

# Supplementary Material for GOP-ACK

Shumin Yao, Qinglin Zhao, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Li Feng,

Peiyun Zhang, *Senior Member, IEEE*, and Aiiad Albeshri

## Appendix A. Design extension

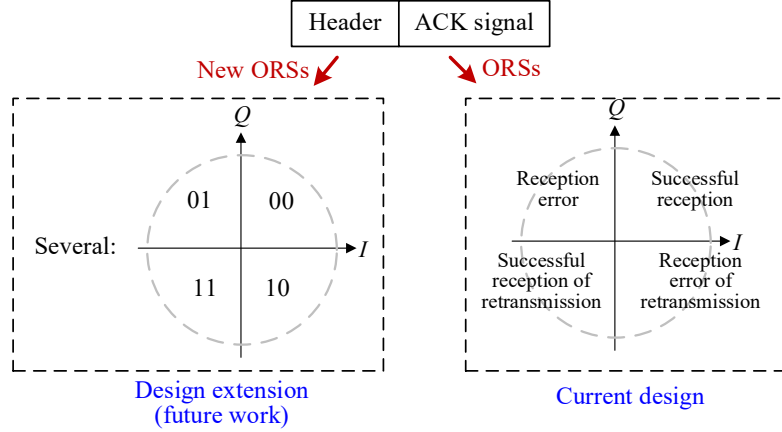


Fig. I. Design extension for the scenario where multiple devices coexist.

As a future extension of our design, we can easily add a lightweight header to each ACK signal in scenarios where multiple devices coexist. For instance, as illustrated in Fig. I, we can append each ACK signal with a header containing essential CTC control information and encode it with another type of ORS, where different quadrants are associated with distinct bit pairs.

We label our header as lightweight because it is specifically designed to convey control information exclusively for cross-technology communication. In contrast, the packet headers in current CTC ACK designs primarily carry control information for in-technology communication, rendering them redundant in the context of cross-technology scenarios. Moreover, the robustness of our header far surpasses existing headers since it is encoded with ORSs.

## Appendix B. Case study: ZigBee-to-WiFi feedback

In this section, we examine a scenario in which a ZigBee node, acting as the ACK sender, acknowledges the receipt of a data packet by sending an ACK message (indicating either successful reception or a reception error) back to a BLE node, which serves as the ACK receiver. We present the design of ZigBee-to-BLE feedback by following the guidelines presented in Section III.

As shown in Fig. II, The ZigBee-to-WiFi link satisfies our minimal firmware requirements for ORS-based ACK transmission and reception. Specifically, ZigBee TX firmware is equipped with an offset quadrature phase-shift keying (OQPSK) modulator [1], while WiFi RX firmware is equipped a phase-shift calculator [2]. These built-in firmware modules enable ZigBee to generate OQPSK-based ORS as ACK signals following the core principles outlined in Section III.B, while enabling WiFi to then sample these ORS signals and process them using their phase-shift calculator, detecting an ORS when the calculated phase shift equals zero.

The remainder of this section is organized as follows. We first present an overview of our design, highlighting the reuse of existing ZigBee and WiFi firmware modules according to Section III.E guidelines. We then detail the ORS construction following Section III.C guidelines, and finally present the design specification for robust and lightweight ORS-based ACK signals following Section III.D guidelines.

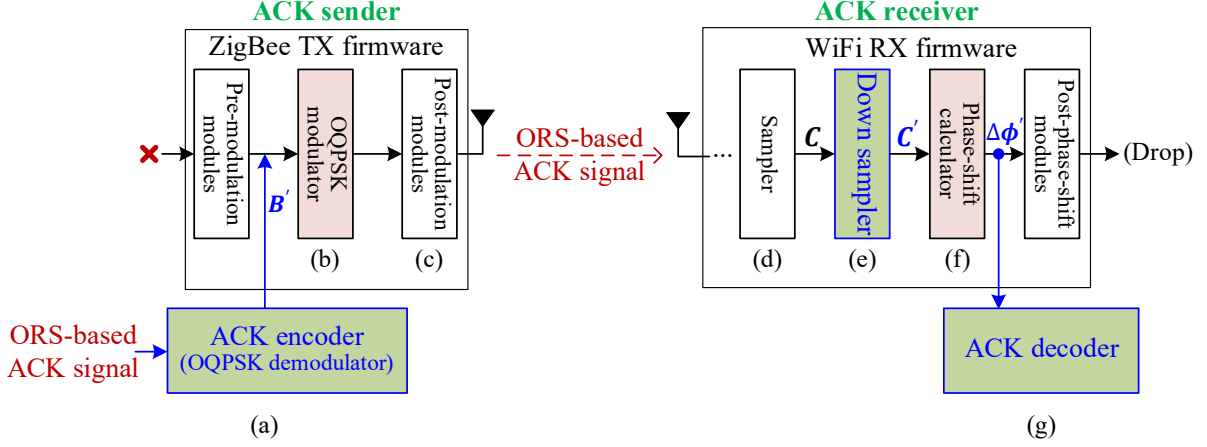


Fig. II. In our ZigBee-to-WiFi feedback, an ORS-supporting ZigBee TX chain shown in (a)-(c) and an ORS-supporting WiFi RX chain shown in (d)-(g).

### B.1 Overview of ZigBee-to-WiFi feedback

As shown in Fig. II, our ZigBee-to-WiFi feedback design consists of two main components: an ORS-supporting ZigBee TX chain and an ORS-supporting WiFi RX chain. Following Section III.E's firmware-modifying method, we integrate software-defined modules (SDMs) with existing firmware: the TX chain integrates an SDM (ACK encoder) with conventional ZigBee TX firmware to reuse ZigBee's modulator, while the RX chain integrates two SDMs (down sampler and ACK decoder) with conventional WiFi firmware to reuse WiFi's phase-shift calculator.

#### B.1.1) ORS-supporting ZigBee TX chain

Our ORS-supporting ZigBee TX chain transmits an ACK message via the following three steps, as shown in Figs. II(a)-(c).

1. The ACK encoder transforms an ACK message (from the upper layer) into a carefully constructed binary chip sequence  $B'$ . This sequence differs from  $B$  in the ZigBee-to-BLE feedback case, to be explained in Section B.2.2).
2. ZigBee's OQPSK modulator modulates the chip sequence  $B'$  into a set of ORSs (instead of regular ZigBee signals), because of our special construction of  $B'$ .
3. The post-modulation modules of ZigBee TX chain amplify and send out the ORSs.

#### B.1.2) ORS-supporting WiFi RX chain

Upon receiving the ORSs, our ORS-supporting WiFi RX chain transforms them to an ACK message via the following four steps, as shown in Figs. II(d)-(g).

1. WiFi's built-in sampler samples the received ORS to obtain a set of complex sampling instances  $C$ .
2. The down sampler, inserted after WiFi's built-in sampler, performs perform 1/5 down sampling by reserving the first of every five instances in  $C$ , resulting in generating  $C'$ . Section B.2.2) explains the rationale for this down sampling.
3. WiFi's phase-shift calculator calculates the phase shifts between each sampling instance and its 16 subsequent instances in  $C'$ . The set of calculation results is formed into a set  $\Delta\phi$ .
4. The ACK decoder obtains the ACK message according to  $C'$  and  $\Delta\phi$ , to be detailed in Section B.3.

## B.2 Construction of OQPSK-based ORS

Here, we design OQPSK-based ORS for ZigBee-to-WiFi feedback. We first examine the characteristics of ZigBee modulator and WiFi phase-shift calculator. Based on these characteristics, we then specify our ORS design following the guidelines in Section III.C.

### B.2.1) ZigBee modulator and WiFi phase-shift calculator

As the ZigBee modulator has been detailed in Section IV.B.A), we focus here on the WiFi phase-shift calculator. The calculator receives a set of sampling instances from the pre-phase-shift modules at  $0.05 \mu s$  intervals. It calculates the phase shift difference between the  $n$ -th and  $(n-16)$ -th sampling instances, for  $n = 1, 2, \dots$ . Thus, the default phase-shift interval of WiFi is  $0.8 \mu s$  ( $16 \times 0.05 \mu s$ ).

### B.2.2) ORS construction

Following the design principle in Section III.B, we design an ORS to consists of two identical waves. Recall that an ORS requires

$$T_w = T_s. \quad (1)$$

In ZigBee-to-BLE feedback explained in Section IV of the main file, ZigBee creates a wave of  $T_w = 1 \mu s$ , while BLE adopts a sampling interval of  $T_s = 1 \mu s$  to calculate one phase shift, thus meeting the requirement (1). However, in ZigBee-to-WiFi feedback, if ZigBee still creates a wave of  $T_w = 1 \mu s$ , while WiFi adopts the default sampling interval of calculating one phase shift, which is  $T_s = 0.8 \mu s$ , then  $T_w \neq T_s$ , violating (1).

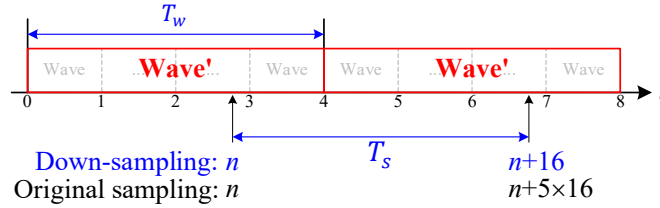


Fig. III. For ZigBee-to-WiFi feedback, ZigBee constructs a ZigBee-to-WiFi wave (called wave') that is comprised of 4 ZigBee-to-BLE waves, while WiFi performs down-sampling with 1/5 of the original sampling rate.

To meet the requirement (1) for ZigBee-to-WiFi feedback, we follow the guidelines in Section III.C to make the following modifications in comparison with ZigBee-to-BLE feedback:

*ZigBee side modifications.* ZigBee constructs a ZigBee-to-WiFi wave (called wave') that is comprised of 4 ZigBee-to-BLE waves and hence the length of one wave is  $T_w = 4 \times 1 \mu s$ , as shown in Fig. III.

*WiFi side modifications.* WiFi performs down-sampling with 1/5 of the original sampling rate, as shown in Fig. III. This adjusts the sampling interval of calculating one phase shift to  $T_s = 5 \times 0.8 = 4 \mu s$ , ensuring  $T_w = T_s$  and meeting the requirement (1). **This necessitates the down sampler after WiFi's built-in sampler, as described in Section B.1.2).**

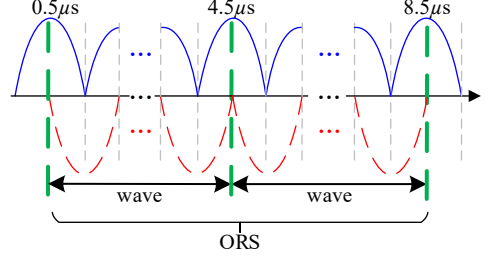


Fig. IV An ORS for ZigBee-to-WiFi feedback.

*ORS modifications.* Recall that an OQPSK signal is the superposition of I-pulses and  $0.5\mu s$ -delayed Q-pulses, each pulse lasting for  $1\mu s$ . Then, an OQPSK-based wave lasting for  $T_w = T_s = 4\mu s$  is the superposition of two half I-pulses, three complete I-pulses, and four complete Q-pulses, as shown in Fig. IV. Thus, to construct an ORS consisting of two identical waves, ZigBee superposes three half I-pulses, seven complete I-pulses, and eight complete Q-pulses and uses the  $8\mu s$  portion from  $0.5\mu s$  to  $8.5\mu s$  for this purpose. To achieve this, as illustrated in Section B.1.1), ZigBee sets one ZigBee-to-WiFi ACK chip sequence to  $\mathbf{B}' = [\mathbf{B}, \mathbf{B}, \mathbf{B}, \mathbf{B}]$  in the TX firmware of ZigBee-to-BLE feedback, where  $\mathbf{B}$  is one ZigBee-to-BLE ACK chip sequence.

### B.3 ORS-based ACK design for ZigBee-to-WiFi feedback

In this section, we follow the guidelines presented in Section III.D to design ORS-based ACK ZigBee-to-WiFi feedback.

We maintain the same ACK design as in ZigBee-to-BLE feedback, with modifications only on the WiFi side. Upon receiving an ACK signal consisting of  $M+1$  ORSs, WiFi keep sampling the signal, as shown in Figs. II(d)-(e). As explained in Section B.2.1), the default sampling interval of WiFi is  $0.05\mu s$ . After 1/5 down sampling, the new sampling interval of  $0.25\mu s$ . As a result, WiFi samples each wave' of the ORSs 16 times, resulting in  $16(M+1)$  sampling instances,  $\mathbf{C}' = \{C_m\}_{m=1}^{16(M+1)}$ , for each ACK signal. Next, as show in Fig. II(f), WiFi's phase-shift calculator calculates the phase shift between  $C_m$  and  $C_{m+16}$  for  $m = 1, \dots, 16M$  and then obtains a set  $\Delta\phi = \{\Delta\phi_{m,i} \mid m \in \{1, \dots, M\}, i \in \{1, \dots, 16\}\}$ , where  $\Delta\phi_{m,i}$  is the  $i$ -th phase-shift value for ORS  $m$ . Then, as shown in Fig. II(g), WiFi feeds  $\mathbf{C}'$  and  $\Delta\phi$  into its ACK decoder, which transforms them into an ACK message via the following two stages.

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

- *C1: Condition of a successful busy channel detection.* Let  $\lambda = \frac{\sum_{m=1}^{16(M+1)} |C_{16m}|^2}{16(M+1)}$  denote the power of  $\{C_m\}_{m=1}^{16(M+1)}$ . C1 is satisfied if  $\lambda$  is greater than a threshold  $\hat{\lambda}$ , namely,
$$\lambda > \hat{\lambda}. \quad (2)$$
- *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  $\hat{N}_{ORS}$ , namely,

$$N_{ORS} > \hat{N}_{ORS}. \quad (3)$$

Note that an ORS consists of two identical waves, each sampled 16 times. In ideal conditions, a decoder detects an ORS if the phase shifts between the two identically positioned sampling instances of the waves are 0. In the presence of noise, however, these phase shifts are not always 0. Hence, in our design, a decoder detects an ORS  $m$  if there are more than  $N_\phi$  of the ORS's phase values are smaller than a threshold  $\Delta\hat{\phi}$ , namely,

$$\sum_{i=1}^{16} \llbracket \Delta\varphi_{m,i} < \Delta\hat{\varphi} \rrbracket > N_{\varphi}, \quad (4)$$

where  $\llbracket s \rrbracket = 1$  if the statement  $s$  is true and 0 otherwise.

**2. ACK-type detection.** The ACK decoding in ZigBee-to-WiFi feedback follows the same principles as in ZigBee-to-BLE feedback. Recall that a type- $j$  ACK signal consists of  $M$  type- $j$  ORSs. WiFi decodes a received ACK signal via two steps:

- *ORS type detection.* Recall that an ORS consists of two consecutive and identical waves, each sampled 16 times. WiFi infers the ORS type by analyzing the quadrant positions of the first wave's sampling instances in the constellation diagram. Let  $N_s^j$  be the number of first wave's sampling instances appearing in quadrant  $j$ . WiFi calculates  $j_1 = \underset{j}{\operatorname{argmax}} \{N_s^j\}_{j=1}^4$  and infers that a type- $j_1$  ORS is detected.
- *ACK signal type detection.* Let  $N_{ORS}^j$  be the number of detected type- $j$  ORSs. WiFi first calculates  $j_0 = \underset{j}{\operatorname{argmax}} \{N_{ORS}^j\}_{j=1}^4$  and infers that the ACK type is  $j_0$ . Finally, it 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  $j_0$ .

*Remarks:* A typical WiFi-to-ZigBee transmission with feedback follows the pattern: packet/SIFS/ACK. That is, WiFi sends a packet to ZigBee, then waits for a short interframe space (SIFS) time, and finally receives an ACK from ZigBee. Nevertheless, WiFi retransmits the packet if one of the following cases occurs: 1) the ACK indicates that the packet is not successfully received by ZigBee, 2) WiFi's ACK decoder fails to decode the ACK, and 3) WiFi 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).

## Appendix C. Supplementary of Tables

Below, we show the tables that helps better understanding the main file of the paper. More specifically, TABLE A lists the the abbreviations used in the paper, TABLE B shows the details of TABLE 2 in the main file, and TABLE C summarizes the main notations of our model.

TABLE A. A list of abbreviations.

Abbreviation	Full name
ACK	Acknowledgement
BLE	Bluetooth Low Energy
BSU	Basic Signal Unit
CTC	Cross-Technology Communication
GOP-ACK	General and Offset-resistant Physical Acknowledgment
OQPSK	Offset Quadrature Phase-Shift Keying
ORS	Offset-Resistant Signal
RX	Receiving
SDM	Software-Defined Module
SNR	Signal-to-Noise Ratio

TABLE B. Mapping among bit sequences, ORS types, and ACK types.

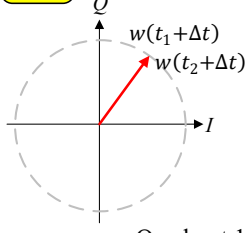
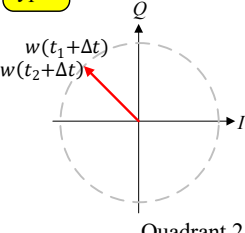
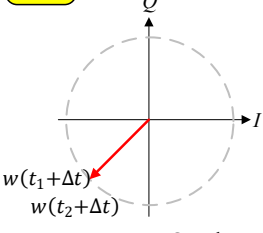
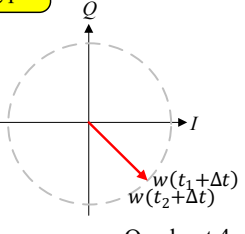
Special bit sequence	Odd bits	I-pulses type	ORS type	ACK type
	Even bits	Q-pulses type		
1,1,1,1,1	1,1,1	Positive	Type-1  Quadrant 1	Type-1: successful reception
	1,1	Positive		
0,1,0,1,0	0,0,0	Negative	Type-2  Quadrant 2	Type-2: reception error
	1,1	Positive		
0,0,0,0,0	0,0,0	Negative	Type-3  Quadrant 3	Type-3: successful reception of retransmission
	0,0	Negative		
1,0,1,0,1	1,1,1	Positive	Type-4  Quadrant 4	Type-4: reception error of retransmission
	0,0	Negative		

TABLE C. Main notions and their descriptions.

Notation	Description
$C_m$	The $m$ -th sampling instance.
$C_m^I$	The in-phase component of $C_m$ .
$C_m^Q$	The quadrature component of $C_m$ .
$x_I(t)$	The in-phase component of an ACK signal.
$x_Q(t)$	The quadrature component of an ACK signal.
$\Delta\varphi_m$	The $m$ -th phase shift value.
$\Delta\hat{\varphi}_{th}$	The phase shift threshold.
$\lambda$	The power of sampling instances.
$\hat{\lambda}_{th}$	The power threshold of sampling instances.
$N_{ORS}$	The total number of detected ORSs.
$N_{ORS}^j$	The number of detected type- $j$ ORSs.
$n_j$	The value of $N_{ORS}^j$ .
$\hat{N}_{ORS}^{th}$	The threshold of $N_{ORS}$ for ACK signal detection.
$\sigma^2$	The noise power.
$\tau$	The sampling interval.
$\Delta T$	The sampling offset.
$\Delta t$	The value of $\Delta T$ .
$P_{ACK}^s$	The successful ACK decoding probability.
$P_{C1}(\Delta t)$	The successful probability of detecting busy channel under sampling offset $\Delta t$ .
$P_{C2}(\Delta t)$	The probability of $N_{ORS} > \hat{N}_{ORS}^{th}$ under sampling offset $\Delta t$ .
$P_{C3}(\Delta t)$	The successful probability of decoding an ACK type under sampling offset $\Delta t$ .
$M$	The total number of ORSs in an ACK signal.
$M'$	The number of waves in a long or short ACK signal.
$\mathcal{T}$	The total number of ACK types.
$j_0$	The inferred ACK type.
$j^*$	The actual ACK type.
$p_{j^*}^j(\Delta t)$	The probability that the ACK decoder identifies a type- $j^*$ ORS as a type- $j$ ORS under sampling offset $\Delta t$ .
$\Omega$	A complete transmission time.
$T_{pkt}$	The duration of a packet transmission.
$T_{SIFS}$	The duration of a short interframe space.
$T_{ACK}$	The duration of an ACK signal.
$T_w$	The duration of one wave.

## Appendix D. Details of no-firmware-modifying method

In this section, we provide more details about the no-firmware-modifying method, which reuses all modules in firmware. Before this, we first specify the TX and RX firmware.

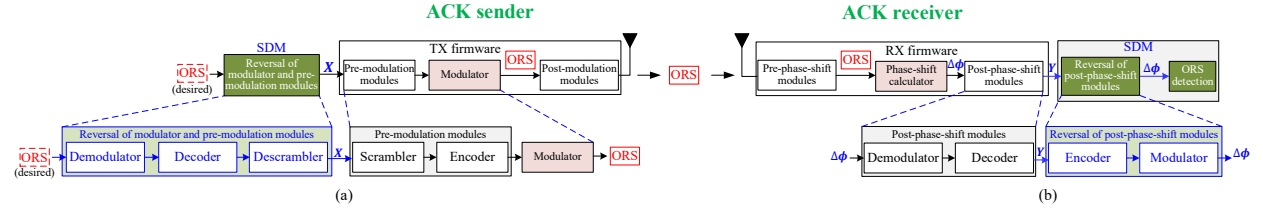


Fig. V. No-firmware-modifying method: (a) reversal of modulator and pre-modulation modules; (b) reversal of post-phase-shift modules.

### D.1 TX & RX firmware

Firmware is a type of software that provides low-level control over radio hardware. 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.

A TX firmware comprises a one-directional TX path with pre-modulation modules, a modulator, and post-modulation modules, as shown in Fig. V. Along the TX path, an upper-layer message passes through these modules in a predefined order, resulting in its conversion into a signal for transmission. First, following the TX path, the message is processed by the pre-modulation modules (e.g., scrambler, encoder, etc.) to become a bit sequence, which is then converted into a signal by the modulator. Finally, the signal passes through the post-modulation modules (e.g., filter, amplifier, upconverter, etc.) to be sent out.

An RX firmware consists of a one-directional RX path with pre-phase-shift modules, a phase-shift calculator, and post-phase-shift modules. Along the RX path, a received signal passes through these modules in a predefined order, resulting in its conversion into a message. First, following the RX path, the signal passes through the pre-phase-shift modules (e.g., a downconverter, a filter, a sampler, etc.) to be converted into sampling instances, of which corresponding phase shifts are then calculated by the phase-shift calculator. Finally, the phase shifts are converted into a message by the post-phase-shift modules (e.g., a demodulator, a decoder, etc.).

### D.2 Method

Consider TX and RX firmware that only allows access to the input and output interfaces. On an ACK sender side, we aim to reuse the entire TX path of the TX firmware to transmit a desired ORS. We can do so by following two steps below. First, we reuse the pre-modulation modules and the modulator in the TX firmware to generate the ORS. To achieve this, we need to construct a special input,  $X$ , as an input to the TX firmware. As  $X$  passes through the pre-modulation modules and modulator along the TX path, it will eventually be converted into the desired ORS. To construct  $X$  with this characteristic, like [1], [3]–[9], we may create an SDM that reverses the pre-modulation and modulation operations in the TX firmware. For example, as shown in Fig. V(a), if the pre-modulation modules are a scrambler and an encoder, we may create an SDM consisting of a demodulator, a decoder, and a descrambler in sequence. As a result, when we pass the desired ORS through the SDM, we can obtain  $X$ . In the second step, the ORS is sent out by reusing the post-modulation modules, which is straightforward. In this way, we can transmit the desired ORS without modifying the TX firmware.

On an ACK receiver side, we reuse the entire RX path of the RX firmware for ORS receptions. We can do so by following three steps below. First, upon detecting a signal, we reuse all the modules on the RX path to receive and decode the signal, obtaining the output,  $Y$ , of the RX firmware. Then, like [10]–[13],



etc., we employ another SDM, which reverses the operation of the post-phase-shift modules, to process  $\mathbf{Y}$  and yield the desired phase-shift,  $\Delta\phi$ , namely, the output of the phase-shift calculator in the RX firmware. For example, as shown in Fig. V(b), if the post-phase-shift modules consist of a demodulator and a decoder, we may create an SDM consisting of an encoder and a modulator in sequence. As a result, when we pass  $\mathbf{Y}$  through the SDM, we can obtain  $\Delta\phi$ . Finally, if  $\Delta\phi$  is zero, we detect an ORS. In this way, we can achieve an ORS reception without modifying the RX firmware.

## Appendix E. Supplementary of the performance model

Here, we first present the details of calculating,  $P_{C1}(\Delta t)$ , which is the probability of satisfying C1. Then, we calculate the joint probability mass function of  $N_{ORS}^1, \dots, N_{ORS}^T$  under  $\Delta t$ ,  $P_{\Delta t}(n_1, \dots, n_T)$ .

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

In GOP-ACK, a wave lasts for  $T_w = \alpha T_b$ . When receiving a long or short ACK signal consisting of  $KM'$  waves, the decoder samples its each wave  $K$  times at different positions. Then, the average power of these  $KM'$  sampling instances,  $\lambda$ , is calculated as

$$\lambda = \frac{\sum_{m=1}^{KM'} |C_m|^2}{KM'}. \quad (5)$$

According to C1, a busy channel is successfully detected when  $\lambda > \hat{\lambda}$ . Therefore, the successful probability  $P_{C1}(\Delta t)$  of detecting a busy channel is given as

$$\begin{aligned} P_{C1}(\Delta t) &= P(\lambda > \hat{\lambda}) \\ &= P\left(\frac{\sum_{m=1}^{KM'} |C_m|^2}{KM'} > \hat{\lambda}\right) \\ &= P\left(\sum_{m=1}^{KM'} |C_m|^2 > KM' \hat{\lambda}\right) \\ &= P\left(\sum_{m=1}^{KM'} [(C_m^I)^2 + (C_m^Q)^2] > KM' \hat{\lambda}\right). \end{aligned} \quad (6)$$

Here, as specified in (7) in the main file,  $C_1^I, C_1^Q, \dots, C_{KM'}^I$ , and  $C_{KM'}^Q$  are  $2KM'$  Gaussian distributed random variables with the same variance  $\sigma^2/2$  but different means, which are  $x_I(\tau + \Delta t), x_Q(\tau + \Delta t), \dots, x_I(KM'\tau + \Delta t), x_Q(KM'\tau + \Delta t)$ , respectively. Furthermore, they are mutually independent since they correspond to different components of different waves. Then, according to [14],  $\sum_{m=1}^{KM'} [(C_m^I)^2 + (C_m^Q)^2]$  is a non-central chi-squared random variable with  $2M'$  degrees of freedom and a non-centrality parameter  $\sum_{m=1}^{LM'} [x_I^2(m\tau + \Delta t) + x_Q^2(m\tau + \Delta t)]$ . Then we can calculate have:

$$P_{C1}(\Delta t) = Q_{KM'}\left(\frac{\sqrt{\sum_{m=1}^{KM'} (x_I^2(m\tau + \Delta t) + x_Q^2(m\tau + \Delta t))}}{\sigma}, \frac{\sqrt{M' \hat{\lambda}}}{\sigma}\right), \quad (7)$$

where  $Q_{LM'}(a, b) = \frac{1}{a^{KM'-1}} \int_b^\infty \varepsilon^{KM'} e^{-\frac{a^2 + \varepsilon^2}{2}} I_{KM'-1}(a\varepsilon) d\varepsilon$  is the generalized Marcum Q function of order  $KM'$  and  $I_{KM'-1}$  is the modified Bessel function of the first kind.

## E.2 Calculation of $P_{\Delta t}(n_1, \dots, n_J)$

Note that a type- $j^*$  ACK signal consists of  $M$  type- $j^*$  ORSs,  $j^* = 1, \dots, J$ . When receiving this signal, a decoder detects the type of each ORS independently. Under a sampling offset  $\Delta t$ , for **each ORS detection** operation, the decoder might identify type  $j^*$  as type  $j$  with probability  $p_{j^*}^j(\Delta t)$ ,  $j = 1, \dots, J$ . Recall that  $N_{ORS}^j$  is the number of detected type- $j$  ORSs. Then,  $\{N_{ORS}^j\}_{j=1}^J$  follows a multinomial distribution [15] with a probability mass function  $P_{\Delta t}(n_1, \dots, n_J) = P_{\Delta t}(N_{ORS}^1 = n_1, \dots, N_{ORS}^J = n_J \text{ under } \Delta t)$ :

$$P_{\Delta t}(n_1, \dots, n_J) = \begin{cases} \frac{M!}{\prod_{j=1}^J n_j!} \prod_{j=1}^J \left(p_{j^*}^j(\Delta t)\right)^{n_j}, & \text{when } \sum_j n_j = M, \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

We now explain how to calculate  $p_{j^*}^j(\Delta t)$  in (8). For each ORS, the decoder samples each wave  $K$  times. Under non-ideal conditions, each sampling instance may fall into any of  $J$  quadrants in the constellation diagram. For a type- $j^*$  ORS, the decoder processes  $K$  sampling instances of the first wave and independently detects the quadrant of each instance. Let  $A_j$  denote the area of quadrant  $j$ . The decoder infers the ORS as type  $j$  if  $A_j$  contains the largest number of sampling instances among all quadrants. With  $L_j$  denoting the number of instances in  $A_j$ ,  $p_{j^*}^j(\Delta t)$  is calculated as:

$$\begin{aligned} p_{j^*}^j(\Delta t) &= P_{j^*} \left( L_j > \max_{j' \neq j} L_{j'} \right) \\ &= \sum_{\text{all } (L_1, \dots, L_J) \in D_j} P_{j^*} \left( L_j > \max_{j' \neq j} L_{j'} \right), \end{aligned} \quad (9)$$

where  $D_j \triangleq \left\{ L_1, \dots, L_J \mid L_j > \max_{j' \neq j} L_{j'} \text{ and } \sum_{j=1}^J L_j = k \right\}$ .

For large  $K$  values, calculating (9) becomes computationally intensive due to numerous cases in  $D_j$ . To address this issue, we apply a normal approximation [15]:

$$\begin{aligned} P_{j^*} \left( L_j > \max_{j' \neq j} L_{j'} \right) &\approx \prod_{j' \neq j} P_{j^*} (L_j > L_{j'}) \\ &\approx \prod_{j' \neq j} P_{j^*} (L_j - L_{j'} > 0). \end{aligned} \quad (10)$$

Let  $Z_j = L_j - L_{j'}$ . Then  $P_{j^*}(L_j - L_{j'} > 0)$  in (10) is calculated as follows:

$$P_{j^*}(L_j - L_{j'} > 0) \approx P_{j^*}(Z_j > 0) = 1 - \Phi \left( \frac{0 - \hat{\mu}_j}{\hat{\sigma}_j} \right), \quad (11)$$

where  $\Phi(\cdot)$  is the cumulation function of the standard normal distribution, and  $\hat{\mu}_j$  and  $\hat{\sigma}_j^2$  are the expected value and variance of  $Z_j$ , respectively. These are calculated as:

$$\begin{aligned} \hat{\mu}_j &= \mu_j - \mu_{j'} \\ \hat{\sigma}_j^2 &= \sigma_j^2 + \sigma_{j'}^2 - 2\text{Cov}(L_j, L_{j'}). \end{aligned} \quad (12)$$

Here,  $\mu_j$  and  $\sigma_j^2$  are the expected value and variance of  $L_j$ , respectively, and  $\text{Cov}(L_j, L_{j'})$  is the covariance between  $L_j$  and  $L_{j'}$ . They can be obtained as follows. Let  $C[k] = C^I[k] + jC^Q[k]$ ,  $k = 1, \dots, K$ , denote the  $k$ -th sampling instances for the first wave of one received type- $j^*$  ORS. Define  $p_{j^*}^{j,k}(\Delta t) \triangleq p_{j^*}^j(C[k] \in A_j \text{ under } \Delta t)$  as the probability that  $C[k]$  falls within  $A_j$ . Then,

$$\begin{aligned}
\mu_j &= \sum_{k=1}^K p_{j*}^{j,k}(\Delta t), \\
\sigma_j^2 &= \sum_{k=1}^K p_{j*}^{j,k}(\Delta t) \left(1 - p_{j*}^{j,k}(\Delta t)\right), \\
\text{Cov}(L_j, L_{j'}) &= - \sum_{k=1}^K p_{j*}^{j,k}(\Delta t) p_{j'}^{j',k}(\Delta t).
\end{aligned} \tag{13}$$

Assuming independent in-phase and quadrature components of ORSs:

$$p_{j*}^{j,k}(\Delta t) = \iint_{C[k] \in A_j} f(i, q) di dq, \tag{14}$$

where  $f(i, q)$  represents the joint probability density function of two independent Gaussian random variables  $C^I[k]$  and  $C^Q[k]$  and is given as:

$$f(i, q) = \frac{1}{2\pi\sigma^2} e^{-\frac{\left[(i-x_I(k\tau+\Delta t))^2 + (q-x_Q(k\tau+\Delta t))^2\right]}{2\sigma^2}}. \tag{15}$$

## Appendix F. Supplementary of evaluation

We verify the accuracy of our theoretical model and compare the performance GOP-ACK with that of the state of the art via simulations. Our simulator is developed based on MATLAB 2020b [16]. It consists of ZigBee, BLE, AWGN channel, and WiFi modules. They are based on the OQPSK modulator module [17], the Gaussian minimum shift keying modulator module [18], the white Gaussian noise generator module [19], and WLAN toolbox [20], respectively.

### F.1 Model verification

We verify the successful probability of detecting a busy channel,  $P_{C1}(\Delta t)$ , the probability of detecting enough ORSs,  $P_{C2}(\Delta t)$ , and the successful probability of detecting ACK type,  $P_{C3}(\Delta t)$ , under different sampling offset  $\Delta t$  and SNRs. The parameter settings are the same as in Section VI.B in the main file.

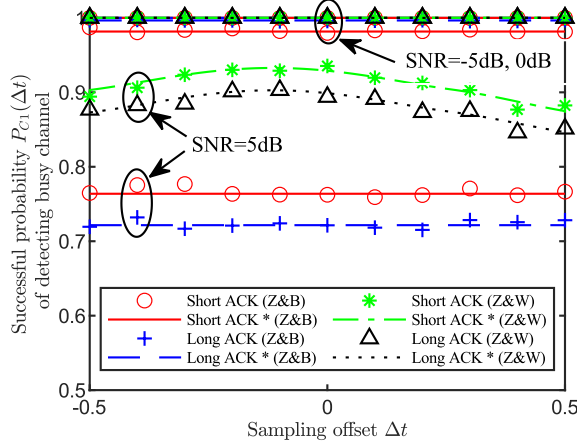


Fig. VI.  $P_{C1}(\Delta t)$  vs. different  $\Delta t$  and SNR.

Fig. VI plots  $P_{C1}(\Delta t)$  versus  $\Delta t$  for SNR values of -5, 0, and 5 dB, where  $\Delta t = -0.5, \dots, 0.5 \mu s$ . From this figure, we have the following observations:

- In ZigBee-to-BLE feedback,  $P_{C1}(\Delta t)$  remains stable across different  $\Delta t$  values at any given SNR. This manifests that when detecting a busy channel, GOP-ACK can well resist sampling offsets that are inherent in ZigBee-to-BLE feedback.
- For ZigBee-to-WiFi feedback,  $P_{C1}(\Delta t)$  shows slight decreases with increasing  $|\Delta t|$  at high SNR (e.g., 5dB). WiFi samples each wave 16 times and infers a busy channel when the average power  $\lambda$  across all sampling instances exceeding threshold  $\hat{\lambda}$ . At large  $|\Delta t|$ , some edge samples fall outside the ORS range. While this can trigger false alarms, ZigBee-to-WiFi feedback still maintains higher  $P_{C1}(\Delta t)$  than ZigBee-to-BLE feedback.
- For both feedback cases, given  $\Delta t$ ,  $P_{C1}(\Delta t)$  decreases with SNR increasing. The reason is that when SNR is high,  $\lambda$  is often lower than,  $\hat{\lambda}$ , leading to more frequent miss detections. These results imply that we should set a smaller  $\hat{\lambda}$  when SNR is high.
- For both feedback cases, given  $\Delta t$ ,  $P_{C1}(\Delta t)$  for short signals is higher (lower) than that for long signals when SNR is low (high). This implies that for maximizing the busy channel detection performance solely, ZigBee should adopt a short (long) ACK signal when SNR is low (high).

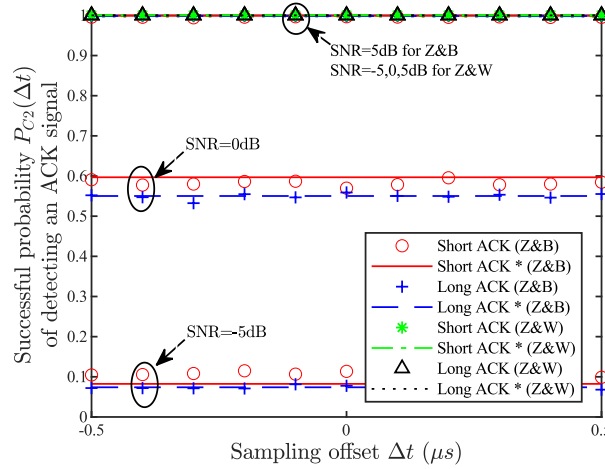


Fig. VII.  $P_{C2}(\Delta t)$  vs. different  $\Delta t$  and SNR.

Fig. VII shows  $P_{C2}(\Delta t)$  versus  $\Delta t$  for SNR values of -5, 0, 5 dB, where  $\Delta t = -0.5, \dots, 0.5\mu s$ . From it, we conclude that:

- $P_{C2}(\Delta t)$  does not vary with  $\Delta t$  at any given SNR for both feedback cases, demonstrating GOP-ACK's resilience to inherent CTC sampling offsets in ACK signal detection.
- ZigBee-to-WiFi feedback maintains high  $P_{C2}(\Delta t)$  at any given SNR value, while ZigBee-to-BLE feedback shows three distinct performance levels: low  $P_{C2}(\Delta t)$  at SNR = -5 dB, moderate at 0 dB, and high at 5 dB. This performance advantage of WiFi feedback stems from its higher sampling rate, which provides more phase-shift measurements and thus enhances ORS-based ACK signal detection reliability.
- $P_{C2}(\Delta t)$  shows nearly identical performance between shortened and regular-length signals across most conditions, with only a slight difference observed in ZigBee-to-BLE feedback at SNR = 0 dB. Overall, this confirms that our overlapping-ORS approach effectively maintains ACK signal detection performance while reducing signal length.

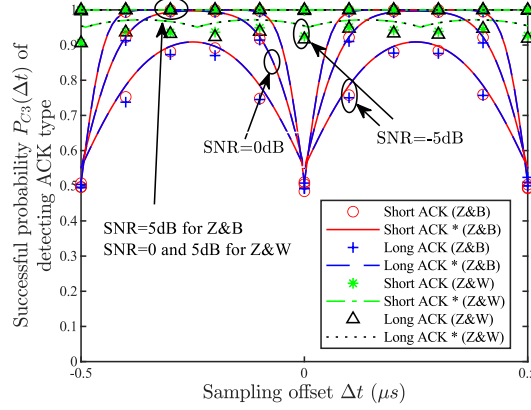


Fig. VIII.  $P_{C3}(\Delta t)$  vs. different  $\sigma^2$  and  $\Delta t$ .

Fig. VIII shows  $P_{C3}(\Delta t)$  versus  $\Delta t$  for SNR values of -5, 0, 5 dB, where  $\Delta t = -0.5, \dots, 0.5 \mu s$ . From it, we have:

- Both feedback cases show decreasing  $P_{C3}(\Delta t)$  with increasing SNR at any given  $\Delta t$ , indicating the negative impact of noises on ACK decoding performance.
- For each feedback case, short and long signals achieve the same  $P_{C3}(\Delta t)$  at any given  $\Delta t$  and SNR value, demonstrating that our overlapping-ORS approach preserves ACK decoding performance while reducing signal length.
- In ZigBee-to-BLE feedback, given a SNR value  $P_{C3}(\Delta t)$  for decrease as  $|\Delta t|$  approaches 0 or  $0.5 \mu s$ . We explain the reason for the case of  $|\Delta t|$  being close to 0. As shown in Figs. 7(d') and (e') in the main file, when  $|\Delta t| \approx 0$ , we have  $(C^I, C^Q) \approx (0, -1)$ , where  $C^I$  and  $C^Q$  are the in-phase and quadrature components of the sampling instance  $C$ , respectively. In the presence of noise, with probability 0.5,  $C$  might appear in quadrants 3 or 4, e.g., when  $(C^I, C^Q) \approx (-0.01, -0.99)$  or  $(+0.01, -0.99)$ . Since sampling offsets are inherent in CTC,  $|\Delta t|$  is rarely close to 0. Thus,  $P_{C3}(\Delta t)$  of GOP-ACK is high in general.
- In ZigBee-to-WiFi feedback, given a SNR value,  $P_{C3}(\Delta t)$  shows minor decreases as  $|\Delta t|$  approach 0, 0.25 or  $0.5 \mu s$ . At these specific offsets, some WiFi sampling instance fall near quadrant boundaries, making them slightly more susceptible to noise-induced quadrant crossings. Despite these minor variations, WiFi's higher sampling rate maintains robust performance across most sampling offsets, and the impact on decoding errors remains minimal.

Lastly, in Figs. IV-VI, the close match between the theoretical and simulation curves manifests that our performance model is very accurate.

## F.2 Comparison with the state of the art

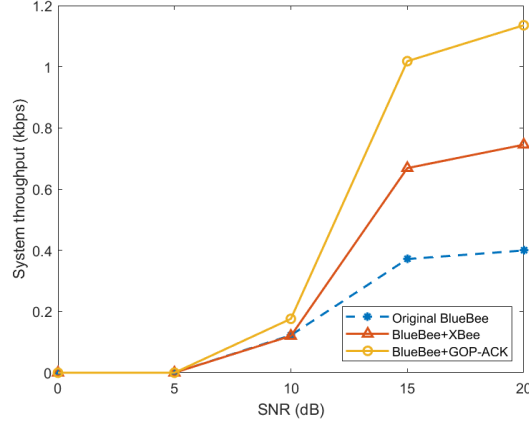


Fig. IX. ZigBee-to-BLE throughput of the original BlueBee system, the Blue-Bee+GOP-ACK system, and the BlueBee+XBee system under different SNRs.

Consider CTC communications between BLE and ZigBee. We compare three schemes in terms of system throughput:

- BlueBee [21]: BLE adopts the famous CTC design in [21] for BLE-to-ZigBee transmissions without ACK feedback and it repeatedly transmits each data 20 times to ensure one reliable reception.
- BlueBee+XBee: BLE adopts BlueBee for BLE-to-ZigBee transmissions while ZigBee adopts another famous XBee CTC design [22] for transmitting ZigBee MAC-layer ACK packet as feedback.
- BlueBee+GOP-ACK: BLE adopts BlueBee for BLE-to-ZigBee transmissions while ZigBee adopts our ZigBee-to-BLE design for ACK feedback.

In our simulation, for each simulation run, BLE sends 50,000 ZigBee packets, each consisting of a 32-bit preamble and a payload of 25 bytes (which is the typical payload length for ZigBee communications [1]). In each transmission, we set  $\Delta\hat{\phi}_{th} = 0.6$ ,  $\hat{N}_{ORP}^{th} = 10$ , and  $M = 3$  for BlueBee+GOP-ACK and assume a random sampling offset  $\Delta t$  that is uniformly distributed in  $[-0.5\mu s, 0.5\mu s]$ .

Fig. IX plots the system throughput of the original BlueBee scheme, BlueBee+GOP-ACK, and BlueBee+XBee, when SNR varies from  $0dB$  to  $20dB$ . From this Figure, we can see that the system throughput of each scheme increases as SNR increases. In addition, when  $SNR \geq 10dB$ , BlueBee+GOP-ACK and BlueBee+XBee outperform the original BlueBee scheme because a BLE sender in the former two schemes only performs necessary retransmissions and hence avoids bandwidth waste. BlueBee+GOP-ACK outperforms BlueBee+XBee by 46.67-52.48%, manifesting that GOP-ACK is more efficient and robust than XBee.

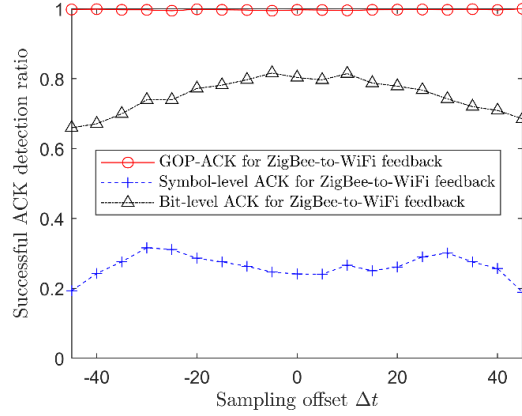


Fig. X Successful detection ratio of GOP-ACK, symbol-level ACK, and bit-level ACK for ZigBee-to-WiFi feedback under different sampling offsets and SNR=5dB.

Fig. X plots the successful detection ratios of GOP-ACK, symbol-level ACK, and bit-level ACK for ZigBee-to-WiFi feedback under various sampling offsets ( $\Delta t$  from  $-0.45\mu s$  to  $0.45\mu s$ ) when SNR = 5dB. The results reveal several key findings:

- Symbol-level ACK exhibits consistently lower detection ratios compared to both GOP-ACK and bit-level ACK, confirming its vulnerability under poor channel conditions as discussed in Section II.B.
- GOP-ACK achieves superior detection ratios compared to bit-level ACK across all sampling offsets, demonstrating its enhanced reliability over existing designs.
- Most notably, only GOP-ACK maintains a consistent successful detection ratio regardless of  $\Delta t$ , validating that its ORS-based design effectively mitigates sampling offset impacts.

## Appendix G. Biographies



**Shumin Yao** received his Ph.D. and MS degrees from Macau University of Science and Technology, Macau, China, in 2022 and 2018, respectively, and his BS degree from Beijing Institute of Technology, Zhuhai, China, in 2016. He is currently a Postdoctoral researcher in Pengcheng Laboratory, Shenzhen, China. His interests include semantic communication, cross-technology communication, wireless LAN, and Internet of Things.



**Qinglin Zhao** received his Ph.D. degree from the Institute of Computing Technology, the Chinese Academy of Sciences, Beijing, China, in 2005. From May 2005 to August 2009, he worked as a postdoctoral researcher at the Chinese University of Hong Kong and the Hong Kong University of Science and Technology. Since September 2009, he has been with the School of Computer Science and Engineering at Macau University of Science and Technology and now he is a professor. He serves as an associate editor of IEEE Transactions on Mobile Computing and IET Communications. His research interests include blockchain and decentralization computing, machine learning and its applications, Internet of Things, wireless communications and networking, cloud/fog computing, software-defined wireless networking.



**MengChu Zhou** (Fellow, IEEE) received his Ph. D. degree from Rensselaer Polytechnic Institute, Troy, NY in 1990 and then joined New Jersey Institute of Technology where he is now a Distinguished Professor. His interests are in Petri nets, automation, robotics, big data, Internet of Things, cloud/edge computing, and AI. He has 1100+ publications including 14 books, 750+ journal papers (600+ in IEEE transactions), 31 patents and 32 book-chapters. He is Fellow of IFAC, AAAS, CAA and NAI.



**Li Feng** received her MS degree in operation research from the Department of Mathematics, University of Hong Kong, Hong Kong, in 2007, and her Ph.D. degree in electronic information technology from the School of Computer Science and Engineering (SCSE), Macau University of Science and Technology (MUST), Macao, China, in 2013. She is currently an Associate Professor in SCSE, MUST. Her current research interests include Internet of Things, wireless and mobile networks, power saving, software defined networking, and performance analysis.



**Peiyun Zhang** (M'16–SM'17) received her Ph.D. in computer science from the School of Computer Science and Technology at the Nanjing University of Science and Technology, Nanjing, China in 2008. She is currently a professor at the Engineering Research Center of Digital Forensics of Ministry of Education, and the School of Computer Science, Nanjing University of Information Science & Technology. Her interests include blockchains, and cloud/ trust computing. She has published over 70 papers in these fields.



**Aiiad Albeshri** Received M.S. and Ph.D. degrees in Information Technology from Queensland University of Technology, Brisbane, Australia in 2007 and 2013 respectively. He has been an associate professor at Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia since 2018. His current research focuses on Information Security, Trust in Cloud computing, Big Data and HPC.



## Reference

- [1] 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.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] 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.
- [10] W. Jeong *et al.*, "SDR receiver using commodity wifi via physical-layer signal reconstruction," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, London United Kingdom, Sep. 2020, pp. 1–14. doi: 10.1145/3372224.3419189.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston: McGraw-Hill, 2008.
- [15] S. M. Ross, *Introduction to probability models*, 10th ed. Amsterdam ; Boston: Academic Press, 2010.
- [16] "R2020b - updates to the MATLAB and simulink product families." [https://www.mathworks.com/products/new\\_products/release2020b.html](https://www.mathworks.com/products/new_products/release2020b.html) (accessed Jun. 13, 2024).
- [17] "Modulation using OQPSK method - MATLAB - MathWorks." <https://www.mathworks.com/help/comm/ref/comm.oqpskmodulator-system-object.html> (accessed Jun. 20, 2024).
- [18] "Modulate using GMSK method - MATLAB - MathWorks." <https://www.mathworks.com/help/comm/ref/comm.gmskmodulator-system-object.html> (accessed Jun. 20, 2024).
- [19] "Generate white gaussian noise samples - MATLAB wgn - MathWorks." <https://www.mathworks.com/help/comm/ref/awgn.html> (accessed Jun. 21, 2024).
- [20] "WLAN Toolbox." <https://www.mathworks.com/products/wlan.html> (accessed Jan. 27, 2023).
- [21] 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.
- [22] 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.