Protocol-Aware Backscatter Communication Using Commodity Radios

Longzhi Yuan[1], Rongrong Zhang[2], Kai Yang[3], Jianping An[3], Si Chen[4], Wei Gong[1*]
[1]University of Science and Technology of China, Hefei, China
[2]Capital Normal University, Beijing, China
[3]Beijing Institute of Technology, Beijing, China
[4]Simon Fraser University, Canada
[*]Corresponding: weigong@ustc.edu.cn

***Abstract*—Backscatter is one of the important techniques of IoTs as it can offer low-cost and low-energy wireless communication. With the help of widely available ambient signals, backscatter communication can even work without specialized carrier generation. The current solutions, however, are unable to identify various ambient signals. So, we introduce a backscatter system to do this. In particular, we realize the identification of OFDM WiFi (802.11a/g/n), 802.11b WiFi, Bluetooth Low Energy, and ZigBee. Further, we implement our backscatter in the FPGA hardware to evaluate the design. Comprehensive field studies show that our backscatter can identify four protocols at an average identification accuracy of about 90%. We also demonstrate that our identification is compatible with different backscatter modulation for all four signals in 2.4GHz band.**

I. Introduction

Different from traditional active-radio communication systems, backscatter communication works in a passive way. As there is no carrier generation module on the tag, it uses incoming signals from readers and modulates those signals, leading to ultra low power communication [1] [2]. Recently, a number of backscatter systems with commodity radios as readers have been proposed [3] [4] [5] [6] [7], aiming to get rid of dedicated and expensive readers [5] [8]. Among all commodity radios, WiFi, Bluetooth Low Energy (BLE), and ZigBee are most widely used and studied. Using such pervasive radios as readers enables ubiquitous backscatter deployment since nowadays our personal electronic devices have at least one of those commodity radios, e.g., BLE with smartwatch, WiFi and BLE with smartphones, ZigBee with SmartBulbs.

Existing backscatter systems, however, can only support a fixed signal as carrier, which leads to limited application scenarios. For example, in [3] [9], the tag uses only 802.11b signal; in [9] tag uses only OFDM WiFi signal; in [6] tag uses only BLE signal. The first step towards using multiple ambient signals as the carrier is identifying them so that corresponding modulation schemes can be taken. To address the problem that existing backscatter systems have no effective ways to identify the protocol of excitation, we aim to identify signals of 802.11b/n, BLE, and ZigBee so that it can adapt to corresponding protocol and choose proper backscatter modulation schemes.

[*]Corresponding author: Wei Gong.

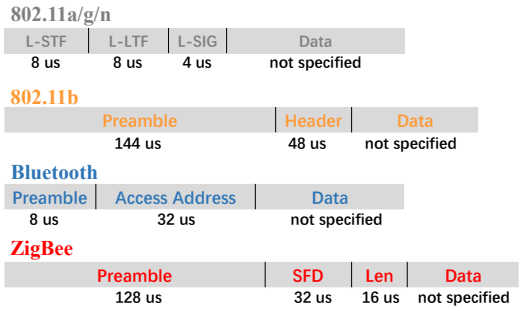


Fig. 1. Packet formats of 802.11n WiFi, 802.11b WiFi, Bluetooth, and ZigBee. 802.11n has a preamble of $20\mu s$(L-STF $8\mu s$ + L-LTF $8\mu s$ + L-SIG $4\mu s$), 802.11b has a preamble of $144\mu s$, Bluetooth has a preamble of $8\mu s$, and ZigBee has a preamble of $128\mu s$.

To realize such a backscatter system, we encounter two challenges. First, packets of those protocols have different physical characteristics, such as packet formats, data modulation formats, and bandwidths. The packet formats of those protocols are shown in Fig.1. Preambles, the first part of packets, are used for packet detection, and they are also our focus for protocol-identification. The preamble of OFDM WiFi is composed of three parts: L-STF, L-LTF, and L-SIG, and the total duration is $20\mu s$. The durations of 802.11b, Bluetooth, and ZigBee are $144\mu s$, $8\mu s$, and $128\mu s$, respectively. The typical bandwidths are 20MHz, 11MHz, 1MHz, and 2MHz, respectively. If we design a separate identification sub-system for each protocol, the complexity would be too high for a low-power tag. Therefore, we need to design a unified identification scheme to differentiate those protocols.

Second, computing resource is limited for the backscatter system. For example, microprocessor and low-power FPGA widely used in backscatter systems can support no more than hundreds of multipliers. This means the current identification schemes for active radios are not suitable and a ultra-low-power solution for protocol identification is much-needed.

To meet the above requirements, the main contributions of this paper are:

- We design a unified protocol identification method for four popular protocols in 2.4GHz band. It is lightweight and can be implemented with an ultra-low-power FPGA.

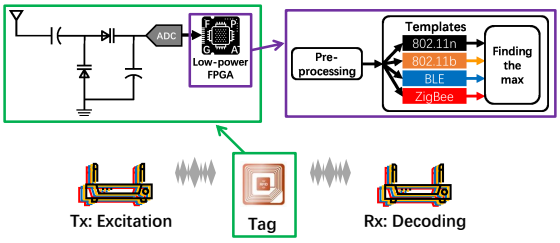


Fig. 2. Overview of our backscatter framework. The tag first identifies the protocol of the excitation signal. Then it modulates sensor data onto the carrier and backscatters to the receiver. Our backscatter rag is composed of an envelope detector, and ADC, and a low-power FPGA. The main job of the FPGA is baseband processing, including pre-processing, cross-correlation, and modulation.

- We build a real backscatter system using customized tags and commodity radios. Extensive experiments have been conducted and results show that our tag effectively realizes protocol-identification at average accuracy of 90% and supports backscatter communication with all four different ambient signals.

II. Protocol-Identification

The system overview is shown in Fig.2. The excitation source, which could be WiFi, BLE, or ZigBee device, provides RF signal for the tag. The tag receives the excitation signal and then the envelope detector and ADC convert it to baseband signal and provide it to the low-power FPGA. The FPGA removes the DC component in the digital signal(pre-processing) as the DC component contains little information and contributes little to the identification of protocols. Then the FPGA gets similarities of the DC-free signal with four templates (one for a protocol), and identifies the excitation to the protocol whose template gets the maximum similarity. Then the tag modulates its data on excitation based on the protocol by controlling the toggling of an RF-switch which is not related to the identification and not shown in the figure. Finally, the tag data is decoded at the receiver. Notice that if the protocol is misidentified, the tag will take a modulation scheme not suitable for the excitation signal, causing serious errors in decoded tag data at the receiver.

A. Data sampling and pre-processing

Before considering the method to realize protocol-identification we have to consider how to get baseband signal for our FPGA. The envelope detector is used in many backscatter systems because of its simple structure and low energy consumption [3] [6]. It plays the role of down-converting the RF signal to the baseband signal. The envelope detector outputs the absolute value of the baseband signal contained in the excitation. In the consideration of power and system complexity, we choose to use the envelope detector in our system.

Notice that the output of the envelope detector is an analog signal, which cannot be accepted by FPGA. So we use an ADC to convert the analog signal to a digital one.

Packets of different protocols take different modulation formats: 802.11n WiFi takes orthogonal frequency division multiplexing (OFDM), BLE takes Gaussian frequency shifting keying (GFSK), both 802.11b WiFi and ZigBee take phase shifting keying (PSK). Ideally, GFSK and PSK are constant-envelope modulations and the envelope wouldn't change at all, which means that envelope-based protocol identification will fail. But with the limitation of bandwidth and the need for reducing inter-symbol interference (ISI), pulse shaping filters are used: root raised-cosine filter in 802.11b, Gaussian filter in BLE, half-sine filter for ZigBee [10] [11]. With these pulse shaping filters, the envelope signals of those protocols are no longer constant.

The envelope signals of those protocols have different patterns, based on which protocol-identification is realized. As they are the absolute values of baseband signals, there are strong DC components. For example, we test with 802.11n and 802.11b signals and find that the DC components cover 47% and 77% signal power, respectively. For BLE and ZigBee the portions are both above 95%. Such DC components are useless for protocol-identification, so we need to remove them in the pre-processing phase.

We use a sliding window to realize DC removal. Let p be the current sampling index, $V_d(p), V_d(p-1), ..., V_d(p-w+1)$ be the samples of ADC in the current window where w means the number of samples the window covers (defined as processing window size). Then the mean value $\mu(p)$, and the new sample $\widehat{V}_d(p)$ are calculated as follows.

$$\begin{aligned} \mu(p) &= \frac{\sum_0^{i=w-1} V_d(p-i)}{w} \\ \widehat{V}_d(p) &= V_d(p) - \mu(p) \end{aligned} \tag{1}$$

When a new sample comes the sampling index becomes $p+1$. The processing window moves a step forward so that it covers $V_d(p+1), V_d(p), ..., V_d(p-w+2)$, then we can get $\widehat{V}_d(p+1)$ in the same way as(1).

B. Protocol-identification

We use cross-correlation for protocol-identification. Formally, at time index k, we have prior known template sequence $t(i), i = 1, 2, ..., N-1$, and N newest samples $\widehat{V}_d(i), i = 1, 2, ..., N-1$. We can use cross-correlation as (2) to measure the similarity of the template and sample sequence. When the template matches with the sample sequence, the similarity S gets a peak.

$$S(k) = \frac{\sum_{i=0}^{N-1} \widehat{V}_d(i) t(i)}{\sqrt{\sum_{i=0}^{N-1} \widehat{V}_d(i)^2 \sum_{i=0}^{N-1} t(i)^2}} \tag{2}$$

We aim to identify packets of 802.11b/n, BLE, and ZigBee. Naturally, we can design a template for every protocol, and then we can get a maximum value for every protocol. Among those maximum values, we can find the maximum one and identify the packet to be corresponding protocol.

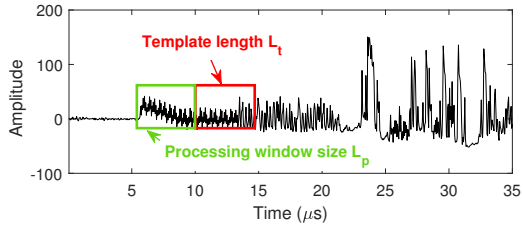
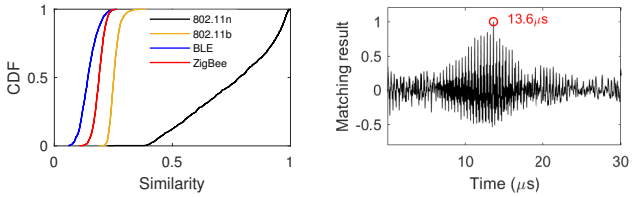


Fig. 3. The influence of DC removal. The beginning part of packets will be distorted after DC removal, which is harmful for protocol-identification. So that we ought to choose our templates out of *processing window*.



(a) Matching similarity of 802.11n packets with four templates .

(b) Matching result of 802.11n packets with template of 802.11n.

Fig. 4. Cross-correlation result of 802.11n packets with four templates. In 4(a) we can see the similarity (maximum matching result) of 802.11n packets with four templates, the similarity with 802.11n is mostly above those with templates of other protocols. In (b) we can see the peak appears at $13.6\mu s$.

We now have to consider how to get proper templates. Preambles of those protocols have fixed content and are used for the detection of packets in traditional communication systems. The preamble of BLE has a duration of only $8\mu s$, which is the shortest in WiFi, BLE, and ZigBee. So we choose our target sequences from the first $8\mu s$ in envelope signals for all those protocols.

However, the usage of the sliding window for DC removal may distort envelope signals, the example of 802.11n is shown in Fig.3. We can see that the length of the distorted part is equal to that of the processing window size (defined as L_p). Let L_t be template length, then the total length of the processing window and template window ought to be less than that of $8\mu s$-preamble, the field from which the target sequences are selected (we define the field as *matching window*, and denote its length as L_m). Then we have $L_p + L_t \leq L_m$. We set L_p, L_t to be the lengths corresponding to the durations of $2\mu s$ and $6\mu s$ at different sampling rates, respectively. For example, at the sampling rate of 20Msps, $L_p = 2\mu s \times 20Msps = 40$, and $L_t = 6\mu s \times 20Msps = 120$. We know $L_m = 8\mu s \times 20Msps = 160$, in this way equation $L_p + L_t \leq L_m$ is satisfied. At other sampling rates that equation will also be satisfied.

Notice that peaks of similarity appear at the time when our templates match with envelope signals. Our target sequences for those protocols are set to be the head parts with the same durations of about $8\mu s$ of corresponding packets. We can have a coarse estimation about the start of the packet by monitoring the energy of sampled data. So we focus the searching range for similarity peaks within the $8\mu s$ field after high energy is detected.

TABLE I
RESOURCE CONSUMPTION FOR CROSS-CORRELATION

Protocols	Multipliers	Adders	LUTs
802.11n	120	119	13671
802.11b	120	119	13671
BLE	120	119	13671
ZigBee	120	119	13671
Total	480	476	54684
XC3S500E			9312
XC7A35T			20800
Quantification Scheme			1233

We firstly sample envelope signals of those protocols and process them in the method we introduced above, and analyse the protocol-identification performance of our method in PC. The result shows that the average identification accuracy is above 99% at 20Msps. When 802.11n packets act as the excitation signal, the distribution of maximum similarities with four templates are shown in Fig.4(a). The similarity of 802.11n packets with the template of 802.11n is mostly over 0.5 while with other templates similarities are lower than 0.4, so that we can easily identify those packets to 802.11n. A single case of the result of 802.11n packet cross-correlating with the template of 802.11n is shown in Fig.4(b). We get a peak of cross-correlation as high as 0.95 at $13.6\mu s$. And this packet starts at about $5.5\mu s$. The time interval between the start and the peak is $13.6\mu s - 5.5\mu s = 8.1\mu s$, close to $8\mu s$.

There are also lower peaks that appear periodically, thats because our target sequence for 802.11n is part of the L-STF, which is made up of 10 identical parts. As a result, when the 802.11n packet partly matches its template there will also be peaks in cross-correlation result.

C. Low Resource Matching

Protocol-identification can be realized with the introduced method, but too much computing resource is needed. We aim to identify packets of four protocols, so we have to get a similarity value as (2) for every protocol when a new sample comes. The output of ADC has 12 bits in which only the lowest 9 bits are used, we need *9*9* multipliers and *9*9* adders to realize the mentioned method.

A low-power FPGA is used in our backscatter, we now estimate the resource consumption based on FPGA. For Xilinx FPGA the adders and multipliers can be made up by Lookup Tables (LUTs). LUT is an important resource in FPGA, a $9bits \times 9bits$ multiplier takes 105 LUTs and a *9bits + 9bits* adder takes 9 LUTs.

We need to calculate the similarities of envelope signal with four templates of length 120. We consider only the calculation of numerator in (2), and the other parts in that fraction for normalization are not counted. Then 480 multipliers and 476 adders are needed. The corresponding LUT resource and the supply capacities of two widely used FPGA chips Artix7(XC7A35T) and Spartan3(XC3S500E) are shown in Table I.

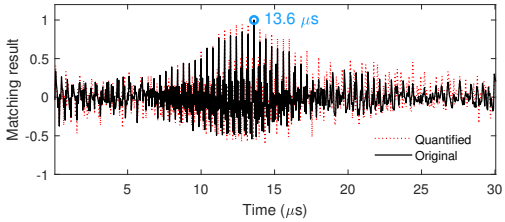


Fig. 5. Cross-correlation results of original and quantified 802.11n envelope signals with their templates at 20Msps. After rhe envelope signal and template are quantified to ± 1 there is still a peak at 13.6μs.

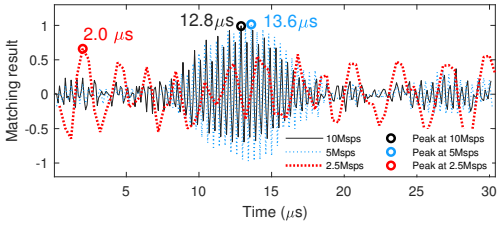


Fig. 6. Cross-correlation results of quantified 802.11n with its templates at 10Msps, 5Msps, and 2.5Msps. At 10Msps and 5Msps the peaks appear near 13μs, but at 2.5Msps the peak appears at 2μs

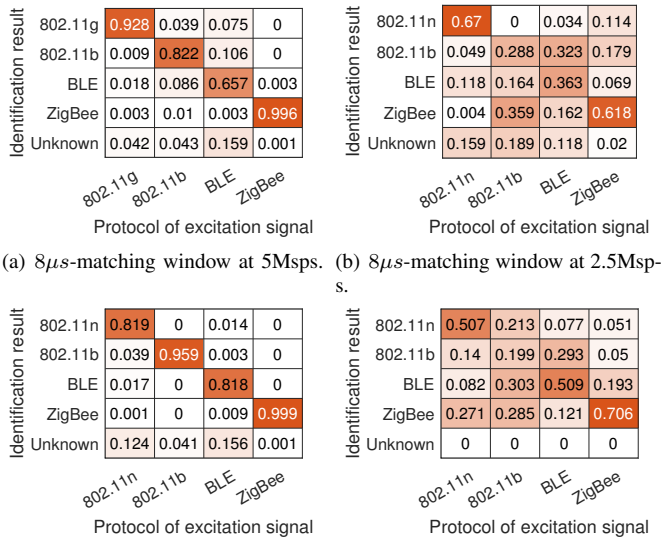
We can see that the listed FPGAs are even unable to supply LUTs needed by the calculation of similarities. In fact, besides that, we also need LUTs for pre-processing data and for controlling. As the protocol-identification method introduced above is resource consuming, it should be simplified.

1) Quantification: A natural thought is reducing data width, for example, we may use samples of 3 bits instead of 9 bits then each multiplier consumes only 13 LUTs. In our system, we use only 1 bit in our 9 bits samples. Specifically, we use only the sign of DC-free envelope signal. Then we can get rid of resource-consuming $9bits \times 9bits$ multipliers. The corresponding LUT resource consumption is shown in Table I at the line *Quantification Scheme*. In total 1233 LUTs are used for protocol-identification, that can be supplied by listed FPGA. The total number of used LUTs in our prototype is 1528, including 221 used in pre-processing data and control. The cross-correlation result of an 802.11n packet and its template is shown in Fig.5. Even quantified to ± 1, the envelope signal and the template still match well, as the peaks at about 13.6μs of both situations are distinguishable.

2) Down sampling: For reducing template length, note that 20Msps is chosen based on the standard sampling rate of 802.11n WiFi. Detection, synchronization and even data demodulation OFDM WiFi at reduced sampling rate are realized in [12], showing that 20Msps is necessary for OFDM WiFi. For other protocols including 802.11b, BLE, and ZigBee, 20Msps is not necessary, either. So we can reduce the sampling rate to 10Msps, 5Msps or even lower and have a try.

We now investigate the protocol-identification performance at 10Msps, 5Msps, and 2.5Msps after quantification. At sampling rate lower than 20Msps, our target sequences are still chosen from the beginning 8μs in the packet. L_p and L_t are still the lengths corresponding to the durations of 2μs and 8μs, respectively. For example, at 10Msps $L_p = 10Msps \times 2\mu s = 20$, $L_t = 10Msps \times 6\mu s = 60$.

The cross-correlation results at 10Msps, 5Msps, and 2.5Msps are shown in Fig.6. We can see that their peaks at 10Msps and 5Msps but at slightly different times: 12.8μs and 13.6μs respectively. The difference is $13.6\mu s - 12.8\mu s = 0.8\mu s$, which is exactly the duration of one in ten identical parts of L-STF (the first field in OFDM WiFi packet for packet detection). The difference is caused by the peaks generated when the template and the packet partly matched. However, at 2.5Msps the peak appears at 2.0 μs, indicating that its not caused by our target sequence.



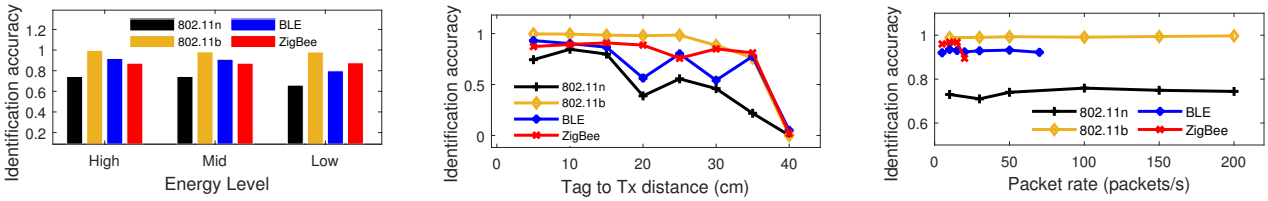
(a) 8μs-matching window at 5Msps. (b) 8μs-matching window at 2.5Msps.

(c) 40μs-matching window at 2.5Msps. (d) 40μs-matching window at 1Msps.

Fig. 7. Identification results with different sampling rates and different matching window sizes. At 2.5Msps, when the matching window is 8μs the average identification accuracy is 44%. When the window is 40μs the accuracy is 90%.

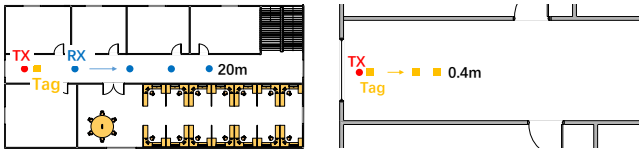
The identification results in Fig.7(a), and Fig.7(b) show that at 5Msps and 2.5Msps the average accuracies are about 85% and 44%, respectively. As expected the identification result at 2.5Msps is not good enough, but notice that 2.5Msps is not too low compared to the bandwidths of BLE (1MHz) and ZigBee (2MHz). While at 2.5Msps their accuracies are as low as 36.3% and 61.8% respectively. The bad performance at 2.5Msps may be caused by too short templates, four sequences whose lengths are all 15 (at 2.5Msps, $L_t = 2.5Msps \times 6\mu s = 15$).

We extend the matching window so that longer templates can be selected. BLE has the shortest preamble in those protocols, with a duration of only 8μs. If we extend the matching window so that it includes the field of *AccessAddress* (the second field in BLE after preamble) which has a duration of 32μs, then the matching window is 40μs. For other protocols, after extending the matching window we can get longer templates, too. After extending the matching window to 40μs the identification result at 2.5Msps is shown in Fig.7(c).



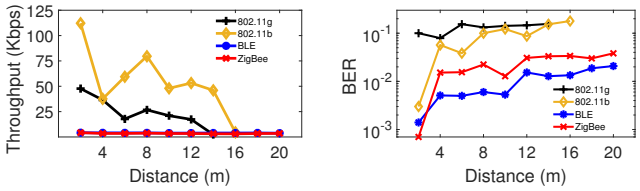
(a) Influence of energy level. (b) Influence of Tx-tag distance. (c) Influence of packet rate.

Fig. 8. The identification accuracies with different energy levels, Tx-tag distances, and packet rates. In (a) we set 3 different Tx power levels for every protocol. The **High** is 30dBm, 30dBm, 19dBm, and 19.5dBm for 802.11n, 802.11b, BLE, and ZigBee; the **Mid** is 27dBm, 27dBm, 15dBm, 14.5dBm for those protocols respectively; the **Low** is 24dBm, 24dBm, 12dBm, and 11.5dBm for those protocols respectively.



(a) Experimental area for end-to-end experiments. (b) Experimental area for testing protocol-identification accuracy.

Fig. 9. Experiment scenario: a hallway about 30m



(a) Throughput. (b) BER.

Fig. 10. Throughputs and BERs of our backscatter with radios of four protocols.

We can see that with the extended matching window the average identification accuracy gets about 90%. We have to notice that for 802.11a/g WiFi the preamble is 20μs, but 802.11n has an extra preamble (HT-preamble) including which the preamble is 40μs in total. That's to say our tag can support BLE packets with fixed Access Address, and it no longer supports 802.11a/g. However, even after extending the matching window, the average accuracy is still as low as 48% at 1Msps.

III. Performance Evaluation

A. Implementation and Experiment Setup

We implement our system in the Xilinx Artix XC7A35TFTG-1 FPGA platform. Besides FPGA, we also use an ADG902 RF-switch, an envelope detector made up of diodes, capacitors, and resistances, and a commercial AD9235 ADC whose sampling rate is set to 2.5Msps.

We use commodity devices as excitation providers and receivers. For WiFi, PCs with Qualcomm Atheros AR938X NICs are used as both transmitter and receiver. We set the transmission data rate of 802.11b to 1Mbps and 802.11n to 6.5Mbps. For BLE, we use the TI CC2540 as the transmitter and TI CC2650 as the receiver. Our experiment shows that the maximum packet rate is around 70 packets per second. For ZigBee, the TI CC2530 plays the role of transmitter and TI CC2650 as the receiver. TI CC2530 can transmit no more than 20 packets per second.

B. Protocol-Identification Accuracy

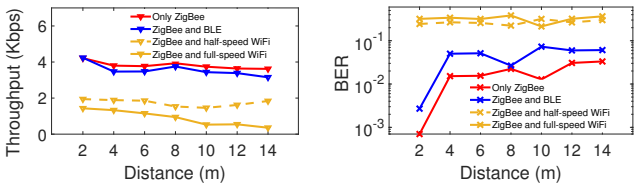
We evaluate the effectiveness of protocol-identification with different Tx energy levels, different Tx-tag distances, and different Tx packet rates in the scenario shown in Fig.8(a). We fix the tag at the position 0.15m away from the transmitter and measure the protocol-identification accuracies under different transmission power levels. The specific transmission powers are noted in Fig.8. The identification results with different power levels are shown in Fig.8(a). When the power level changes from **High** to **Low**, the accuracies of 802.11b and ZigBee decrease for less than 1%, while those of 802.11n and BLE drop about 8%. The average accuracy of four protocols changes from 89% to 83%. We can see that the identification accuracy of 802.11n is always the lowest in four protocols, and at the **Low** power state its around 70%. That may because the sampling rate of 2.5Msps is too low for 802.11n.

The influence of transmission packet rates is evaluated with the same experimental setup as above. The result is shown in Fig.8(c). The accuracies of 802.11n, 802.11b, and BLE fluctuate in a range smaller than 2%. That range of ZigBee is the biggest, which about 6%. We can see that identification accuracies change slightly with packet rates.

We set the power level to **High** state and fix the packet rates, then move the backscatter gradually away from the transmitter and measure the identification accuracies. As shown in Fig.8(b), when the distance increases from 0.05m to 0.3m the average accuracy decreases from about 89% to about 69%, while when the distance increases from 0.3m to 0.4m, that value drops from 69% to about 3% which means seldom packets can be detected. That indicates our backscatter tag should work in the area near the excitation source.

C. Communication Performance

We now evaluate the end-to-end performance of our backscatter with commodity radios of four protocols. The tag is placed 0.15m away from the transmitter, and the transmitter power is set to the **High** state noted in Fig.8(a). We measure throughputs and BERs of our backscatter system with different tag-to-receiver distances.



(a) Throughput. (b) BER .

Fig. 11. Throughput and BER of our backscatter with ZigBee device when radios of different protocols coexist.

The result is shown in Fig.10. We can see that the maximum communication distances with 802.11n, 802.11b, BLE and ZigBee radios are 14m, 16m, 20m, and 20m, respectively. And at distance no more than 2m, tag data throughputs are about 50Kbps, 110Kbps, 4Kbps, and 4Kbps, respectively. The low throughputs with BLE and ZigBee are caused by the packet rate limitation in our BLE and ZigBee chips. The BER of tag data with 802.11n radios is the highest among those protocols, which is about 10% even when the tag-to-receiver distance is 2m. With 802.11b radios, the BER is lower than 10% when the distance is below 14m. With radios of BLE or ZigBee, BERs are much lower. At the distance of 2m, they are both about 0.1%, and even when the distance increases to 20m they are still lower than 5%. Those experiment results show that our backscatter can realize communication using excitation signals of commodity radios.

D. Coexistence Performance

We also evaluate the performance of our backscatter when interfering radios exist. Specifically, we evaluate the performance of backscatter with ZigBee devices when BLE transmitter or 802.11b transmitter exists. And the transmitter powers of the ZigBee, BLE, and 802.11b are all set to the High state noted in Fig. 8. We define four scenarios: *Only ZigBee*, *ZigBee and BLE*, *ZigBee and half-speed WiFi*, and *ZigBee and full-speed WiFi*, standing for the scenario ZigBee transmitter and receiver coexist with no extra radio, or BLE transmitter, or 802.11b device that transmits packets at half of the maximum packet rate, or 802.11b device that transmits packets at half of the maximum packet rate, respectively.

As shown in Fig.11, the BLE transmitter influences throughput and BER of our backscatter with ZigBee slightly. While half-speed 802.11b WiFi reduces throughput from about 4Kbps to below 2Kbps and increases BER from below 5% to about 25%. And full-speed 802.11b WiFi makes the performance even worse. Our BLE transmitter transmits at most 70 packets per second, and a BLE packet has a duration of about $300\mu s$, so the total transmitting time of BLE in a second is $300\mu s \times 70 = 21000\mu s$. That's to say, the BLE transmitter stays idle at about 98% of the time. While such proportion in half-speed 802.11b WiFi and full-speed 802.11b WiFi are about 50% and below 5%, respectively. We can conclude that our backscatter can realize communication when interfering radios exist, and the busier the interfering radios are, the worse our backscatter system performs.

IV. Conclusion

Existing backscatter systems are not able to work with multiple ambient signals. In this paper, we have introduced the backscatter system capable of identifying WiFi, BLE, and ZigBee signals which is the first and essential step for using those signals for backscatter communication. We have built an FPGA prototype and conducted extensive experiments to evaluate the protocol-identification effectiveness. And the ability to use those signals for communication is also tested. Experiments show that our backscatter can get an average identification accuracy above 90% at 2.5Msps. Our backscatter has achieved communication using excitation signals of WiFi, BLE, and ZigBee. When the WiFi signal acts as the carrier our backscatter achieves throughput as high as 110Kbps, and with the signal of BLE or ZigBee, the BER of out backscatter system can be as low as about 0.1%. For short, our backscatter can identify signals of WiFi, BLE, and ZigBee, and it supports communication with all those signals as carriers.

Acknowledgment

This work was supported by NSFC Grant No. 61932017 and 61971390, and the Fundamental Research Funds for the Central Universities No. WK2150110013.

References

[1] K. Finkenzeller, *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. John Wiley & Sons, 2010.

[2] J. D. Griffin and G. D. Durgin, "Complete link budgets for backscatter-radio and rfid systems," *IEEE Antennas and Propagation Magazine*, vol. 51, no. 2, pp. 11–25, 2009.

[3] P. Zhang, D. Bharadia, K. Joshi, and S. Katti, "Hitchhike: Practical backscatter using commodity wifi," in *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*, ser. SenSys '16, 2016, pp. 259–271.

[4] P. Zhang, C. Josephson, D. Bharadia, and S. Katti, "Freerider: Backscatter communication using commodity radios," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17, 2017, pp. 389–401.

[5] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions," in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI'16, 2016, pp. 151–164.

[6] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith, "Inter-technology backscatter: Towards internet connectivity for implanted devices," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16, 2016, pp. 356–369.

[7] Y. Li, Z. Chi, X. Liu, and T. Zhu, "Passive-zigbee: Enabling zigbee communication in iot networks with 1000x+ less power consumption," in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '18, 2018, pp. 159–171.

[8] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti, "Backfi: High throughput wifi backscatter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 283–296, 2015.

[9] J. Zhao, W. Gong, and J. Liu, "Spatial stream backscatter using commodity wifi," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2018, pp. 191–203.

[10] X.-G. Xia, "A family of pulse-shaping filters with isi-free matched and unmatched filter properties," *IEEE Transactions on Communications*, vol. 45, no. 10, pp. 1157–1158, 1997.

[11] N. S. Alagha and P. Kabal, "Generalized raised-cosine filters," *IEEE transactions on Communications*, vol. 47, no. 7, pp. 989–997, 1999.

[12] F. Lu, P. Ling, G. M. Voelker, and A. C. Snoeren, "Enfold: Down-clocking ofdm in wifi," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '14, 2014, pp. 129–140.