Supplementary Material for GOP-ACK

Shumin Yao, Qinglin Zhao, *Senior Member*, *IEEE*, MengChu Zhou, *Fellow*, *IEEE*, Li Feng, and Peiyun Zhang, *Senior Member*, *IEEE*, Aiiad Albeshri
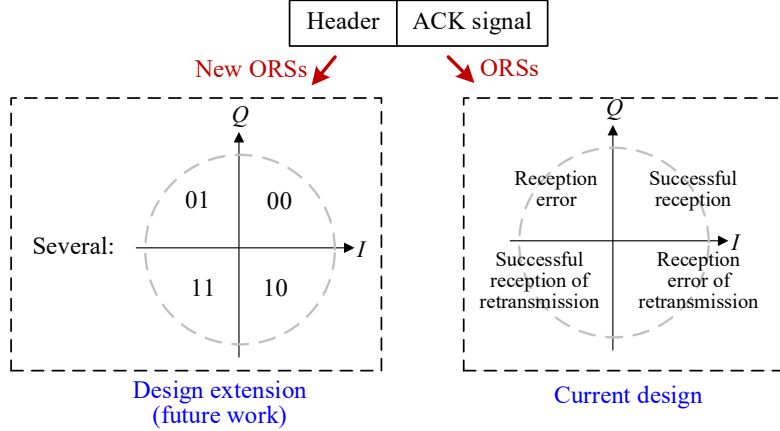
Appendix A. Design extension

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.

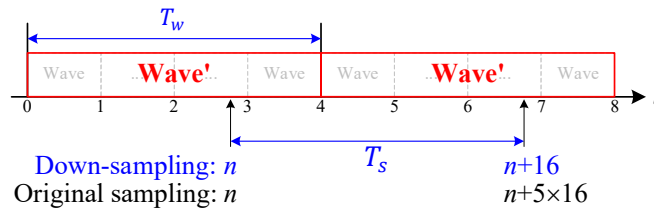## Appendix B. Case study of ZigBee-to-WiFi feedback

Fig. II. 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.

In this section, we follow the guidelines of our framework presented in Section III.B to design ZigBee-to-WiFi feedback. Recall that our approach requires

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

In ZigBee-to-BLE feedback shown in Section IV in the main file, to meet the requirement (1), ZigBee creates a wave of $T_w = 1\mu s$, while BLE adapts a sampling interval of $T_s = 1\mu s$ to calculate one phase shift. 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 $T_w \neq T_s$, thereby violating (1).

To meet (1) for ZigBee-to-WiFi feedback, we only need to make the following modifications in comparison with ZigBee-to-BLE feedback:

ZigBee side modification. 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. II. To achieve this, as shown in Fig. III, ZigBee sets one ZigBee-to-WiFi ACK bit sequence to $\boldsymbol{B'} = [\boldsymbol{B}, \boldsymbol{B}, \boldsymbol{B}, \boldsymbol{B}]$ in the TX firmware of ZigBee-to-BLE feedback, where $\boldsymbol{B}$ is one ZigBee-to- BLE ACK bit sequence.

WiFi side modification. WiFi performs down-sampling with $\frac{1}{5}$ of the original sampling rate, as shown in Fig. II. In this way, the sampling interval of calculating one phase shift is $T_s = 5 \times 0.8 = 4\mu s$, which ensures $T_w = T_s$ and hence meets (1). To achieve this, as shown in Fig. III, WiFi inserts one down sampler after its built-in sampler in its RX firmware, where the former reserves the 1st and 6-th sampling values and drops the others when receiving 6 sampling values from the latter. Note that since WiFi samples each wave' 6 times even with down-sampling, it detects an ACK signal successfully if the following two conditions are met:

- *C1: Condition of a successful busy channel detection.* Let $\lambda = \frac{\sum_{m=1}^{6(M+1)}|C_{6m}|^2}{6(M+1)}$ denote the power of $\{C_m\}_{m=1}^{6(M+1)}$. C1 is satisfied if λ is greater than a threshold $\hat{\lambda}$, namely,
$$\lambda > \hat{\lambda}. \tag{2}$$
- *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,
$$N_{ORS} > \hat{N}_{ORS}. \tag{3}$$
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, however, $\Delta\varphi$ is not always 0. Hence, in our design, a decoder detects an ORS m if there are more than N_φ of its corresponding phase values, $\Delta\varphi_m$, are smaller than a threshold $\Delta\hat{\varphi}$, namely,
$$\sum_{i=6M}^{6M+6} [\![\Delta\varphi_i < \Delta\hat{\varphi}]\!] > N_\varphi, \tag{4}$$
where $[\![s]\!] = 1$ if the statement s is true and 0 otherwise.

The ACK decoding in ZigBee-to-WiFi feedback is the same as that in ZigBee-to-BLE feedback. More specifically, WiFi adopts Algorithm 1 in Section IV in the main file for ACK decoding, where $\boldsymbol{C} = \boldsymbol{C'}$ and $\Delta\boldsymbol{\varphi} = \Delta\boldsymbol{\varphi'}$.
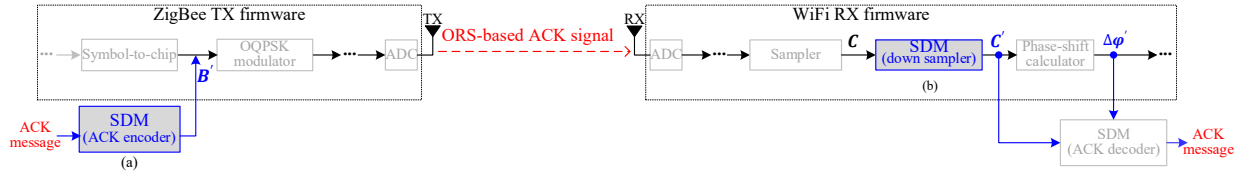

Fig. III. For ZigBee-to-WiFi feedback, ZigBee sets its ACK bit sequence to $\boldsymbol{B'}$ on the TX firmware of ZigBee-to-BLE feedback, and WiFi inserts one down sampler after its built-in sampler in its RX chain and adds $\boldsymbol{C}$, $\boldsymbol{C'}$, and $\Delta\varphi'$ as in the RX firmware of ZigBee-to-BLE feedback.
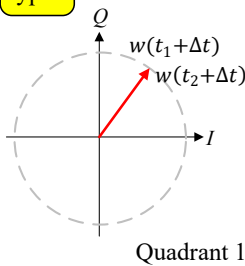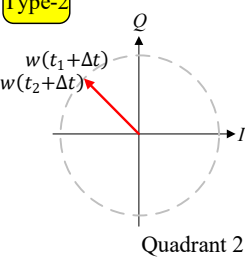
Appendix C. Supplementary of Tables
Below, we show the tables that helps better understanding the main file of the paper. More specifically, TABLE A lists the the abbreviations used in the paper, TABLE B shows the details of TABLE 2 in the main file, and TABLE C summarizes the main notations of our model.

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: successful reception of retransmission
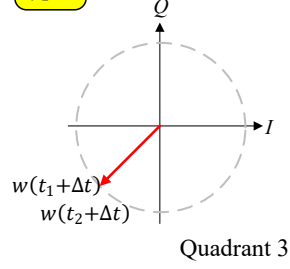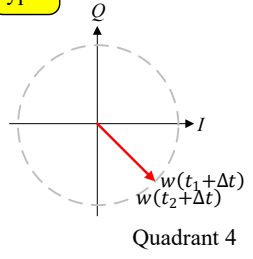
	0,0	Negative	**Type-3** Quadrant 3	
1,0,1,0,1	1,1,1	Positive	**Type-4** Quadrant 4	Type-4: reception error of retransmission
	0,0	Negative		

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.
λ	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.
σ^2	The noise power.
τ	The sampling interval.
ΔT	The sampling offset.
Δt	The value of ΔT.
P_{ACK}^s	The successful ACK decoding probability.
$P_{C1}(\Delta t)$	The successful probability of detecting busy channel under sampling offset Δt.
$P_{C2}(\Delta t)$	The probability of $N_{ORS} > \hat{N}_{ORS}^{th}$ under sampling offset Δt.
$P_{C3}(\Delta t)$	The successful probability of decoding an ACK type under sampling offset Δt.

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 Δt.
Ω	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 firmware-nonmodifying method

In this section, we provide more details about the firmware-nonmodifying method, which reuses all modules in firmware. Before this, we first specify the TX and RX firmware.
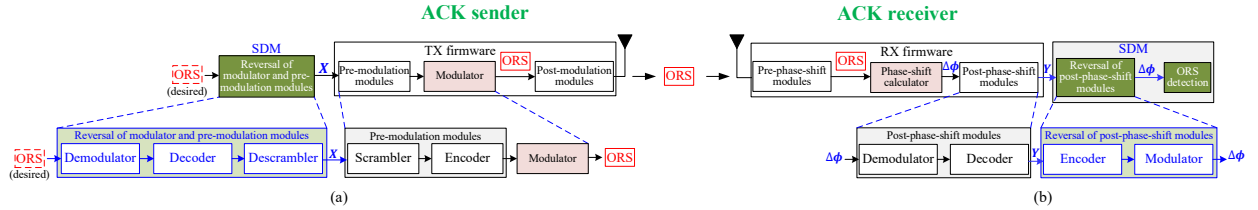

Fig. IV. Firmware-nonmodifying method: (a) reversal of modulator and pre-modulation modules; (b) reversal of post-phase-shift modules.

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. IV. 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]–[8], we may create an SDM that reverses the pre-modulation and modulation operations in the TX firmware. For example, as shown in Fig. IV(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 [9]–[12], etc., we employ another SDM, which reverses the operation of the post-phase-shift modules, to process 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. IV(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 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 $N_{ORS}^1, \dots, N_{ORS}^{\mathcal{T}}$ under Δt, $P_{\Delta t}(n_1, \dots, n_{\mathcal{T}})$.

E.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. Then, the average power of these M' sampling instances, λ, is calculated as

$$\lambda = \frac{\sum_{m=1}^{M'}|C_m|^2}{M'}. \tag{5}$$

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

$$\begin{aligned} P_{C1}(\Delta t) &= P\left(\lambda > \hat{\lambda}\right) \\ &= P\left(\frac{\sum_{m=1}^{M'}|C_m|^2}{M'} > \hat{\lambda}\right) \\ &= P\left(\sum_{m=1}^{M'}|C_m|^2 > M'\hat{\lambda}\right) \end{aligned} \tag{6}$$

$$= P\left(\sum_{m=1}^{M'}\left[(C_m^I)^2 + (C_m^Q)^2\right] > M'\hat{\lambda}\right).$$

Here, as specified in (7) in the main file, $C_1^I, C_1^Q, \ldots, C_{M'}^I$, and $C_{M'}^Q$ are $2M'$ Gaussian distributed random variables with the same variance $\sigma^2/2$ but different means, which are $x_I(\tau + \Delta t), x_Q(\tau + \Delta t), \ldots, x_I(M'\tau + \Delta t), x_Q(M'\tau + \Delta t)$, respectively. Furthermore, they are mutually independent since they correspond to different components of different waves. Then, according to [13], $\sum_{m=1}^{M'}\left[(C_m^I)^2 + (C_m^Q)^2\right]$ is a non-central chi-squared random variable with $2M'$ degrees of freedom and a non-centrality parameter $\sum_{m=1}^{M'}\left[x_I^2(m\tau+\Delta t)+x_Q^2(m\tau+\Delta t)\right]$. Then we can calculate have:

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

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.

E.2 Calculation of $P_{\Delta t}(n_1, \ldots, n_{\mathcal{T}})$

Note that a type-j^*, $j^* = 1, \ldots, \mathcal{T}$, ACK signal consists of M type-j^*ORSs. When receiving such a signal, a decoder detects the type of each ORS independently. Under a sampling offset Δt, for each detection operation, the decoder might identify type j^* as type j with probability $p_{j^*}^j(\Delta t), j = 1, \ldots, \mathcal{T}$. Recall that N_{ORS}^j is the number of detected type-j ORSs. Then, $\left\{N_{ORS}^j\right\}_{j=1}^{\mathcal{T}}$ follows a multinomial distribution [14] with a probability mass function $P_{\Delta t}(n_1, \ldots, n_{\mathcal{T}}) = P_{\Delta t}\left(N_{ORS}^1 = n_1, \ldots, N_{ORS}^{\mathcal{T}} = n_{\mathcal{T}}\right.$ under Δt):

$$P_{\Delta t}(n_1, \ldots, n_{\mathcal{T}}) = \begin{cases} \dfrac{M!}{\prod_{j=1}^{\mathcal{T}} n_j!}\prod_{j=1}^{\mathcal{T}}\left(p_{j^*}^j(\Delta t)\right)^{n_j}, & \text{when } \sum_j n_j = M, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Below, we explain how to calculate $p_{j^*}^j(\Delta t)$ in (8). In our design, a decoder samples each of the M type-j^*ORSs twice. Let $C = C^I + jC^Q$ denote the first sampling instance of one received type-j^*ORS. Let A_j be the area of quandrant j of the constellation diagram. In the case study of ZigBee-to-BLE feedback, a type-j^* ORS is inferred as a type-j ORS if its C appears in A_j. Hence,

$$\begin{aligned} p_{j^*}^j(\Delta t) &= \text{Prob}\left(C \in A_j\right) \\ &= \iint_{(C^I, C^Q)\in A_j} f(i,q)didq, \end{aligned} \tag{9}$$

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

$$f(i,q) = \frac{1}{2\pi\sigma^2}e^{\frac{-\left[\left(i-x_I(t+\Delta t)\right)^2 + \left(q-x_Q(t+\Delta t)\right)^2\right]}{2\sigma^2}}. \tag{10}$$

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 [15]. It consists of ZigBee, BLE, AWGN channel, and WiFi modules. They are based on the OQPSK modulator module

[16], the Gaussian minimum shift keying modulator module [17], the white Gaussian noise generator module [18], and WLAN toolbox [19], 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 Δt and noise power σ^2. The parameter settings are the same as in Section VI.B in the main file.
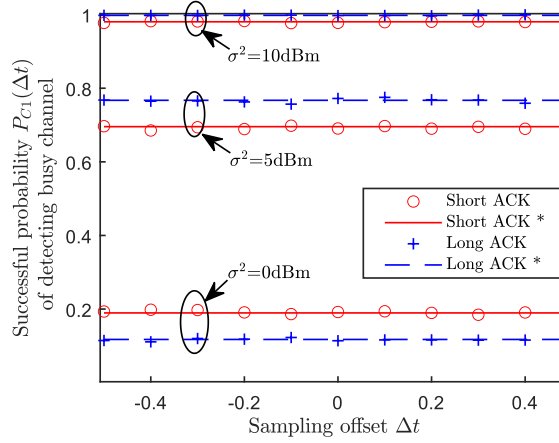

Fig. V. $P_{C1}(\Delta t)$ vs. different Δt and σ^2.

Fig. V plots $P_{C1}(\Delta t)$ vs. Δt and σ^2, when $\Delta t = -0.5, \dots, 0.5 \mu s$ and $\sigma^2 = 0,5,10$dBm. From this figure, we have the following observations:

- Given σ^2, $P_{C1}(\Delta t)$ does not vary with Δt, manifesting that when detecting a busy channel, GOP-ACK can well resist sampling offsets that are inherent in CTC.
- Given Δt, $P_{C1}(\Delta t)$ decreases as σ^2. The reason is that when σ^2 is lower, the power λ of sampling instances is often lower than $\hat{\lambda}$, leading to more frequent miss detections. These results imply that we should set a smaller $\hat{\lambda}$ when σ^2 is low.
- Given Δt, $P_{C1}(\Delta t)$ for short signals is higher (lower) than that for long signals when σ^2 is low (high). This implies that for maximizing the busy channel detection performance solely, ZigBee should adopt a short (long) ACK signal when σ^2 is low (high).
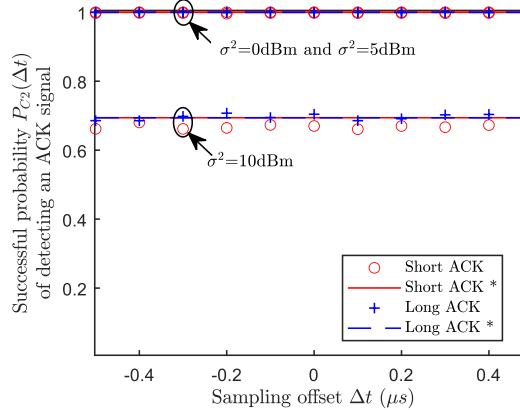

Fig. VI. $P_{C2}(\Delta t)$ vs. different Δt and σ^2.

Fig. VI shows $P_{C2}(\Delta t)$ vs. Δt and σ^2, when $\Delta t = -0.5, \ldots, 0.5\mu s$ and $\sigma^2 = 0, 5, 10$dBm. From it, we conclude that:

- Given σ^2, $P_{C2}(\Delta t)$ does not vary with Δt, manifesting that when detecting ACK signals, GOP-ACK can well resist sampling offsets that are inherent in CTC.
- Given Δt, $P_{C2}(\Delta t)$ is high except that when σ^2 is as high as 15dBm, which indicates that only strong noise negatively impacts the successful probability of ORS detection.

Given Δt and σ^2, $P_{C2}(\Delta t)$ is the same, whether short or long signals, implying that our overlapping-ORS approach for short signals does not affect the signal detection performance.
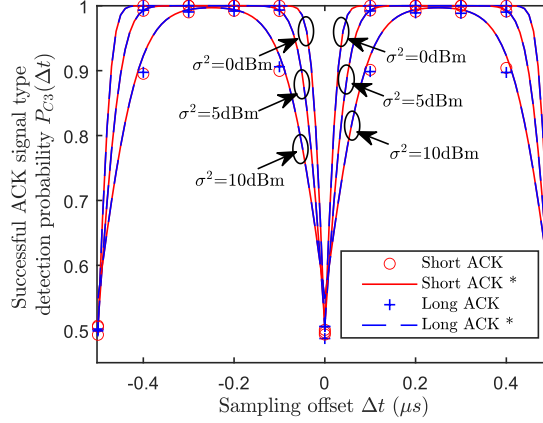

Fig. VII. $P_{C3}(\Delta t)$ vs. different σ^2 and Δt.

Fig. VII shows $P_{C3}(\Delta t)$ vs. Δt and σ^2, when $\sigma^2 = 0, 5, 10$dBm and $\Delta t = -0.5, \ldots, 0.5\mu s$. From it, we have:

- Given Δt, $P_{C3}(\Delta t)$ decreases as σ^2 increases, indicating the negative impact of noises on ACK decoding performance.
- Given Δt and σ^2, $P_{C3}(\Delta t)$ is the same, whether short or long signals, implying that our overlapping-ORS approach for short signals does not affect the performance of ACK message decoding.
- Given σ^2, $P_{C3}(\Delta t)$ decrease as $|\Delta t|$ is close to 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.

Lastly, in Figs. IV-VI, the close match between the theoretical and simulation curves manifests that our performance model is very accurate.
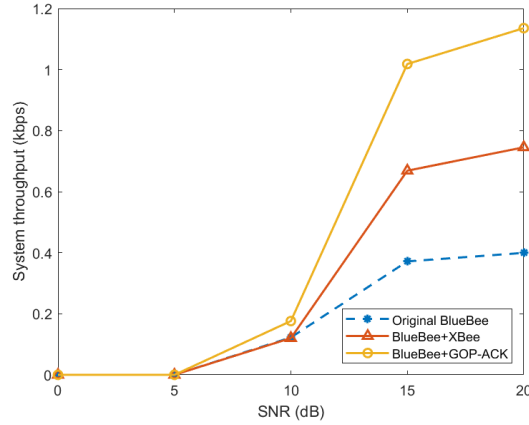
F.2 Comparison with the state of the art

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

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

- BlueBee [20]: BLE adopts the famous CTC design in [20] 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 [21] 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 $\Delta\hat{\varphi}_{th} = 0.6$, $\widehat{N}_{ORP}^{th} = 10$, and $M = 16$ for BlueBee+GOP-ACK and assume a random sampling offset Δt that is uniformly distributed in $[-0.5\mu s, 0.5\mu s]$.

Fig. VIII plots the system throughput of the original BlueBee scheme, BlueBee+GOP-ACK, and BlueBee+XBee, when SNR varies from $0 dB$ to $20 dB$. From this Figure, we can see that the system throughput of each scheme increases as SNR increases. In addition, when $SNR \geq 10 dB$, 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.

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 Peng Cheng 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.
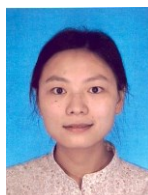
Li Feng received her MS degree in operation research from the Department of Mathematics, University of Hong Kong, Hong Kong, in 2007, and her Ph.D. degree in electronic information technology from the School of Computer Science and Engineering (SCSE), Macau University of Science and Technology (MUST), 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]   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.

[3]   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.

[4]   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.

[5]   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.

[6]   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.

[7]   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.

[8]   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.

[9]   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.

[10]   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.

[11]   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.

[12]   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.

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

[14]   S. M. Ross, *Introduction to probability models*, 10th ed. Amsterdam ; Boston: Academic Press, 2010.

[15]   "R2020b - Updates to the MATLAB and Simulink product families." https://www.mathworks.com/products/new_products/release2020b.html (accessed Jun. 13, 2022).

[16]   "Modulation using OQPSK method - MATLAB - MathWorks." https://www.mathworks.com/help/comm/ref/comm.oqpskmodulator-system-object.html (accessed Dec. 30, 2022).

[17]   "Modulate using GMSK method - MATLAB - MathWorks." https://www.mathworks.com/help/comm/ref/comm.gmskmodulator-system-object.html (accessed Dec. 31, 2022).

[18]   "Generate white Gaussian noise samples - MATLAB wgn - MathWorks." https://www.mathworks.com/help/comm/ref/awgn.html (accessed Dec. 30, 2022).

[19]   "WLAN Toolbox." https://www.mathworks.com/products/wlan.html (accessed Jan. 27, 2023).

[20]   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.

[21]   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.