Enabling Native WiFi Connectivity for Ambient Backscatter

Longzhi Yuan
University of Science and Technology of China
longzhi@mail.ustc.edu.cn

Wei Gong*
University of Science and Technology of China
weigong@ustc.edu.cn

ABSTRACT

WiFi backscatter communication has required unwanted constraints on either excitations or receivers since its inception eight years ago. We present Chameleon, the first native WiFi backscatter system where WiFi tags can generate native WiFi packets using uncontrolled productive WiFi as carriers. Our tag-only solution requires no particular excitation patterns and no change for software/hardware on WiFi NICs. The key insight is that the Chameleon tag can demodulate productive WiFi and backscatter this arbitrary carrier into a full-function packet using on-the-fly modulation. We prototype WiFi tags using ultra-low-power FPGAs and evaluate them in real-world scenarios where excitations are ambient traffic and backscatter receivers are a range of COTS NICs. Comprehensive field studies show that the maximal backscatter throughput of Chameleon is almost 1 Mbps, which is over 125× and 1000× better than WiTAG and FS-Backscatter. Also, we show that Chameleon can natively communicate with various COTS WiFi devices on Windows, iOS, and Android platforms. We believe this native WiFi backscatter design will enable ubiquitous WiFi connectivity for billions of IoT devices via widely available mobile gadgets and existing wireless infrastructure.

CCS CONCEPTS

• **Networks** → **Layering**; **Network design principles**;

KEYWORDS

Backscatter, WiFi, 802.11b

ACM Reference Format:
Longzhi Yuan and Wei Gong. 2023. Enabling Native WiFi Connectivity for Ambient Backscatter. In *ACM International Conference on Mobile Systems, Applications, and Services (MobiSys '23), June 18–22, 2023, Helsinki, Finland.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3581791.3596868

1 INTRODUCTION

Ever since its inception [29], WiFi backscatter communication has become one of the most promising ways to deliver the Internet of WiFi things where WiFi tags can reuse ambient signals and connect to the Internet with WiFi infrastructure [5, 6, 10, 20, 24, 30, 33, 43, 47–52]. This direction is fascinating because there are over 3 billion WiFi devices globally shipped each year [4]. If WiFi backscatter tags, as shown in Fig. 1, can be embedded into everyday objects and connect to Internet with the help of those massively deployed WiFi devices, the vision of pervasive connectivity for everything [12] could soon become a reality.

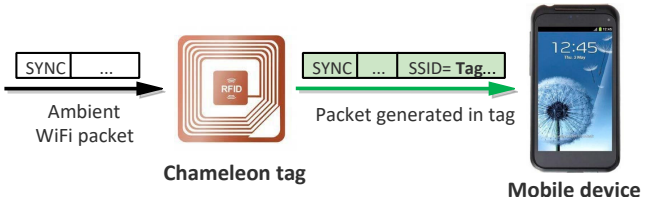


Figure 1: Conceptual design of Chameleon that generates native WiFi packets out of uncontrolled ambient 802.11b signals, ensuring 100% compatibility with COTS WiFi devices.

The above dream seems pleasant and exciting, but the reality is harsh and painful. Different from traditional backscatter communication, e.g., RFID, and active radios, such as WiFi and Bluetooth, one of the unique features of ambient backscatter is the content varying carriers instead of continuous waves (CW). To tackle this challenge, the first generation of WiFi backscatter systems [5, 29, 49] adopts packet-level ASK modulation on RSSIs by reflecting or absorbing packets. While this idea is simple and easy to implement, those packet-based systems inevitably suffer from poor downlink throughput and barely work when the excitations are sporadic and intermittent. Afterward, the second generation attempts to achieve high throughput using symbol-level backscatter and introduces an additional receiver to obtain the productive data from carriers [47, 48]. Even though tens of Mbps throughput can be achieved using either SDR [33] or commodity NICs [44], the two-receiver design requires both receivers working and synchronizing well, and thus experiences high communication instability [20].

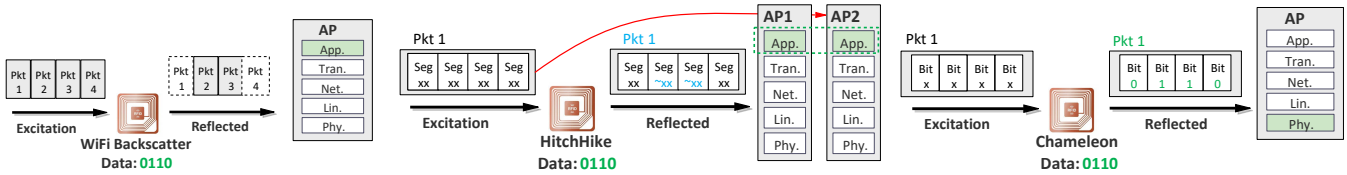
Besides, all prior ambient WiFi backscatter systems are not WiFi native. Ideally, we expect that a COTS WiFi NIC is adequate to demodulate tag data from a backscattered packet alone, the same way as demodulating an active WiFi packet. So we would call a backscatter system WiFi native if a WiFi NIC does not need to distinguish active or backscattered packets and achieves tag data transparency on the PHY layer. Yet, as shown in Fig. 2, the first-generation systems are not native since they have to demodulate tag data from RSSIs by writing an App on the application layer [29, 49]. The second-generation systems are not native either because they need to perform reverse codeword translation either on the application layer [47, 48] or modified WiFi PHY layer [33].

We deduce that prior systems lose support for native WiFi because their approaches are indirect, making the problem of content varying carriers just postponed but not overcome. In particular, the first generation does not touch the varying content and chooses

*Wei Gong is the corresponding author.

MobiSys '23, June 18–22, 2023, Helsinki, Finland

ACM ISBN 979-8-4007-0110-8/23/06…$15.00
https://doi.org/10.1145/3581791.3596868



(a) 1st generation WiFi backscatter. (b) 2nd generation WiFi backscatter. (c) Our proposal.

Figure 2: Comparison of three generations for WiFi backscatter where our proposal is the first native WiFi backscatter system.

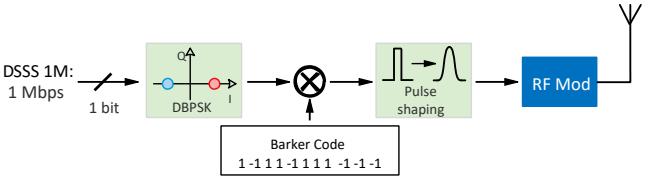


Figure 3: 802.11b signal generation.

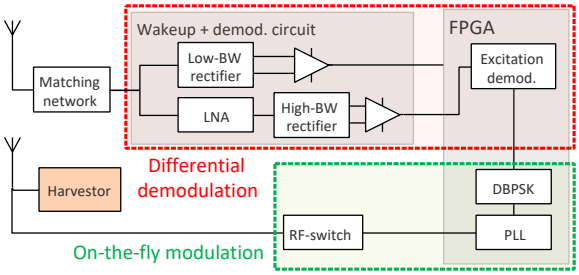


Figure 4: Chameleon overview.

to work on the packet level. Those second generation systems use codeword translation alike to encode tag data, making an extra dedicated receiver for varying productive data necessary and becoming innative. In contrast, we decide to address this fundamental problem directly, i.e., we demodulate ambient WiFi signals on the tag, removing negativity brought by content-varying carriers once and for all.

We present Chameleon, the first native WiFi backscatter system that can backscatter productive carriers into native WiFi packets. As aforementioned, previous ambient WiFi backscatter systems are not native and all the means to generate native WiFi packets requires CW. Instead, Chameleon proposes a bold idea; if the tag can demodulate the productive carrier, the whole carrier would become a virtual CW. We only need to modulate the difference between productive data and target tag data. Doing so brings several unprecedented advantages. 1) *Native.* Our backscattered packet is of full function since every symbol is under control. 2) *Transparent.* A single WiFi NIC can receive our backscattered packets without the need for even software change because our packet is native from the PHY layer to the application layer. 3) *Fast.* The backscattered packets can achieve the same rate as the carrier.

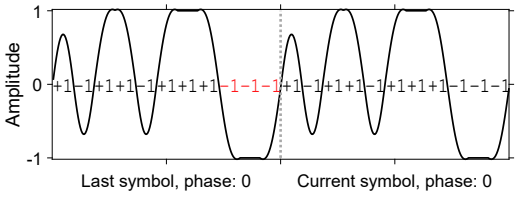
Repurposing ambient content-varying carriers is challenging for two main reasons.

- Passive Demodulation for WiFi. Although recent advanced systems introduce several novel designs for backscatter downlinks, none of them can demodulate productive WiFi data. MIXIQ [37] does not work with random content-varying WiFi because it requires an intelligent (constrained) helper signal. Saiyan [21] introduces a frequency demodulation method and thus does not fit for phase-modulated WiFi. Passive DSSS [31] is a closed-loop system where both excitations and receivers are specially dedicated, incompatible with commodity WiFi. Unlike those works, we attempt to demodulate 802.11b WiFi signals based on a key observation that phase modulated 802.11b signals can be differentiated by their pulse widths, which is detailed in Section 2.2. To ensure high-quality demodulation, we also design a novel template-matching-based synchronization to achieve accurate symbol timing.
- Backscatter Modulation. To support native WiFi backscatter, it requires modulation dependent on the carrier and tag data at the same time. However, all the prior backscatter modulation methods depend only on tag data [10, 24, 47]. As a result, we design on-the-fly modulation, which creatively backscatters the difference between the tag data and the carrier data, ensuring the backscattered packets 100% native.

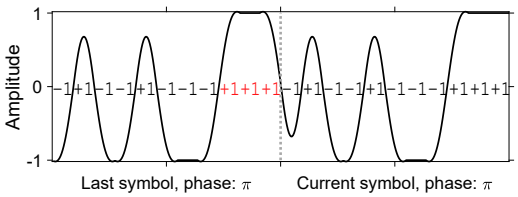
We build several tag prototypes using low-power FPGAs and IC simulation. Various real-world tests show that Chameleon realizes an uplink throughput of 992.4 kbps, which is over 125×, over 1000×, and over 100000× higher than WiTAG, FS-Backscatter, and WiFi-Backscatter, respectively. Moreover, we show that the backscattered signals of Chameleon can be natively supported by a range of COTS WiFi devices, including different laptops, tablets, and smartphones. Currently, only the DBPSK-802.11b is supported in Chameleon. This is because the different pulse width patterns for bit '0' and bit '1', which are the key for differential demodulation, only exist in DBPSK-802.11b. Finally, we show that the battery-free prototype that can harvest energy from RF signals and office light achieves up to 9.2 pkts/s in real-world applications.

Contribution: Our technical contributions are as follows.

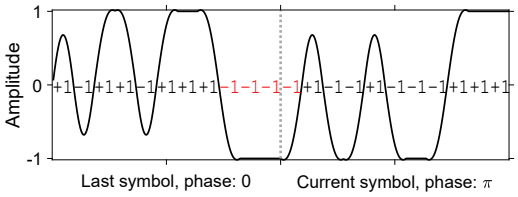
- We propose the first passive demodulation for 802.11b WiFi, of which the insight is to use pulse widths to distinguish WiFi symbols.
- We design a novel on-the-fly modulation that can turn arbitrary 802.11b carriers into native WiFi packets.



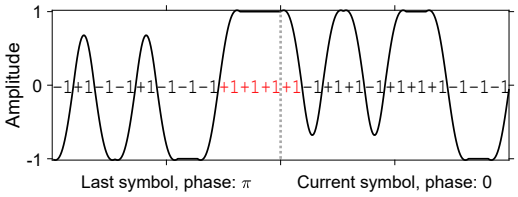
(a) Phase <0, 0>, bit '0'.



(b) Phase <π, π>, bit '0'.

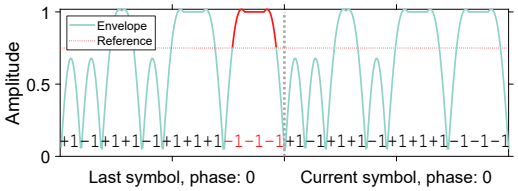


(c) Phase <0, π>, bit '1'.

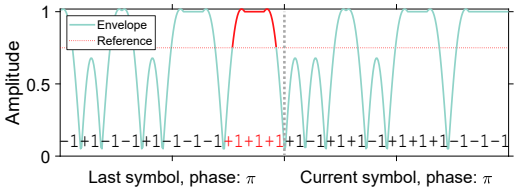


(d) Phase <π, 0>, bit '1'.

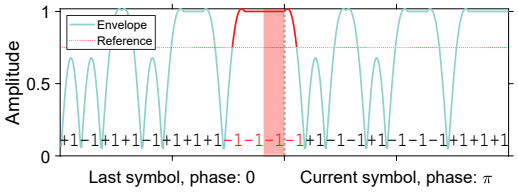
Figure 5: Waveforms of two consecutive symbols on the transmitter side.



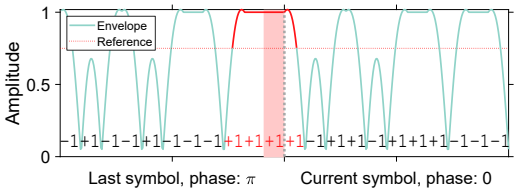
(a) Phase <0, 0>, bit '0'.



(b) Phase <π, π>, bit '0'.



(c) Phase <0, π>, bit '1'.



(d) Phase <π, 0>, bit '1'.

Figure 6: Envelopes of two consecutive symbols on the tag side.

- We conduct extensive prototypes and experiments to show Chameleon's feasibility and effectiveness.

2 DESIGN

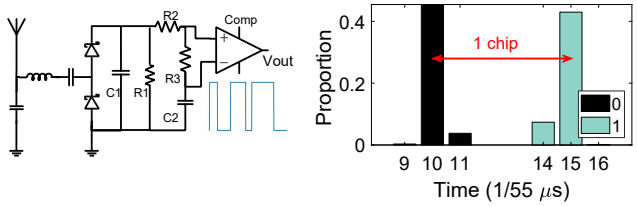
2.1 Overview

As shown in Fig. 4, Chameleon mainly consists of the harvesting module, differential demodulation, and on-the-fly modulation. Differential demodulation detects ambient RF signals, finds ongoing packets, and then wakes up the system. Afterward, a high-bandwidth rectifier extracts the binary envelope and feeds it to the demodulation algorithm on an FPGA. The carrier protocol can be identified using envelope signal as Multiscatter[20]. And whether it is the target DBPSK-802.11b is determined in the demodulated its physical header, which is modulated using DBPSK in all 802.11b packets. While the carrier data is being demodulated, the tag modulates tag data together with demodulated carrier data on the fly. This enables reshaping the old WiFi packet into another native WiFi packet. The harvester module is embedded into our battery-free prototype that will be introduced in Section. 3 to harvest from ambient light and RF signals.

2.2 Observations of Symbol Pulses

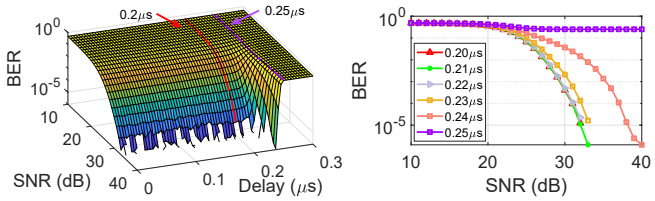
According to WiFi standards, the symbols of 802.11b packets are phase modulated. So the common wisdom for demodulation is to extract the phases of symbols through a mixer, which is usually power hungry for high-frequency signals. Instead, we seek to use a low-power envelope detector to finish this job. Specifically, we will walk through how DBPSK 802.11b signals are generated on the WiFi transmitter side and what those signals would be like after an envelope detector on the tag side. The procedure is shown in Fig. 3.

At the WiFi transmitter, a series of raw data bits are first modulated using DBPSK, then are spread with an 11-bit Barker code sequence, and pass through a shaping filter, e.g., Gaussian, Root



(a) Envelope demodulation circuit.(b) Pulse widths at symbol boundaries.

Figure 7: We design an envelope-based demodulation circuit and use it to measure pulse widths at symbol boundaries for random 802.11b signals. Results show that the pulse widths of '0' and '1' are distinguishable.



(a) $0\mu s \sim 0.3\mu s$. (b) $0.2\mu s \sim 0.25\mu s$.

Figure 8: Downlink demodulation BER vs controlled sync errors.

Cosine, to remove unwanted high frequencies. During DBPSK modulation, raw data bits '0' and '1' are mapped to phase differences of 0 and π between two adjacent symbols. To modulate a bit '0', the phase of current symbol will be exactly the same as the last one, e.g., $< 0, 0 >$ in Fig. 5a or $< \pi, \pi >$ in Fig. 5b. Similarly, a bit '1' can be mapped to two symbols whose phases differ by π, such as $< 0, \pi >$ in Fig. 5c or $< \pi, 0 >$ in Fig. 5d.

After DBPSK modulation, the transmitter performs DSSS to spread codes, which are originally designed for interference suppression. In the 802.11b standard, this spreading sequence is a 11-bit Barker code, which is predefined as [+1−1+1+1−1+1+1+1−1−1−1]. This operation divides an 802.11 symbol into 11 chips, each of which is $\frac{1}{11}\mu s$. Along with the spreading process, we observe that if the raw bit is '0', there are three identical chips around the symbol boundary on matter the symbol phases are $< 0, 0 >$ or $< \pi, \pi >$, as shown in Fig. 5a and 5b. On the other side, if the raw bit is '0', we always observe four identical chips at the symbol boundary, as shown in Fig. 5c and 5d. Such observations motivate an idea; if we can tell differences between the pulse width at the symbol boundary, deducing the raw data bits is made easy. Let us continue the transmission process and see whether the above features hold. The next important procedure is the shaping filter. So we adopt a common Gaussian filter to check its effects. As shown in Fig. 5, the change from +1 to -1 or from -1 to +1 just becomes smooth and does not destroy pulse-width features for different raw data bits. Obviously, the signal envelope is not constant anymore. Therefore, we guess it is possible to demodulate this WiFi signal by measuring the pulse widths at symbol boundaries.

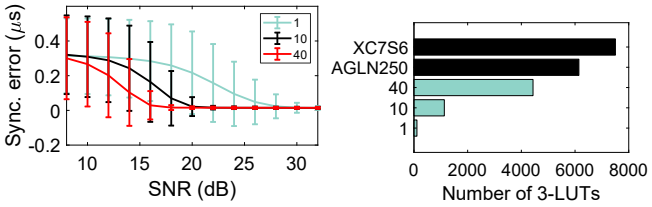
To verify this idea, we design a passive rectifier to extract the baseband envelope. From the acquired envelopes on the tag shown in Fig. 6, it is clear that the high duration of a raw bit '1' is longer than that of a raw bit '0'. The additional identical chip causes such a difference as we analyze the transmission process previously. Specifically, this difference equals to the duration of a single chip, $\frac{1}{11}\mu s$. To further corroborate this, we use a low-power comparator to digitize the rectifier output, whose circuit design is shown in Fig. 7a, and process the sampling results in the FPGA. Through extensive evaluation, we plot a subset of our empirical measurement results in 7b, which contain 1000 packets, and each has 100 bytes payload. From the figure, we can see that two clusters are naturally formed, of which the centers are $\frac{10}{55}\mu s$ and $\frac{15}{55}\mu s$. The distance between the two centers is exactly $\frac{1}{11}\mu s$.

After reviewing the transmission process and WiFi envelopes on the tag, we can conclude that we are able to demodulate 802.11b signals on the tag by examining envelope differences. From mission impossible to possible, the key is that 802.11b DSSS using barker codes makes differences at symbol boundaries, i.e., a raw bit '1' has a discernible longer pulse width than a raw bit '0', which achieves the first passive WiFi demodulator.

2.3 Differential Demodulation with Accurate Timing

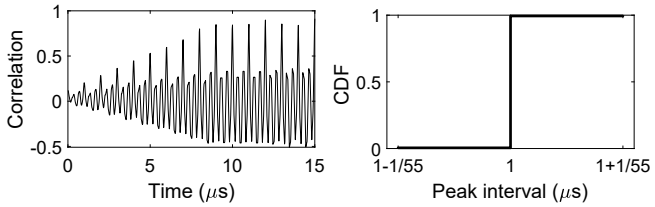
From the previous discussion, we show passive WiFi demodulation is possible if the pulse widths of symbol boundaries are acquired. So we first examine how accurate symbol synchronization we should achieve and then design a low-power solution using FPGA implementation to fulfill the requirements.

2.3.1 Timing Requirements. For synchronization requirements, we conduct simulations in *Matlab* to investigate demodulation BERs under different controlled synchronization errors. Fig. 8 shows the relationship between demodulation BER and the timing delay. As can be seen in Fig. 8a, the delay below $0.2\mu s$ has negligible impact on demodulation quality. But when it is more than $0.25\mu s$, the demodulation is bound to fail. Fig. 8b provides the BER when the timing delay is between $0.2\mu s$ and $0.25\mu s$. To conclude, when the delay exceeds $0.2\mu s$, its further increase will cause sharp BER rise. So, we should control the delay below $0.2\mu s$ to keep the downlink demodulation sound. However, none of the prior synchronization schemes of backscatter systems can meet this requirement. For example, the synchronization error of MIXIQ is as high as 0.8 μs [37], and that of Saiyan [21] is even worse, which is of milliseconds. Hitchhike [47] does not fit either since its synchronization accuracy is around 2 μs. The most recent work, SyncScatter [10] can realize the synchronization accuracy as low as $0.15\mu s$. Yet, it only works with excitations from a fixed transmitter and may degrade to over 1 μs synchronization error due to dynamic power ramp time of ambient WiFi packets [22]. Actually, all rising edge-based synchronization, including [8, 10, 47], suffers from this drawback. As a result, our goal is to design a symbol synchronization on the tag, whose accuracy is within $0.2\mu s$.



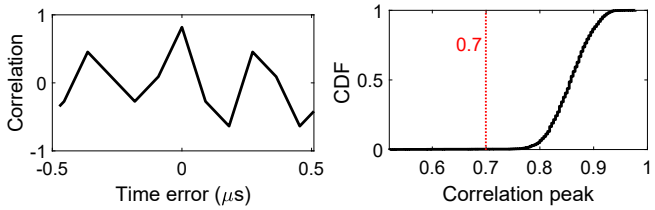
(a) Sync errors at different template lengths. (b) Required FPGA resources for different templates.

Figure 9: Optimal template length for low-power sync.



(a) Correlation results with an 802.11b packet. (b) CDF of peak intervals for packets of random payload.

Figure 10: Correlation peaks to locate symbol boundaries.



(a) Correlation waveform of bit '1' sequence with that of bit '0'. (b) CDF of peak values.

Figure 11: Periodic correlation peaks.

2.3.2 Low-power Synchronization using Correlation. Different from all prior synchronization on tags, we choose correlation to realize accurate timing. Correlation-based timing is widely used in active WiFi [1, 2] and proves its high performance in many real-world applications. The synchronization on the tag is designed as follows. First, we choose the binary envelope corresponding to the leading symbols in the preamble as the reference sequence. Then, a new sample is collected using the rectifier circuit and the comparator in FPGA every clock cycle. After the newest samples are correlated with the reference waveform, we use correlation peaks to identify symbol boundaries. Specifically, we judge the symbol boundary based on whether the correlation peak exceeds a predefined threshold. The following boundaries are deduced every $1\mu s$, which is the time duration of a standard 802.11b symbol.

Several important design considerations are discussed as follows.

Template design. Simply porting correlation-based synchronization for active WiFi would bring significant computation overhead for resource-constrained tags. On the other hand, if we oversimplify the correlation process, the synchronization error would go skyrocket. Hence, our design must carefully strike a balance between computation overhead and accuracy.

- First, we reduce computation overhead by making sample points binary values, unlike 10 or 16 bits for each sample in active WiFi.
- Second, we try to find a minimum template length that can deliver decent performance. In particular, we vary the template length and plot the results in Fig. 9a. We can see that if the template of 1-symbol long, the synchronization error drops below $0.1\mu s$ only when the SNR exceeds 30dB, which is unacceptable. In contrast, When the template length is 10 or 40, the required SNRs are 20dB and 18dB, respectively. Hence, we pick 10 as a candidate for template length but need to verify its consumed hardware resources later.
- Third, we choose to use addition to replace multiplication for further computation saving. Multiplication of two 1-bit operands can be realized using an **AND** gate. The FPGA can sum up the output of all **AND** gates to obtain the correlation results.

As an example, let us put the whole synchronization operation in an ultra-low-power FPGA, nano AGLN250 FPGA. This simplified correlation can be achieved using three-input look-up tables (3-LUT). Based on this, we calculate how many 3-LUTs are necessary for different template lengths and the results are shown in Fig. 9b. We can see that when the template lengths are 1, 10, and 40-symbol long, the required number of 3-LUTs are 111, 1131, and 4435, respectively. At the same time, the AGLN250 FPGA can provide as many as 6144 ones, and the XC7S6 FPGA has 7504 ones. Therefore, we confirm that setting the template length at 10 makes a good tradeoff between hardware resources and synchronization accuracy.

Synchronization accuracy. To verify our solution's feasibility, we test it with 1000 packets of random payload. We plot a representative correlated waveform in Fig. 10a and the statistics of peak intervals in Fig. 10b. As each packet starts at $0\mu s$, we observe that the correlation has a high peak at $10\mu s$ (corresponding to the end of the tenth symbol), which means our tag is well synchronized to the tested 802.11b packet. In addition, in more than 99% of cases, the peak interval is $1\mu s$, which exactly matches the time duration of a single 802.11b symbol. Someone may notice the wrong peak instants have a drift of $\pm\frac{1}{55}\mu s$. This small error is negligible since the clock rate for sample collection is 55MHz, meaning such a drift only corresponds to a single clock cycle. In short, the above tests demonstrate our solution locates symbol boundaries and meets design requirements at a low cost.

One may be curious how come our fixed template can locate boundaries for both bits '1' and '0' in the presence of random payload. The answer is that 802.11b designs similar binary sequences (barker codes) for bits '0' and '1'. To prove this, we correlate the waveform of bit '0' with that of '1' as shown in Fig. 11a. It shows that the peak still appears when the two bits are aligned. Furthermore, we find that the values of over 99.9% of those periodic peaks are higher than 0.7, as shown in Fig. 11b. The exceptions are when a new packet just starts, and there are not enough samples for correlation, as the nine short peaks in 0-9 μs in Fig. 10a. In this case, after checking whether the peak is above 0.7, we further make sure

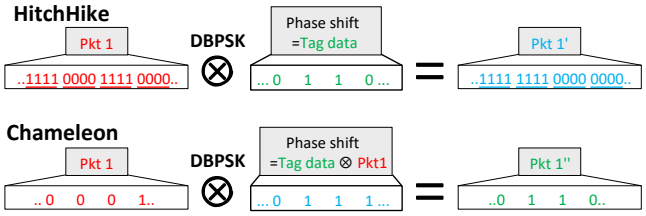
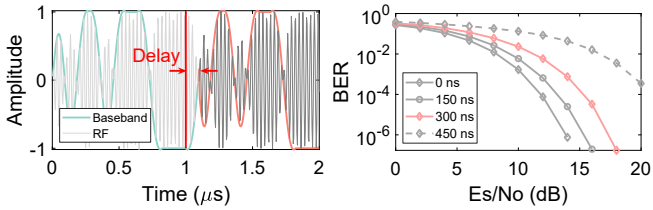


Figure 12: Our tag considers both excitation and tag bits for backscatter modulation, making any random signal into a native WiFi packet. In contrast, codeword-translated-based systems, e.g., Hitchhike, are not native.



(a) Cause of modulation delay. (b) Modulation delay vs BER.

Figure 13: Modulation delay and its impact.

this peak should be higher than the neighboring peaks. Therefore, all the expected symbol boundaries are located by these periodic peaks.

Wrap up. Finally, we summarize the demodulation in the following steps. First, the tag counts the duration of the high envelope pulse, and resets at every rising edge. At the same time, the tag monitors the correlation result. If a peak is detected, we demodulate the current bit by distinguishing whether the pulse width is long enough.

2.4 On-the-Fly Modulation

After we demodulate content-varying 802.11b packets, our next job is to modulate those random packets into native WiFi packets. In particular, we present our modulation scheme and then demonstrate how we handle modulation delay and demodulation errors. Of course, the frequency shifting is leveraged to avoid interference from carrier signal[49].

2.4.1 Modulation Scheme. The native WiFi backscatter problem can be formulated as follows. Given tag data, e.g., {..., 0, 1, 1, 0, ...}, and content-varying excitations, a native WiFi backscatter system should generate WiFi packets exactly containing tag data as payload. Such a design goal would enable a range of novel applications as any standard WiFi device can directly demodulate this backscattered signal without any problems. However, none of the prior systems can achieve this because their modulation depends only on the tag data. From the above definition, we know there is no way to be native if the modulation is unaware of varying content because the target packet is fixed. As a result, we design on-the-fly modulation, which removes the uncertainty brought by the content-varying 802.11b packets. In particular, it includes the demodulation results in our modulation, i.e., our modulation depends on carrier data and tag data at the same time. For example, as shown in Fig. 12, Hitchhike [47] applied phase shift modulation according to tag data only, making the requirements of excitation and backscattered data necessary for demodulating tag data. In contrast, our modulation is the difference between tag data and carrier data, naturally leading to a native packet.

For FPGA implementation, we first store the current tag data bit in a register. After the carrier data is demodulated, we obtain the modulation bit as the **XOR** of the demodulation bit and the tag bit. Then after applying the modulation bit on the carrier, the backscattered bit makes the perfect tag bit as expected. In short, given content-varying excitations, only our on-the-fly modulation that includes dynamic demodulation results can generate native target packets.

2.4.2 Modulation Delay. To ensure robust modulation, our on-the-fly modulation needs to take care of unavoidable modulation delays caused by symbol boundary locating. After we locate a symbol boundary, a current symbol's start has been missed. Hence, the intended phase shift cannot be applied to the whole excitation symbol. As depicted in Fig. 13a, the phase shift occurs *Delay*, which is after the start of the new symbol. To accurately quantify the impact of this delay, we perform another set of controlled experiments, where each is tested with 1000 random packets with various delays. From the modulation BER results shown in Fig. 13b, when the delay is less than $0.3\mu s$, BERs drop slightly. When the delay is $0.45\mu s$, we have to increase 8 dB for signals to realize similar BERs at a $0.3\mu s$ delay. From those experiments, we can see that if the delay is within $0.3\mu s$, the backscatter quality would be adequate for many real-world applications.

Next, we want to examine whether the empirical modulation delay can meet the above goal. In real-world scenarios, this modulation delay includes timing offset caused by synchronization algorithm and hardware delay brought by the passive rectifier, the comparator and FPGA processing. In our ADS simulation, the 40MHz passive rectifier introduces a delay of about $0.01\mu s$, and the low-power comparator takes about $0.05\mu s$. As for the FPGA processing delay, half of the time corresponding to the first Barker chip of a symbol, whose specific duration is $\frac{1\mu s}{11}$, should also be countered. That is because it is vital for demodulation, and only after demodulation will tag be able to conduct on-the-fly modulation. In our implementation, the FPGA spends additional 3 clock cycles in logic processing and propagation. Therefore, the whole delay (excluding timing offset) can be estimated to: $0.01\mu s$(by rectifier)+$50\mu s$(by comparator)+$\frac{1}{2} \times \frac{1\mu s}{11}$(by half of the Barker chip for decoding)+$3 \times \frac{1\mu s}{55} = 0.16\mu s$. There lefts only $0.3\mu s - 0.16\mu s = 0.14\mu s$ for the synchronization algorithm, which can be easily satisfied as long as SNR exceeds 20dB, as shown in Fig. 9a.

Since it is nearly impossible to measure the accuracy of symbol synchronization in practice, here we investigate its impact through an indirect way, end-to-end experiments. Besides the delays introduced by the synchronization circuit or FPGA processing, we manually delay the backscatter modulation in the tag and then observe BERs from a COTS WiFi NIC. Results are shown in Fig.

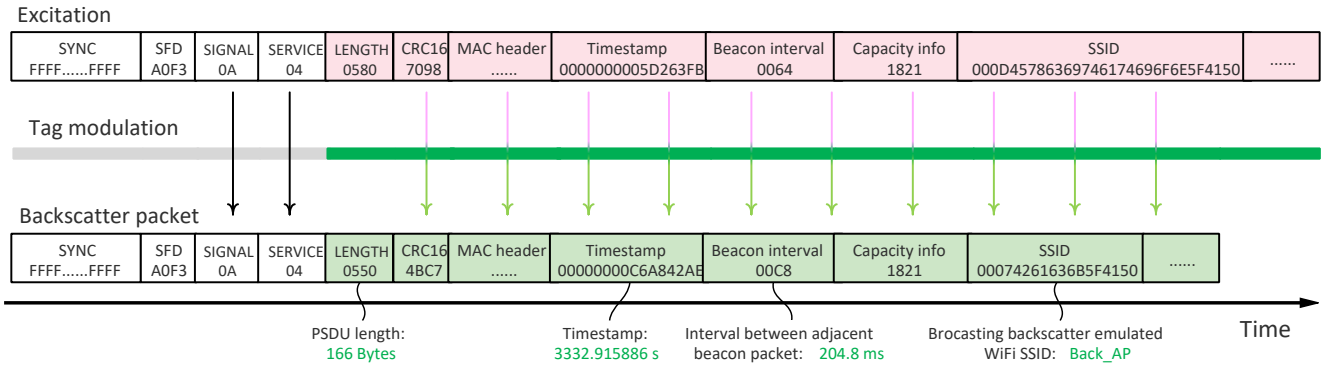
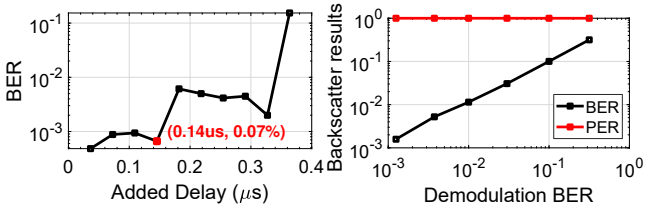


Figure 14: How Chameleon takes an old (random) beacon packet and turns it into a new beacon packet.



(a) Timing offset vs BER.

(b) Impact of demodulation BER and PER on backscatter modulation.

Figure 15: On-the-fly modulation performance.

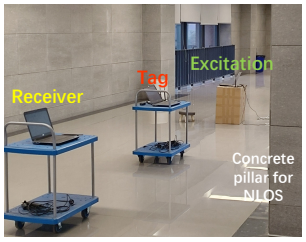


Figure 16: Experiment scenario.

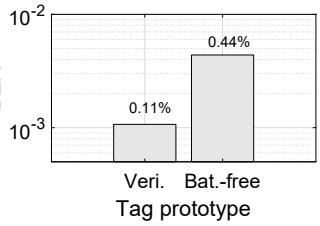


Figure 17: Performance of two prototypes.

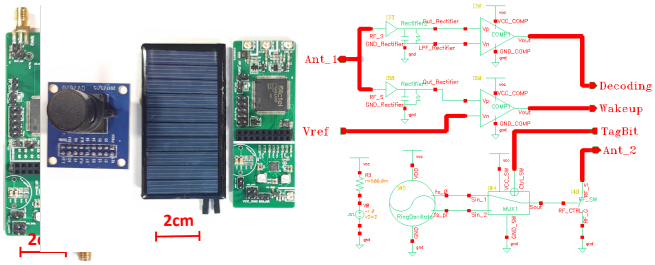
15a. When the additional delay is within 140 ns, the decoding BER of backscatter modulation is below 0.1%. The BER is not as low as in the simulation results of Fig. 13b. That is because, in real-world experiments, ambient WiFi transmissions interfered with backscatter modulation. On the contrary, when the delay exceeds 140 *ns*, the BER increases significantly. Still, these results show that our synchronization design provides a margin of around 140 *ns*, which meets our design goal.

2.4.3 Impact of Demodulation Errors. Different from prior works, our backscatter modulation is highly dependent on the demodulation quality. If the demodulation results are incorrect, backscatter transmissions will be significantly affected. This is a limitation of Chameleon. Hence, we need to investigate the relationship between demodulation BERs and backscatter modulation performance.

As 1 Mbps 802.11b signals are modulated using DBPSK, the error bits bring additional phase shifts to all the following backscatter modulated symbols. But the phase differences between them, which is used in tag data decoding in WiFi receiver, are kept intact. That means demodulation errors in tag will cause about the same amount backscatter errors. One may also concern that the demodulation errors in *preamble* or *header* will destroy backscatter transmission as they carry necessary information including the packet modulation method and the packet size. Two factors make this problem not fatal. First, the preamble contains 144 fixed bits. Chameleon tag stores them and then correct the corresponding errors. Second, the header contains only 32-bit packet information that is protected by a dedicated CRC. The demodulation of such a small-size field is much easier. Even if errors occur, they can be founded by the CRC. Thus, we manually introduce different amounts of demodulation errors in the *PSDU* field of the backscattered packets and then observe the backscatter modulation BER. As shown in Fig. 15b, the backscatter BER is close to the demodulation BER. That means our backscatter modulation can tolerate a decent amount of demodulation errors. But the PER equals to 1 with all error settings. That is because we manually introduce errors into all packets. In real scenario, the decoding errors distribute randomly among packets and their influence will not be so serious as our controlled experiments.

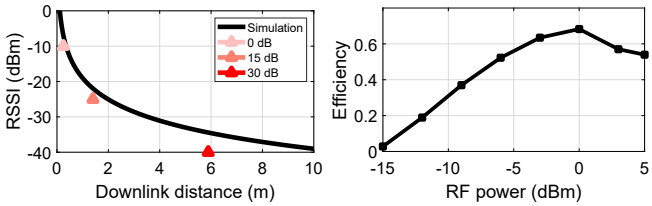
2.5 Case Study

Putting everything together, we showcase how our tag can backscatter a random beacon packet into another beacon packet whose content is predefined and native, as shown in Fig. 14. In our design, the tag does not modify the preamble and header fields and focuses on modulating the MPDU field. In particular, our tag frequency shifts SYNC, SFD, SIGNAL, and SERVICE fields to the target channel without any modification. Then, for the *LENGTH* and *CRC-16* fields, we modify them according to the target packet parameters. Note that the ability to change the CRC field is essential because all the packets regenerated by prior systems are discarded by COTS NICs [48, 50]. In the MPDU field, a broadcasting beacon with WLAN `SSID = Back_AP` is padded to emulate a new AP, which is our tag. The LENGTH field is set to 166 bytes according to the MPDU length.



(a) Tag-1. (b) Tag-2. (c) Tag-3.

Figure 18: Tag-1: verification prototype; Tag-2: battery-free prototype; Tag-3: IC simulation.



(a) Sensitivity and downlink distance. (b) Harvesting efficiency.

Figure 19: Backscatter sensitivity and harvesting performance.

Also, our tag changes the *TIMESTAMP* and *BEACON INTERVAL* fields, indicating our tag-AP broadcasts this message at a new time and different interval. To keep compatibility with WiFi radio, the final CRC is calculated and appended to corresponding fields. After our on-the-fly modulation, it turns the old beacon packet into a new beacon one, which is of full function, native, and acceptable to all COTS WiFi radios. This case cannot be achieved by any prior systems that take random WiFi as excitations.

3 IMPLEMENTATION

We built multiple prototypes to verify Chameleon's superiority in transmission performance and power consumption. The testing experiments are conducted in a 15m × 30m public area. Fig. 16 shows the scenario.

Verification prototype. We first build a verification prototype to explore the best performance of Chameleon. The key components, demodulation and backscatter modulation algorithms are realized in a Microsemi AGLN250 nano FPGA. The tag modulation is realized using an ADG902 RF-switch. In the decoding circuit, a passive rectifier composed of capacitors and HSMS-2862 diodes, and a high-speed NCS2250 comparator are used. As shown in Fig. 18a, our verification prototype includes two 15dB SKY65405-21 low noise amplifiers (LNA), which are designed for better downlink range. In this prototype, synchronization and backscatter modulation algorithms run at 55MHz. Note that a DSSS chip of 802.11b lasts for $\frac{1}{11}\mu s$, which can be easily recognized with a 20 MHz clock. In other words, the synchronization requirements are satisfied.

Battery-free prototype. We also design a battery-free prototype, as shown in Fig. 18b. While some parts are reused from the verification prototype, the LNA is not included for power reduction. We design a wakeup circuit to wake up FPGA from the *Flash*Freeze* state when excitation comes. It is composed of a low-bandwidth passive rectifier and a low-power comparator NCS2200. The running clock is also reduced from 55 MHz to 22 MHz. The direct benefit is a power reduction of about 2.5× for digital processing in FPGA. Furthermore, this also leads to shorter reference sequence for correlation-based synchronization, which also lowers system consumption. For RF harvesting, a similar rectifier circuit is used to convert the RF signal to DC voltage. After that, TI BQ25570 harvesting management chip stores energy in a storage capacitor and intermittently powers the system up.

It may be concerned whether the battery-free prototype can do the job. We test end-to-end performances of verification prototype and battery-free prototype with identical setting. Results are shown in Fig. 17. BER of 0.44% is still acceptable.

Power consumption comparison. We use a Keysight 34450A to measure power consumption for the above prototypes. Results are shown in Table. 1. The verification prototype costs 90.19 mW in total, including 15.41 mW by FPGA, 590 μW by an oscillator, 1.83 mW by RF-switch, and 72.36 mW by the wakeup and decoding circuits. The main dissipation comes from the FPGA, which runs at a high frequency. The wake and demodulation circuits contain two LNAs, which take about 72 mW and are not included in the battery-free prototype. Compared to the verification prototype, the power consumed by FPGA and wakeup/demodulation circuits is reduced to 5.23 mW and 360 μW in the battery-free design. In summary, over 10× power reduction is realized in the battery-free prototype.

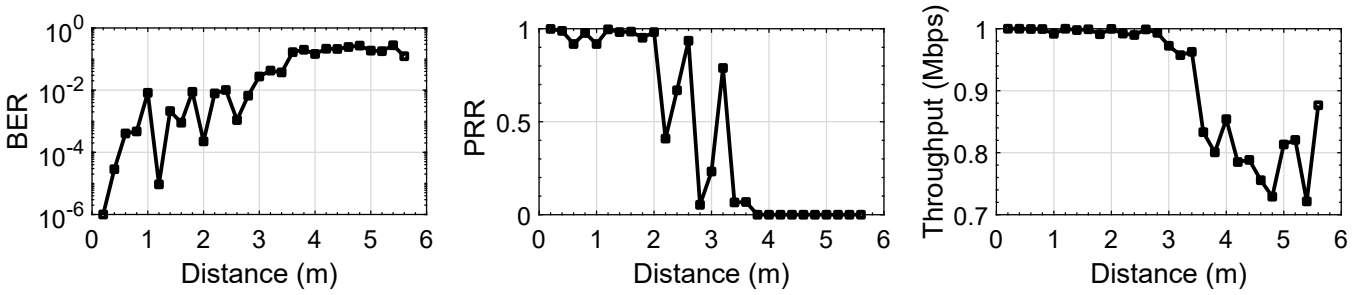
IC prototype. Although the application prototype consumes much less power than the verification prototype does, it is possible to make further power savings using ASIC techniques. In particular, we simulate an IC prototype using Cadence IC6.17 Virtuoso software and TSMC 0.18μm CMOS process design kits. The total power is estimated to be about 595.8 μW, which is only 0.66% of the verification prototype.

Ambient excitations and COTS receivers. We mainly use a TI CC3200 Wireless MCU (Campus WiFi) to provide excitations. For end-to-end and micro-benchmark experiments, we primarily deploy a laptop equipped with a Qualcomm Atheros AR938X NIC as the backscatter receiver because our competitor, e.g., FS-Backscatter, requires a WiFi NIC that can turn off CRC check. For receivers in application experiments, we include a range of COTS WiFi devices, including laptops, iOS and Android devices, to examine Chameleon's compatibility.

Sensitivity and downlink range. As the rectifier bandwidth in the demodulation circuit is as high as 40MHz, the corresponding sensitivity is estimated at about -10 dBm. Weaker excitation may still have a good quality to realize the downlink decoding, but it is not high enough to drive the comparator. We set the transmitter power to be 18 dBm, which is near to the maximum allowed by WiFi in most areas[3]. The simulated received signal strength indicator (RSSI) with different distances is depicted in Fig. 19a. It comes from the relationship between transmitter power P_{TX} and received power P_{RX}: $P_{RX} = P_{TX}G_{TX}G_{RX}(\frac{\lambda}{4\pi r})^2$, where G_{TX}, G_{RX},

Table 1: Power consumption of three prototypes.

	power consumption breakdown				Total
	Digital core	Oscillator	RF-switch	Wakeup+Demod.	
Verification	15.41 mW (100%)	590 μW (100%)	1.83 mW (100%)	72.36 mW (100%)	90.19 mW (100%)
Battery-free	5.23 mW (36.53%)	590 μW (100%)	1.83 mW (100%)	360 μW (0.50%)	8.41 mW (9.32%)
IC	324 μW (2.10%)	199.7 μW (33.84%)	2.41 μW (0.13%)	67.58 μW (0.09%)	595.8 μW (0.66%)



(a) BER. (b) Packet receiving ratio. (c) Throughput.

Figure 20: Differential demodulation performance.

and λ are transmitter antenna gain, receiver antenna gain, and wavelength, respectively. We use 3 dBi glue stick antennas in both transmitter and tag, $G_{TX} = G_{RX} = 3dB$.

As shown in Fig. 19a, the -10dBm sensitivity corresponds to a distance well below 0.5 m, which is not practical in everyday applications. So we consider deploying LNA in the tag as done in many related works[10, 21, 37]. If we use a single 15dB LNA, the sensitivity is improved to about -25 dBm. The measured downlink range improves to 1.4 m. And when we further cascade two LNAs, the sensitivity becomes about -40dBm, and the corresponding downlink distance increases to 5.9 m, which is welcome in most indoor applications.

Harvesting efficiency. We also examine the harvesting efficiency of our RF-harvesting module. In particular, we record the needed time T to charge the capacitor C_{store} from empty to full, in which procedure the capacitor voltage is increased from V_{min} to V_{max}. Combining the input RF power P_{in}, the harvesting efficiency can be calculated as: $\eta = \frac{C_{store}(V_{max}^2 - V_{min}^2)}{2P_{in}T}$. The result with different RF power levels is shown in Fig. 19b. The harvesting efficiency grows from 2.8% to 68.27% when the power level of RF signals increases from -15 dBm to 0 dBm. That's as with weak RF signals, the dissipation caused by the diode forward voltage in the passive rectifier circuit and charging management chip BQ25570 becomes significant and makes low efficiency. When RF power is below -15 dBm, such dissipation becomes not affordable, so harvesting fails.

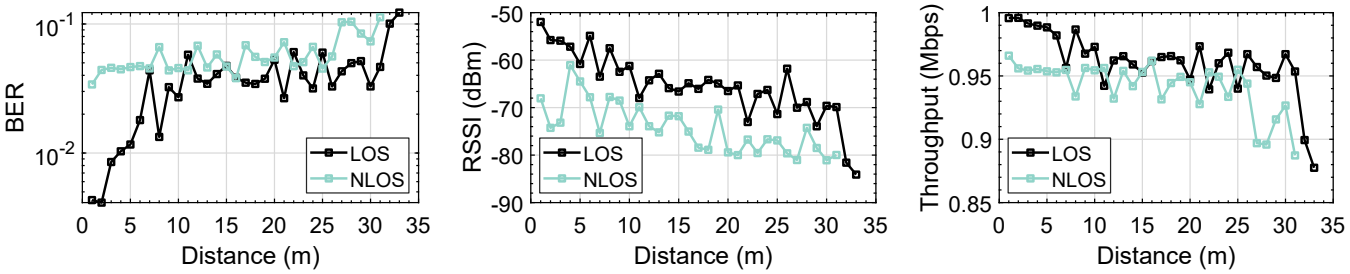
4 EVALUATION

First, we evaluate demodulation and modulation quality. We then investigate end-to-end performance and compare Chameleon against prior works. Finally, we demonstrate how Chameleon is compatible with all COTS WiFi devices.

4.1 Micro Benchmarks

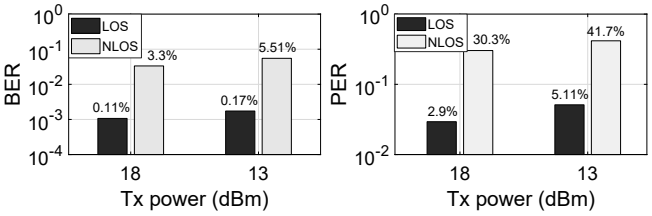
Differential demodulation. We conduct tests with different Tx-tag distances and set the transmitter to generate packets at the power level of 18 dBm. For each position, over 1000 random packets are sampled, and results are shown in Fig. 20. We observe that when the downlink range is below 1 m, the BER is less than 0.1%. When the range increases to 3 m, the BER jumps to 1%. Meanwhile, the packet receiving ratio (PRR) is above 90% when the downlink range is within 2m. Moreover, when the downlink range is less than 3 m, the throughput is always above 0.95 Mbps. Even better, the throughputs at all tested ranges are above 0.7 Mbps. Those figures indicate that our tag is able to demodulate 802.11b WiFi signals reliably.

On-the-fly modulation. Next, we examine how on-the-fly modulation works. Since the results of the synchronization algorithm have been showed in the previous section, we focus on its performance with different tag-Rx ranges. As shown in Fig. 21, the tag is placed 0.3 m away from the WiFi transmitter. Note that the previous discussion shows that the downlink demodulation can achieve nearly perfect excitation bits with this range. For NLoS experiments, we use a load-bearing concrete pillar to block signals from the transmitter to the tag. From results in Fig. 21, we see that BERs rise with increasing distances for both LoS and NLoS scenarios. In LoS cases, the BER is below 1% when the distance is shorter than 4 m. While in NLoS settings, the BER is above 3% at a distance of 1 m, which is mainly due to decreased RSSIs at longer distances and NLoS blockage. In addition, we observe that our tag supports maximal uplink ranges of 33 m and 30 m for LoS and NLoS scenarios, respectively. Furthermore, the backscatter throughput is above 0.85 Mbps in all tested settings. Such good performance is attributed to our on-the-fly modulation, which can tolerate reasonable delays and demodulation errors.



(a) BER. (b) RSSI. (c) Throughput.

Figure 21: On-the-fly modulation performance.



(a) BER. (b) PER.

Figure 22: Influence of excitation power level.

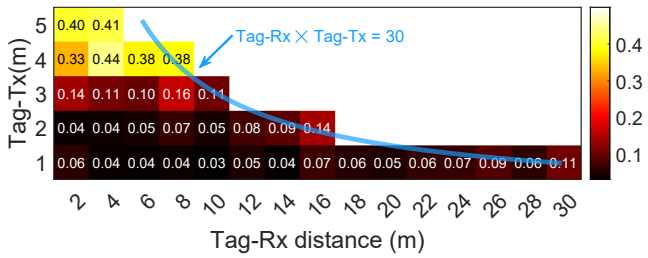


Figure 23: Two-way communication.

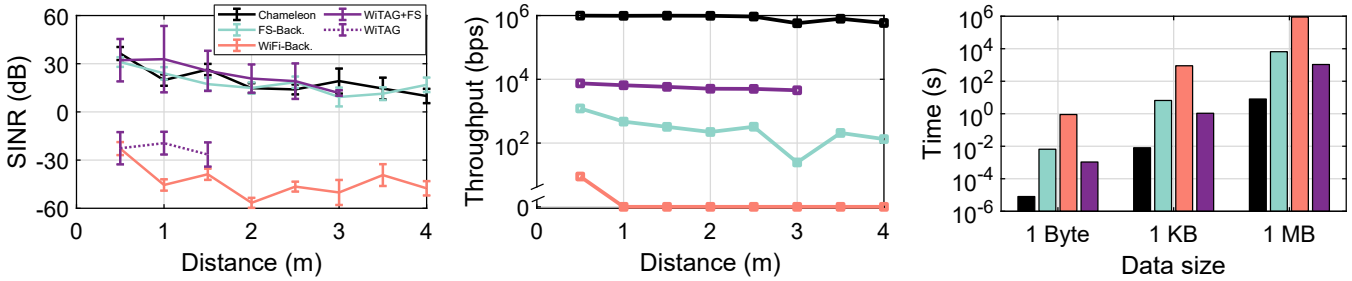
Excitation power level. We also investigate the influence of the transmitter RF signal strength. The exciter-to-tag distance and the tag-to-receiver distance are both set to 0.3 m. The BER and PER results with 18 dBm and 13 dBm excitation signal are shown in Fig. 22. As can be seen that when transmission power drops from 18 dBm to 13 dBm, the BER rises from 0.11% to 0.17% in LOS scenario. In NLOS scenario, BER rises from 3.3% to 5.51%. Similar performance degradation also occurs in the PER. That is because the weaker excitation signal means lower receiver SNR, which leads to worse BER and PER.

4.2 End-to-end performance

Coherent demodulation and modulation. The next is to combine demodulation and on-the-fly modulation as they are coherent in nature. We conduct tests at a wide range of different uplink and downlink ranges and the results are shown in Fig. 23. The supported backscatter uplink range is inversely proportional to the downlink range. When the downlink range is within 2 m, the BER gradually rises from about 5% to above 10% with increasing tag-Rx distances. However, when the downlink range exceeds 3 m, the BER is quite high even with a short tag-receiver range. From Fig. 20a it can be seen that when downlink range is more than 3m, the downlink decoding performance worsens quickly. As the on-the-fly modulation is coherent with downlink decoding, uplink BER will also be high. Furthermore, differential demodulation errors also occur in the physical packet header, this destroys necessary information for the on-the-fly modulation. Still, the concrete pillars, the ambient WiFi APs also show their influence and make the BER change irregularly.

Comparison against prior works. Even though none of the prior systems support native WiFi backscatter with random excitations, we make a comparatively loose selection criterion for our competitors: supporting content-varying excitations and a single COTS receiver at the same time. Following this criterion, TScatter [33], BackFi [6], and CAB [43] are not included because they do not support COTS WiFi receivers; Hitchhike [47], MOXcatter [50], and SubScatter [44] are not good candidates since they use two COTS receivers; Interscatter [24], Passive WiFi [30] can indeed generate native WiFi packets but they only work with CW excitations. Finally, we decide to compare Chameleon with three closest works: WiTAG[5], FS-Backscatter[49], and WiFi Backscatter[29]. For WiTAG, we use the Xilinx Zynq-7000 + AD9361 FMCOMMS3 platform to generate A-MPDU 802.11n packets. We set the HT-MCS to 7 and use a power amplifier to boost signal level to about 18dBm. The transmitter-tag distance and tag-receiver distance are set the same. 1000 random are generated every second.

We first compare the signal to interference and noise ratio (SINR) in the receiver side. WiFi backscatter and WiTAG do not frequency-shift backscattered packets into another channel to reduce self-interference. As a result, their SINR are around -30dB, which can be shown in Fig. 24a. This leads to bad communication performance. To investigate the benefit from the usage of A-MPDU in backscatter, we compare Chameleon with WiTAG + frequency shift instead (donated as "WiTAG+FS" in Fig. 24). Chameleon, WiTAG+FS, and FS-Backscatter avoid interference from excitation and realize good SINR thanks to frequency shift. Results in Fig. 24b show that Chameleon realizes a throughput of 992.4 kbps, while those for



(a) SNR and SINR. (b) Throughput. (c) Transmission time.

Figure 24: Comparison with state-of-the-art systems.

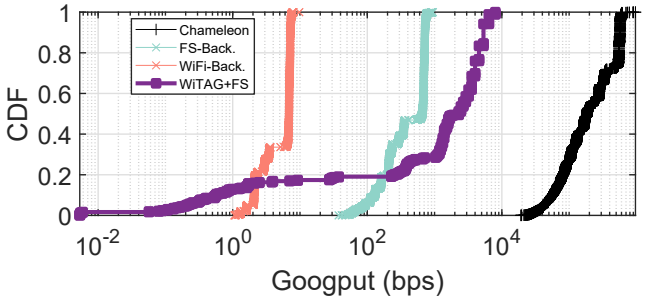


Figure 25: Goodput comparison.

WiTAG and FS-Backscatter are 8 kbps and 1 kbps, respectively. And the WiFi Backscatter only achieves a throughput of 9 bps at 0.5 m and barely works with longer ranges. For WiTAG+FS and FS-Backscatter, lost packets and wrong packets caused by ambient interference cause wrong bits. This causes a significant performance drop with increasing ranges. For example, the throughput of FS-Backscatter is less than 100 bps when the distance is over 3 m. If we use those systems to transmit data of 1 Mb, it takes Chameleon about 1 second, while FS-Backscatter, WiFi-Backscatter, and WiTAG+FS need 1000 seconds, 1000000 seconds, and 125 seconds respectively. The root cause of the above differences is that WiTAG, FS Backscatter, and WiFi Backscatter both conduct packet-level modulation. In contrast, Chameleon achieves symbol-level backscatter and supports a much higher throughput.

Furthermore, we compare the goodputs of WiTAG+FS, FS Backscatter, and WiFi-Backscatter. The excitation rate is set at 1000 pkts per second. Note that as goodput is the effective data rate at the application layer. Packets with incorrect CRCs cannot go through the MAC layer and make no contribution to it. Chameleon realizes a goodput of about 0.28 Mbps, while those for WiTAG+FS, FS-Backscatter, and WiFi-Backscatter are 2.51 kbps, 0.47 kbps, and 5.46 bps, respectively. Notice that those rates are all below their theoretical maximum values. This is possibly due to packet failures of fast rate and intensive ambient interference. It can also be seen that WiTAG+FS goodput varies from 10^{-2} bps to 8 kbps, and it is below 1 bps in about 13% cases. That's mainly because WiTAG conveys bit '0' and '1' using correct and wrong packet respectively. This means the PER of carrier packets can be seen as WiTAG BER. Ambient interference makes such transmission very fragile.

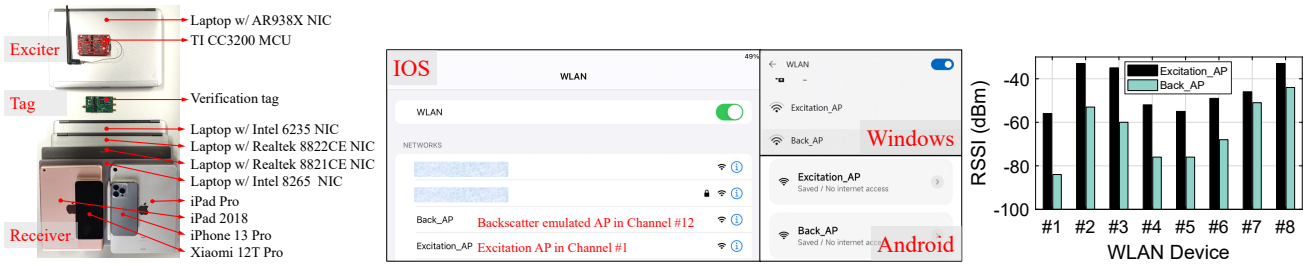
4.3 Applications

Tag as WiFi AP. With the ability to reshape WiFi excitations, we leverage Chameleon to emulate a WiFi AP, which generates a new beacon packet on top of the ambient beacon signal. Specifically, commodity radios are used to provide broadcasting beacon packets with `SSID = Excitation_AP` at WiFi channel #1. Chameleon demodulates those packets and then generates new beacon packets at channel #12 with `SSID = Back_AP` as signatures. Ideally, both excitation and backscattered beacon packets should be received at the COTS WiFi radios. The screenshots for iOS, Windows, and Android are shown in Fig. 26b, from which we observe both `Excitation_AP` and `Back_AP` signals in the network list. Our compatibility test includes 3 iOS devices (iPhone 13 Pro, iPad Pro, iPad 2018), 4 laptops equipped with COTS NICs (Intel Advanced-N 6235 NIC, Intel WirelessAC 8265, Realtek 8821CE, and Realtek 8822CE). The measured RSSIs for old and new beacon packets are shown in Fig. 26c. Those results clearly demonstrate that Chameleon supports native WiFi backscatter.

Harvesting and communication. Finally, we evaluate how Chameleon behaves with the battery-free prototype, which can harvest energy from both ambient RF and light. We configure a TI CC3200 radio to continuously generate packets at the power level of 18 dBm. Our test shows that when the tag-Tx distance is within 0.4 m, battery-free Chameleon can harvest from those excitation packets and use them for backscatter transmission simultaneously. When excitation power is -9 dBm, the needed time to harvest 0.51 mJ (the capacity of the storage capacitor) is about 22s. After energy harvesting, the tag performs demodulation and on-the-fly modulation until the power is dissipated. The working time in a charging cycle is about $0.51mJ/8.41mW = 60.9ms$. During this period, about 61 packet can be backscattered, which translates to 2.8 pkts/s. When we use a 35×70mm solar cell, it can speed up the charging period to about 6.6 s with 160 Lux office light, which increases the packet rate to 9.2 pkts/s.

5 DISCUSSION

Differential demodulation limitation. Currently, Chameleon only supports DBPSK-802.11b. This does limit Chameleon's usage.



(a) Test devices. (b) Screenshots of AP scanners for different platforms. (c) Measured RSSIs.

Figure 26: Our compatibility test includes #1: iPhone 13 Pro, #2: Xiaomi 12T Pro, #3: iPad Pro 11-inch, #4: iPad 2018, #5: Intel(R) Advanced-N 6235 NIC, #6: Intel WirelessAC 8265NIC, #7: Realtek 8821CE , #8: Realtek 8822CE. For iOS, Windows, and Android devices, the Apple Airport Utility, WiFi Analyzer and Scanner, and Cellular-Z are used to measure the beacon RSSIs.

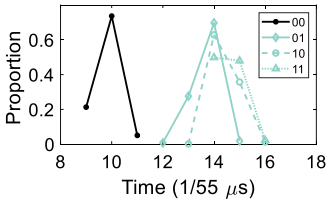


Figure 27: DQPSK-802.11b pulse width.

But it is worth noting that DBPSK-802.11b signal is widely used in WiFi system for beaconing and MAC control for its robustness. Chameleon will still have plenty of transmission chances. Still, our differential modulation design distinguishes raw bit '1' and '0' in 802.11b packets using the pulse width at symbol boundaries. Only 1 DSSS chip of all 11 ones take effect. This means at least additional 11× SNR is needed in Chameleon tag to realize similar demodulation performance in commercial WiFi devices. But note that Chameleon downlink range is also limited by the strength loss caused by backscatter reflection. If the tag is far away from the exciter, reflected signal will be too weak for the receiver. This limitation is shared in all backscatter systems and is more severe.

Demodulation of other WiFi signals. We also consider the possibility of demodulating other WiFi signals in Chameleon. The DQPSK-802.11b is closest to the DBPSK-802.11b signal, and we first investigate whether Chameleon can decode it. One DQPSK-802.11b symbol contains two bits and has four different phase states. We analyze their pulse width at symbol boundaries. Results are shown in Fig. 27. As can be seen that '01', '10', '11' have similar pulse widths, which are different from that for '00'. This means Chameleon is only able to decode part of DQPSK-802.11b bits, but the native support for it cannot be realized. CCK-802.11b and OFDM WiFi (802.11g/n) signals have more differences with DBPSK-802.11b and thus are more difficult to support in Chameleon. We plan to have more observation and try to find a solution for those signals.

6 RELATED WORK

Backscatter has been a hot topic in the wireless and networking community [7, 9, 13–19, 23, 25, 27, 28, 34, 41, 53] and Chameleon makes contribution at the following two forefronts.

Commodity backscatter. There are a great deal of prior works that study how to design backscatter systems using commodity radios as receivers, including WiFi backscatter [10, 47, 48], Bluetooth backscatter [11, 45, 46], LTE backscatter [8], ZigBee backscatter [48], and LoRa backscatter [26, 36, 38, 39]. All those works eliminate the need for dedicated hardware as exciters and receivers but fail to achieve native commodity connectivity. This is because they adopt codeword translation alike techniques to deal with content-varying ambient excitations; two receivers becomes necessary. To break this barrier, a number of single-receiver backscatter systems are proposed [32, 35, 40, 42, 43]. Unfortunately, these works either pose restrictions on excitations or require SDRs for tag data demodulation. Unlike those works, Chameleon proposes a novel on-the-fly modulation and enables native WiFi backscatter for the first time.

Ambient signal demodulation. Another key function of backscatter tags is to demodulate signals from ambient environments. Past works propose various demodulation on the packet level [47, 48, 50, 51] and symbol level [21, 31, 37]. While these approaches achieve either better throughput [21], or longer ranges, e.g., downlink range of 52 m [31], none of them can demodulate ambient WiFi signals. In contrast, Chameleon builds the first passive WiFi demodulator by turning rectifying phase-modulation signals into ASK signals.

7 CONCLUSION

We have presented Chameleon, the first native WiFi backscatter system with uncontrolled content-varying excitations. The key observation is that the low-power tag can demodulate WiFi signals in an ASK way without the need to differentiate the symbol phases. On top of that, we have designed on-the-fly modulation that can turn any excitations into a native WiFi packet. Extensive comparisons and experiments with COTS devices have demonstrated Chameleon's superior performance and strong compatibility with different kinds of WiFi devices. We envision that the Chameleon design will open the door to making battery-free WiFi connectivity ubiquitous for real-world IoT applications.

8 ACKNOWLEDGMENT

This work was supported by NSFC Grant No. 61932017 and 61971390. The authors would like to express their heartfelt appreciation for the support from the NSFC.

REFERENCES

[1] [n. d.]. https://github.com/hui811116/gr-wifi-dsss. ([n. d.]).
[2] [n. d.]. https://www.keysight.com/us/en/products/software/pathwave-design-software. ([n. d.]).
[3] [n. d.]. https://w.wol.ph/2015/08/28/maximum-wifi-transmission-power-country/. ([n. d.]).
[4] [n. d.]. Global Wi-Fi Enabled Devices Shipment Forecast. https://www.researchandmarkets.com/reports/5135535/global-wi-fi-enabled-devices-shipment-forecast. ([n. d.]).
[5] A. Abedi, F. Dehbashi, M. Mazaheri, O. Abari, and T. Brecht. 2020. Witag: Seamless wifi backscatter communication. In *Proc. of ACM SIGCOMM*.
[6] D. Bharadia, K. Joshi, M. Kotaru, and S. Katti. 2015. Backfi: High throughput wifi backscatter. In *Proc. of ACM SIGCOMM*.
[7] L. Chen, W. Hu, K. Jamieson, X. Chen, D. Fang, and J. Gummeson. 2021. Pushing the Physical Limits of IoT Devices with Programmable Metasurfaces. In *Proc. of USENIX NSDI*.
[8] Z. Chi, X. Liu, W. Wang, Y. Yao, and T. Zhu. 2020. Leveraging ambient lte traffic for ubiquitous passive communication. In *Proc. of ACM SIGCOMM*.
[9] Q. Dong, J. Wu, W. Hu, and J. Crowcroft. 2007. Practical Network Coding in Wireless Networks. In *Proc. of ACM MobiCom*.
[10] M. Dunna, M. Meng, P. Wang, C. Zhang, P. P Mercier, and D. Bharadia. 2021. SyncScatter: Enabling WiFi like synchronization and range for WiFi backscatter Communication. In *Proc. of USENIX NSDI*.
[11] J. F Ensworth and M. S Reynolds. 2017. BLE-backscatter: Ultralow-power IoT nodes compatible with Bluetooth 4.0 low energy (BLE) smartphones and tablets. *IEEE Transactions on Microwave Theory and Techniques* 65, 9 (2017), 3360–3368.
[12] N. Gershenfeld, R. Krikorian, and D. Cohen. 2004. The internet of things. *Scientific American* 291, 4 (2004), 76–81.
[13] R. Ghaffarivardavagh, S. Afzal, O. Rodriguez, and F. Adib. 2020. Ultra-Wideband Underwater Backscatter via Piezoelectric Metamaterials. In *Proc. of ACM SIGCOMM*.
[14] R. Ghaffarivardavagh, S. Afzal, O. Rodriguez, and F. Adib. 2020. Underwater Backscatter Localization: Toward a Battery-Free Underwater GPS. In *Proc. of ACM HotNets*.
[15] W. Gong, S. Chen, and J. Liu. 2017. Towards higher throughput rate adaptation for backscatter networks. In *Proc. of IEEE ICNP*.
[16] W. Gong, S. Chen, J. Liu, and Z. Wang. 2018. MobiRate: Mobility-Aware Rate Adaptation Using PHY Information for Backscatter Networks. In *Proc. of IEEE INFOCOM*.
[17] W. Gong, H. Liu, J. Liu, X. Fan, K. Liu, Q. Ma, and X. Ji. 2018. Channel-Aware Rate Adaptation for Backscatter Networks. *IEEE/ACM Transactions on Networking* (2018).
[18] W. Gong, H. Liu, K. Liu, Q. Ma, and Y. Liu. 2016. Exploiting channel diversity for rate adaptation in backscatter communication networks. In *Proc. of IEEE INFOCOM*.
[19] W. Gong, K. Liu, and Y. Liu. 2015. Directional Diagnosis for Wireless Sensor Networks. *IEEE Transactions on Parallel and Distributed Systems* (2015).
[20] W. Gong, L. Yuan, Q. Wang, and J. Zhao. 2020. Multiprotocol backscatter for personal IoT sensors. In *Proc. of ACM CoNEXT*.
[21] X. Guo, L. Shangguan, Y. He, N. Jing, J. Zhang, H. Jiang, and Y. Liu. 2022. Saiyan: Design and Implementation of a Low-power Demodulator for LoRa Backscatter Systems. In *Proc. of USENIX NSDI*.
[22] Z. Huang and W. Gong. 2022. EAScatter: Excitor-Aware Bluetooth Backscatter. In *Proc. of IEEE/ACM IWQoS*.
[23] V. Iyer, R. Nandakumar, A. Wang, S. B. Fuller, and S. Gollakota. 2019. Living IoT: A Flying Wireless Platform on Live Insects. In *Proc. of ACM MobiCom*.
[24] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith. 2016. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proc. of ACM SIGCOMM*.
[25] J. Jang and F. Adib. 2019. Underwater backscatter networking. In *Proc. of ACM SIGCOMM*.
[26] J. Jiang, Z. Xu, F. Dang, and J. Wang. 2021. Long-Range Ambient LoRa Backscatter with Parallel Decoding. In *Proc. of ACM MobiCom*.
[27] S. Jog, J. Guan, S. Madani, R. Lu, S. Gong, D. Vasisht, and H. Hassanieh. 2022. Enabling IoT Self-Localization Using Ambient 5G Signals. In *Proc. of USENIX NSDI*.
[28] S. Jog, Z. Liu, A. Franques, V. Fernando, S. Abadal, J. Torrellas, and H. Hassanieh. 2021. One Protocol to Rule Them All: Wireless Network-on-Chip using Deep Reinforcement Learning. In *Proc. of USENIX NSDI*.
[29] B. Kellogg, A. Parks, S. Gollakota, J. R Smith, and D. Wetherall. 2014. Wi-Fi backscatter: Internet connectivity for RF-powered devices. In *Proc. of ACM SIGCOMM*.
[30] B. Kellogg, V. Talla, S. Gollakota, and J. R Smith. 2016. Passive wi-fi: Bringing low power to wi-fi transmissions. In *Proc. of USENIX NSDI*.
[31] S. Li, H. Zheng, C. Zhang, Y. Song, S. Yang, M. Chen, L. Lu, and M. Li. 2022. Passive DSSS: Empowering the Downlink Communication for Backscatter Systems. In *Proc. of USENIX NSDI*.
[32] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R Smith. 2013. Ambient backscatter: Wireless communication out of thin air. *ACM SIGCOMM Computer Communication Review* 43, 4 (2013), 39–50.
[33] X. Liu, Z. Chi, W. Wang, Y. Yao, P. Hao, and T. Zhu. 2021. Verification and Redesign of OFDM Backscatter. In *Proc. of USENIX NSDI*.
[34] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. 2015. Practical, Real-Time Centralized Control for CDN-Based Live Video Delivery. In *Proc. of ACM SIGCOMM*.
[35] A. N Parks, A. Liu, S. Gollakota, and J. R Smith. 2014. Turbocharging ambient backscatter communication. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 619–630.
[36] Y. Peng, L. Shangguan, Y. Hu, Y. Qian, X. Lin, X. Chen, D. Fang, and K. Jamieson. 2018. PLoRa: A passive long-range data network from ambient LoRa transmissions. In *Proc. of ACM SIGCOMM*.
[37] M. Rostami, X. Chen, Y. Feng, K. Sundaresan, and D. Ganesan. 2021. MIXIQ: Re-Thinking Ultra-Low Power Receiver Design for next-Generation on-Body Applications. In *Proc. of ACM MobiCom*.
[38] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. Smith, and S. Gollakota. 2017. Lora backscatter: Enabling the vision of ubiquitous connectivity. In *Proc. of ACM IMWUT*.
[39] Ambuj Varshney, Carlos Pérez-Penichet, Christian Rohner, and Thiemo Voigt. 2017. LoRea: A Backscatter Architecture That Achieves a Long Communication Range. In *Proc. of ACM SenSys*.
[40] A. Wang, V. Iyer, V. Talla, J. R Smith, and S. Gollakota. 2017. {FM} backscatter: Enabling connected cities and smart fabrics. In *Proc. of USENIX NSDI*.
[41] F. Wang, J. Liu, and W. Gong. 2019. WiCAR: WiFi-based in-Car Activity Recognition with Multi-Adversarial Domain Adaptation. In *Proc. of IEEE/ACM IWQoS*.
[42] Q. Wang, S. Chen, J. Zhao, and W. Gong. 2021. RapidRider: Efficient WiFi Backscatter with Uncontrolled Ambient Signals. In *Proc. of IEEE INFOCOM*.
[43] Y. Yang, L. Yuan, J. Zhao, and W. Gong. 2022. Content-agnostic backscatter from thin air. In *Proc. of ACM MobiSys*.
[44] L. Yuan and W. Gong. 2022. SubScatter: Sub-symbol WiFi Backscatter for High Throughput. In *Proc. of IEEE ICNP*.
[45] M. Zhang, S. Chen, J. Zhao, and W. Gong. 2021. Commodity-level BLE backscatter. In *Proc. of ACM MobiSys*.
[46] M. Zhang, J. Zhao, S. Chen, and W. Gong. 2020. Reliable backscatter with commodity ble. In *Proc. of IEEE INFOCOM*.
[47] P. Zhang, D. Bharadia, K. Joshi, and S. Katti. 2016. Hitchhike: Practical backscatter using commodity wifi. In *Proc. of ACM SenSys*.
[48] P. Zhang, C. Josephson, D. Bharadia, and S. Katti. 2017. Freerider: Backscatter communication using commodity radios. In *Proc. of ACM CONEXT*.
[49] P. Zhang, M. Rostami, P. Hu, and D. Ganesan. 2016. Enabling practical backscatter communication for on-body sensors. In *Proc. of ACM SIGCOMM*.
[50] J. Zhao, W. Gong, and J. Liu. 2018. Spatial Stream Backscatter Using Commodity WiFi. In *Proc. of ACM MobiSys*.
[51] J. Zhao, W. Gong, and J. Liu. 2018. X-tandem: Towards multi-hop backscatter communication with commodity wifi. In *Proc. of ACM MobiCom*.
[52] J. Zhao, W. Gong, and J. Liu. 2020. Towards scalable backscatter sensor mesh with decodable relay and distributed excitation. In *Proc. of ACM MobiSys*.
[53] Y. Zhao, S. Afzal, W. Akbar, O. Rodriguez, F. Mo, D. Boyle, F. Adib, and H. Haddadi. 2022. Towards battery-free machine learning and inference in underwater environments. In *Proc. of ACM HotMobile*.