) = E(N_r) T_r, \qquad (14)$$

where $E(N_r)$ is the mean of N_r. Below, we calculate $E(N_r)$ and T_r, respectively.

Calculation of $E(N_r)$. Let P_{TX}^s be the successful packet transmission probability in one transmission round. N_r follows a geometric distribution with parameter P_{TX}^s. Then, we can calculate $E(N_r)$ as:

$$E(N_r) = \frac{1}{P_{\text{TX}}^s}. \qquad (15)$$

For each transmission round, we say that one A-to-B packet transmission is successful if B successfully receives the packet from A, whereas A successfully decodes the ACK that acknowledges B's successful reception. Let P_{RX}^s be the successful packet reception probability at B. Note that the successful ACK decoding probability at A, P_{ACK}^s, is given in (9). We can calculate P_{TX}^s as:

$$P_{\text{TX}}^s = P_{\text{RX}}^s \cdot P_{\text{ACK}}^s. \qquad (16)$$

Calculation of T_r. Recall that each transmission round follows the pattern: packet/SIFS/ACK. Let T_{pkt}, T_{SIFS}, and T_{ACK}, be the durations of a packet transmission, SIFS, and ACK, respectively. Then, we have:

$$T_r = T_{\text{pkt}} + T_{\text{SIFS}} + T_{\text{ACK}}. \qquad (17)$$

Here, $T_{\text{ACK}} = M' T_w$ since an ACK signal comprises M' waves, each lasting for a duration of T_w, as described in (7).

VI. Evaluations

We first verify that GOP-ACK outperforms the state of the art via real-world experiments. Then, we verify the accuracy of our theoretical model via simulations.

A. Comparisons with the state of the art

We conduct real-world experiments to validate the feasibility of GOP-ACK and compare GOP-ACK with the existing symbol-level ACK [19], [26] and bit-level ACK [18] designs. In the comparison, we focus on ZigBee-to-WiFi feedback because it is the focus of the existing designs. For GOP-ACK, we adopt our design presented the Supplementary File of this paper [34]. We use the successful ACK detection ratio as the comparison metric, which is defined as the ratio between the number of successfully detected ACKs and the number of total transmitted ACKs.

In our experiment, we adopt two SDRs Y590s [53] with GNU Radio 3.8 [54] to implement a ZigBee node using ZigBee channel 18 and a WiFi node using WiFi channel 6, which inherently have a 3 MHz carrier offset between them. For each run, the ZigBee node sends 10000 ACK messages to the WiFi node, with a transmission power of -1.4dBm and the transmission interval of 0.02s. In each GOP-ACK transmission, we set $\Delta\hat{\varphi} = 0.8$, $\widehat{N}_{ORS} = 10$, and $M = 15$. In each symbol-level ACK transmission, the WiFi receiver obtains 84 phase shifts. If at least 42 phase shifts are negative (positive), the WiFi receiver infers that an ACK message "0" ("1") is received. Like the symbol-level ACK, in each bit-level ACK transmission, the WiFi receiver obtains 640 phase shifts. If at least 210 of them are negative (positive), the WiFi receiver infers that an ACK message "0" ("1") is received.

Fig. 9 plots the successful detection ratios of GOP-ACK, symbol-level ACK, and bit-level ACK n a ZigBee-to-WiFi feedback scenario, with the distance between the ZigBee and WiFi nodes ranging from 0.5m to 8m. From it, we have two observations as follows.

- All designs exhibit a decrease in ratios as the distance increases. This trend can be attributed to the reduction in signal-to-noise ratio (SNR), an important factor regarding the detection performance, with increasing distance.
- Regardless of the distance, GOP-ACK consistently achieves higher ratios than both bit-level ACK and symbol-level ACK. This indicates that our GOP-ACK approach, by encoding ACK messages into ORS sets, effectively mitigates the adverse effects of sampling offsets and non-ideal channel conditions on ACK detection, thereby

offering superior reliability compared to other ACK designs.

- The ratio of bit-level ACK is higher than that of symbol-level when the distance is less than 6.5 meters. This demonstrates that bit-level ACK enhances detection reliability to a certain degree by extending the sinewave originally proposed by symbol-level ACK (refer to Section II.B).

B. Model verification

We conduct extensive simulations to verify our theoretical model. In our simulations, we focus on the following case: BLE first transmits a packet to ZigBee and then ZigBee adopts the proposed ZigBee-to-BLE feedback to transmit an ACK to BLE. We developed our simulator based on MATLAB 2020b [55]. It consists of ZigBee, BLE, and AWGN channel modules. They are implemented using the OQPSK modulator module [56], the Gaussian minimum shift keying modulator module [57], and the white Gaussian noise generator module [58] respectively.

In all figures shown in this subsection, each simulation value is the average one over 5,000 runs, where each run is for one packet transmission. Labels "Short ACK *" and "Long ACK *" denote the theoretical results about short signals and long signals, respectively; labels "Short ACK" and "Long ACK" denote the simulation results about short ACK signals and long ACK signals, respectively. Below, we first verify the successful ACK decoding probability P^s_{ACK} at BLE and then verify the mean complete transmission time $E(\Omega)$.

1) Successful ACK decoding probability

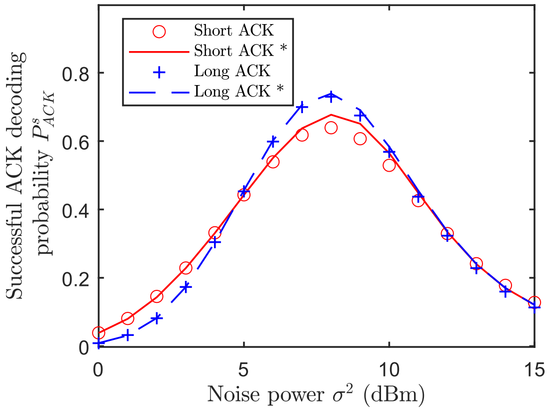


Fig. 11. P^s_{ACK} vs. different σ^2.

To verify the accuracy of P^s_{ACK} in (9), we assume that an ACK signal is comprised of M=15 ORS, set the threshold in condition C1, $\hat{\lambda}$=0.013mW, set the thresholds in C2, $\Delta\hat{\varphi}$=1, and $\hat{N}_{ORS}$=8, and set the ACK transmission power to 10 dBm.

Fig. 11 plots P^s_{ACK} vs. σ^2, when the noise power $\sigma^2 = 0,\ldots,15$ dBm and sampling offset $\Delta t = -0.5,\ldots,0.5\mu s$. From it, we conclude that:

- P^s_{ACK} is low when σ^2 is low. It is caused by frequent miss-detection, as explained in Fig. IV in the Supplementary File of this paper [34].
- P^s_{ACK} is not high when σ^2 is high. It is due to a low successful probability of ACK-type detection, as explained in Fig. VI in the Supplementary File of this paper [34].
- P^s_{ACK} for short signals is higher (lower) than that of long ACK signals when σ^2 is low (medium); they are almost the same when σ^2 is high. Therefore, it prefers to adopt short ACK signals.

In addition, we also verify $P_{C1}(\Delta t)$, $P_{C2}(\Delta t)$, and $P_{C3}(\Delta t)$ in (9) under different Δt and σ^2. For details, please refer to Appendix F in the Supplementary File of this paper [34].

2) Mean complete transmission time

We now verify the accuracy of the mean complete transmission time $E(\Omega)$ in (14). Consider the case: BLE first transmits a packet to ZigBee via a CTC design (e.g., BlueBee [11]), and then ZigBee adopts the proposed ZigBee-to-BLE feedback to transmit an ACK to BLE. According to ZigBee standard [59], the SIFS time is $T_{\text{SIFS}} = 192\mu s$ and each ZigBee packet consists of a preamble with length $T_{\text{pmb}} = 128\mu s$, a header with leagth $T_{\text{hrd}} = 64\mu s$, and a payload of L bytes, where each byte is carried by 2 symbols and each symbol duration T_{sym} is $16\mu s$. Thus, a packet duration is $T_{\text{pkt}} = T_{\text{pmb}} + T_{\text{hrd}} + 2LT_{\text{sym}}$. In addition, we assume that ZigBee receives a packet from BLE successfully with probability $P^s_{\text{pkt}} = 0.8$. Other parameter settings can be found in Section VI.B.1).

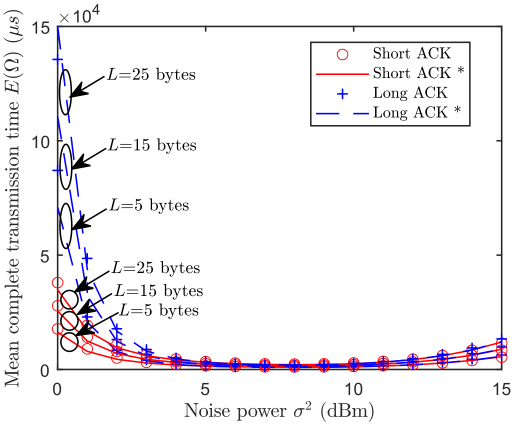


Fig. 10. $E(\Omega)$ vs. different σ^2.

Fig. 10 plots $E(\Omega)$ vs. σ^2, when $\sigma^2 = 0, \ldots 15$dBm and the payload size L is 5, 15, and 25 btyes. From this figure, we have the following observations:

- Given L, $E(\Omega)$ decreases as σ^2 increases from 0 to 8 dBm but increases as σ^2 continues increasing from 8 to 15 dBm. We explain the reason as follows. $E(\Omega)$ is decreasing with the successful ACK decoding probability, P^s_{ACK}, according to (14)-(16), while P^s_{ACK} increases as σ^2 increases from 0 to 8 dBm but decreases as σ^2 continues increasing from 8 to 15 dBm, as shown in Fig. 11.
- Given L, $E(\Omega)$ with short signals is almost always lower than that with long signals. Therefore, we prefer to adopt short ACK signals.
- Given σ^2 and signal type, $E(\Omega)$ increases with L, because a large L increases the packet transmission time.

Lastly, in Figs. 11-10, our theoretical results closely match the corresponding simulation results, indicating that our performance model is very accurate.

VII. Conclusion

Cross-Technology Communication (CTC) enables direct communications among devices with heterogeneous wireless technologies but is essentially unreliable. An efficient way to enhance reliability is to make ACK feedback from the receiver to the sender. However, exiting related studies are very limited. This study presents a general and offset-resistant physical-layer ACK approach to support reliable feedback for CTC cases. Its idea is to encode ACK messages with offset-resistant signals that 1) can be adapted into a wide range of CTC scenarios with minimal adjustment and 2) can be effortlessly and robustly detected even in the presence of sampling offsets. The study provides a framework for ease of applying GOP-ACK to specific CTC cases and presents two specific design cases: ZigBee-to-BLE and ZigBee-to-WiFi feedback. The study then proposes a theoretical model to analyze the performance of GOP-ACK and verifies the accuracy of the model via extensive simulations. It also validates that GOP-ACK outperforms the state of the art via real-world experiments. This study aims to enhance the practicality of CTC greatly.

Acknowledgment

The authors declare a potential conflict of interest with researchers who have expressed disagreement with the CTC implementation approach used in this study. This information has been disclosed to the journal.

References

[1] M. Ghahramani, M. Zhou, A. Molter, and F. Pilla, "IoT-based route recommendation for an intelligent waste management system," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 11883–11892, 2021.

[2] H. Han, W. Ma, M. Zhou, Q. Guo, and A. Abusorrah, "A novel semi-supervised learning approach to pedestrian reidentification," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 3042–3052, 2020.

[3] P. Zhang *et al.*, "A fault-tolerant model for performance optimization of a fog computing system," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1725–1736, 2021.

[4] W. Duo, M. Zhou, and A. Abusorrah, "A survey of cyber attacks on cyber physical systems: Recent advances and challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 5, pp. 784–800, 2022.

[5] C.-C. Tong, W.-S. Jwo, J.-Y. Lin, S.-F. Li, and J.-Y. Li, "The firmware design of analogue and digital filters," in *2011 Digital Signal Processing and Signal Processing Education Meeting (DSP/SPE)*, Jan. 2011, pp. 523–528. doi: 10.1109/DSP-SPE.2011.5739269.

[6] G. Comoretto *et al.*, "The Signal Processing Firmware for the Low Frequency Aperture Array," *J. Astron. Instrum.*, vol. 06, no. 01, p. 1641015, Mar. 2017, doi: 10.1142/S2251171716410154.

[7] K. Chebrolu and A. Dhekne, "Esense: communication through energy sensing," in *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09*, Beijing, China, 2009, p. 85. doi: 10.1145/1614320.1614330.

[8] X. Zhang and K. G. Shin, "Gap Sense: Lightweight coordination of heterogeneous wireless devices," in *2013 Proceedings IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 3094–3101. doi: 10.1109/INFCOM.2013.6567122.

[9] W. Jiang, S. M. Kim, Z. Li, and T. He, "Achieving receiver-side cross-technology communication with cross-decoding," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 639–652.

[10] Z. Li and T. He, "WEBee: Physical-Layer Cross-Technology Communication via Emulation," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, Snowbird, Utah, USA, Oct. 2017, pp. 2–14. doi: 10.1145/3117811.3117816.

[11] W. Jiang, Z. Yin, R. Liu, Z. Li, S. M. Kim, and T. He, "BlueBee: a 10,000x Faster Cross-Technology Communication via PHY Emulation," in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, Delft Netherlands, Nov. 2017, pp. 1–13. doi: 10.1145/3131672.3131678.

[12] R. Liu, Z. Yin, W. Jiang, and T. He, "WiBeacon: expanding BLE location-based services via wifi," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, New Orleans Louisiana, Sep. 2021, pp. 83–96. doi: 10.1145/3447993.3448615.

[13] R. Liu, Z. Yin, W. Jiang, and T. He, "LTE2B: time-domain cross-technology emulation under LTE constraints," in *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, New York New York, Nov. 2019, pp. 179–191. doi: 10.1145/3356250.3360022.

[14] Y. Chae, S. Wang, and S. M. Kim, "Exploiting WiFi Guard Band for Safeguarded ZigBee," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, Shenzhen China, Nov. 2018, pp. 172–184. doi: 10.1145/3274783.3274835.

[15] Z. Chi, Y. Li, Y. Yao, and T. Zhu, "PMC: Parallel multi-protocol communication to heterogeneous IoT radios within a single WiFi channel," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, Toronto, ON, Oct. 2017, pp. 1–10. doi: 10.1109/ICNP.2017.8117550.

[16] Z. Li and Y. Chen, "Achieving Universal Low-Power Wide-Area Networks on Existing Wireless Devices," in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Oct. 2019, pp. 1–11. doi: 10.1109/ICNP.2019.8888066.

[17] S. Wang, S. M. Kim, and T. He, "Symbol-Level Cross-Technology Communication via Payload Encoding," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Jul. 2018, pp. 500–510. doi: 10.1109/ICDCS.2018.00056.

[18] H. He, J. Su, Y. Chen, Z. Li, and L. Li, "Reliable Cross-Technology Communication With Physical-Layer Acknowledgement," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5175–5187, Aug. 2020, doi: 10.1109/TCOMM.2020.2992626.

[19] S. Wang *et al.*, "Networking Support for Bidirectional Cross-Technology Communication," *IEEE Transactions on Mobile Computing*, vol. 20, no. 1, pp. 204–216, Jan. 2021, doi: 10.1109/TMC.2019.2938524.

[20] S. Yao, L. Feng, J. Zhao, Q. Zhao, Q. Yang, and W. Jiang, "PatternBee: Enabling ZigBee-to-BLE Direct Communication by Offset Resistant Patterns," *IEEE Wireless Commun.*, vol. 28, no. 3, pp. 130–137, Jun. 2021, doi: 10.1109/MWC.001.2000346.

[21] W. Wang, X. Liu, Y. Yao, Y. Pan, Z. Chi, and T. Zhu, "CRF: Coexistent Routing and Flooding using WiFi Packets in Heterogeneous IoT Networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Paris, France, Apr. 2019, pp. 19–27. doi: 10.1109/INFOCOM.2019.8737525.

[22] R. Chen and W. Gao, "Enabling cross-technology coexistence for extremely weak wireless devices," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, 2019, pp. 253–261.

[23] S. Yu, X. Zhang, P. Huang, and L. Guo, "Physical-Level Parallel Inclusive Communication for Heterogeneous IoT Devices," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, London, United Kingdom, May 2022, pp. 380–389. doi: 10.1109/INFOCOM48880.2022.9796876.

[24] X. Guo, Y. He, X. Zheng, Z. Yu, and Y. Liu, "LEGO-Fi: Transmitter-Transparent CTC with Cross-Demapping," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 2125–2133. doi: 10.1109/INFOCOM.2019.8737659.

[25] S. Wang *et al.*, "X-Disco: Cross-technology Neighbor Discovery," in *2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Sep. 2022, pp. 163–171. doi: 10.1109/SECON55815.2022.9918568.

[26] Y. Chen, S. Wang, Z. Li, and T. He, "Reliable Physical-Layer Cross-Technology Communication With Emulation Error Correction," *IEEE/ACM Trans. Networking*, vol. 28, no. 2, pp. 612–624, Apr. 2020, doi: 10.1109/TNET.2020.2963985.

[27] Z. Li and T. He, "LongBee: Enabling Long-Range Cross-Technology Communication," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, Apr. 2018, pp. 162–170. doi: 10.1109/INFOCOM.2018.8485938.

[28] S. Yao, L. Feng, Q. Zhao, Q. Yang, and Y. Liang, "ERFR-CTC: Exploiting Residual Frequency Resources in Physical-Level Cross-Technology Communication," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 6062–6076, Apr. 2021, doi: 10.1109/JIOT.2020.3035692.

[29] X. Guo, Y. He, J. Zhang, and H. Jiang, "WIDE: physical-level CTC via digital emulation," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, Montreal Quebec Canada, Apr. 2019, pp. 49–60. doi: 10.1145/3302506.3310388.

[30] Y. Chen, Z. Li, and T. He, “TwinBee: Reliable Physical-Layer Cross-Technology Communication with Symbol-Level Coding,” in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, Apr. 2018, pp. 153–161. doi: 10.1109/INFOCOM.2018.8485816.

[31] Y. He *et al.*, “Cross-Technology Communication for the Internet of Things: A Survey,” *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–29, May 2023, doi: 10.1145/3530049.

[32] S. Wang, Z. Yin, Z. Li, and T. He, “Networking Support For Physical-Layer Cross-Technology Communication,” in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, Cambridge, Sep. 2018, pp. 259–269. doi: 10.1109/ICNP.2018.00042.

[33] D. Xia, X. Zheng, L. Liu, C. Wang, and H. Ma, “*c-Chirp* : Towards Symmetric Cross-Technology Communication Over Asymmetric Channels,” *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1169–1182, Jun. 2021, doi: 10.1109/TNET.2021.3061083.

[34] S. Yao, Q. Zhao, M. Zhou, L. Feng, P. Zhang, and A. Aiiad, “Supplementary material for GOP-ACK.” [Online]. Available: https://github.com/Roller44/SupForGOPACK/blob/main/GOPACKSup.pdf

[35] J. Yan, S. Cheng, Z. Li, and J. Liu, “PCTC: Parallel Cross Technology Communication in Heterogeneous wireless systems,” in *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, May 2022, pp. 67–78. doi: 10.1109/IPSN54338.2022.00013.

[36] R. Chen and W. Gao, “StarLego: Enabling Custom Physical-Layer Wireless over Commodity Devices,” in *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, Austin TX USA, Mar. 2020, pp. 80–85. doi: 10.1145/3376897.3377852.

[37] R. Chen and W. Gao, “TransFi: emulating custom wireless physical layer from commodity wifi,” in *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*, Portland Oregon, Jun. 2022, pp. 357–370. doi: 10.1145/3498361.3538946.

[38] S. Wang, W. Jeong, J. Jung, and S. M. Kim, “X-MIMO: cross-technology multi-user MIMO,” in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, Virtual Event Japan, Nov. 2020, pp. 218–231. doi: 10.1145/3384419.3430723.

[39] H.-W. Cho and K. G. Shin, “BlueFi: bluetooth over WiFi,” in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, Virtual Event USA, Aug. 2021, pp. 475–487. doi: 10.1145/3452296.3472920.

[40] Z. Li and Y. Chen, “BlueFi: Physical-layer Cross-Technology Communication from Bluetooth to WiFi,” in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, Nov. 2020, pp. 399–409. doi: 10.1109/ICDCS47774.2020.00067.

[41] D. Xia, X. Zheng, F. Yu, L. Liu, and H. Ma, “WiRa: Enabling Cross-Technology Communication from WiFi to LoRa with IEEE 802.11ax,” in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, London, United Kingdom, May 2022, pp. 430–439. doi: 10.1109/INFOCOM48880.2022.9796831.

[42] H. S. Kim and D. D. Wentzloff, “Back-Channel Wireless Communication Embedded in WiFi-Compliant OFDM Packets,” *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3181–3194, Dec. 2016, doi: 10.1109/JSAC.2016.2612078.

[43] Z. Li and Y. Chen, “BLE2LoRa: Cross-Technology Communication from Bluetooth to LoRa via Chirp Emulation,” in *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2020, pp. 1–9. doi: 10.1109/SECON48991.2020.9158446.

[44] L. Li, Y. Chen, and Z. Li, “Poster Abstract: Physical-layer Cross-Technology Communication with Narrow-Band Decoding,” in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Oct. 2019, pp. 1–2. doi: 10.1109/ICNP.2019.8888132.

[45] L. Li, Y. Chen, and Z. Li, “WiBle: Physical-Layer Cross-Technology Communication with Symbol Transition Mapping,” in *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jul. 2021, pp. 1–9. doi: 10.1109/SECON52354.2021.9491581.

[46] R. Liu, Z. Yin, W. Jiang, and T. He, “XFi: Cross-technology IoT Data Collection via Commodity WiFi,” in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*, Oct. 2020, pp. 1–11. doi: 10.1109/ICNP49622.2020.9259363.

[47] H.-W. Cho and K. G. Shin, “FLEW: fully emulated wifi,” in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, Sydney NSW Australia, Oct. 2022, pp. 29–41. doi: 10.1145/3495243.3517030.

[48] C. Shao, H. Park, H. Roh, and W. Lee, “DOTA: Physical-Layer Decomposing and Threading for ZigBee/Wi-Fi Co-Transmission,” *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 133–136, Feb. 2019, doi: 10.1109/LWC.2018.2863294.

[49] “ISO/IEC/IEEE International Standard - Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 15-4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs),” *ISO/IEC/IEEE 8802-15-4:2018(E)*, pp. 1–712, Apr. 2018, doi: 10.1109/IEEESTD.2018.8362834.

[50] S. Farahani, *ZigBee Wireless Networks and Transceivers*. USA: Newnes, 2008.

[51] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston: McGraw-Hill, 2008.

[52] R. Pawula, S. Rice, and J. Roberts, “Distribution of the Phase Angle Between Two Vectors Perturbed by Gaussian Noise,” *IEEE Transactions on Communications.*, vol. 30, no. 8, pp. 1828–1841, Aug. 1982, doi: 10.1109/TCOM.1982.1095662.

[53] “Y590s, V3 Technology,.Ltd.” https://www.v3best.com/y590s (accessed Feb. 20, 2024).

[54] “GitHub - gnuradio/gnuradio at maint-3.8.” https://github.com/gnuradio/gnuradio/tree/maint-3.8 (accessed Feb. 25, 2024).

[55] “R2020b - Updates to the MATLAB and Simulink product families.” https://www.mathworks.com/products/new_products/release2020b.html (accessed Jun. 13, 2024).

[56] “Modulation using OQPSK method - MATLAB - MathWorks.” https://www.mathworks.com/help/comm/ref/comm.oqpskmodulator-system-object.html (accessed Jun. 20, 2024).

[57] “Modulate using GMSK method - MATLAB - MathWorks.” https://www.mathworks.com/help/comm/ref/comm.gmskmodulator-system-object.html (accessed Jun. 20, 2024).

[58] “Generate white Gaussian noise samples - MATLAB wgn - MathWorks.” https://www.mathworks.com/help/comm/ref/awgn.html (accessed Jun. 21, 2024).

[59] “IEEE Standard for Low-Rate Wireless Networks,” *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, 2016, doi: 10.1109/IEEESTD.2016.7460875.

Shumin Yao received his Ph.D. degree from Macau University of Science and Technology, Macau, China, in 2022. His interests include wireless networks and Internet of Things.

Qinglin Zhao received his Ph.D. degree from the Chinese Academy of Sciences, Beijing, China, in 2005. He serves as an associate editor of IEEE Transactions on Mobile Computing and IET Communications. His research interests include Internet of Things, wireless communications and networking.

MengChu Zhou (Fellow, IEEE) received his Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY in 1990 and has been Distinguished Professor at New Jersey Institute of Technology since 2013. His interests are in Internet of Things and AI. He has 1200+ publications including 17 books, 850+ journal papers (650+ in IEEE transactions), and 31 patents. He is Fellow of IFAC, AAAS, CAA and NAI.

Li Feng received her Ph.D. degree in electronic information technology from, Macau University of Science and Technology (MUST), Macao, China, in 2013. Her current research interests include Internet of Things, wireless and mobile networks, and software defined networking.

Peiyun Zhang (M’16–SM’17) received her Ph.D. from the Nanjing University of Science and Technology, Nanjing, China in 2008. Her interests include blockchains, and cloud/ trust computing. She has published over 70 papers in these fields.

Aiiad Albeshri Received Ph.D. degree in Information Technology from Queensland University of Technology, Brisbane, Australia. His current research focuses on Information Security and Big Data.