FreeRider: Backscatter Communication Using Commodity Radios

Pengyu Zhang[1], Colleen Josephson[1], Dinesh Bharadia[2], Sachin Katti[1]

Stanford University[1], UCSD[2]
{pyzhang,skatti}@cs.stanford.edu,cajoseph@stanford.edu,dineshb@ucsd.edu

ABSTRACT

We introduce the design and implementation of FreeRider, the first system that enables backscatter communication with multiple commodity radios, such as 802.11g/n WiFi, ZigBee, and Bluetooth, while these radios are simultaneously used for productive data communication. Furthermore, we are, to our knowledge, the first to implement and evaluate a multi-tag system. The key technique used by FreeRider is *codeword translation*, where a tag can transform a codeword present in the original excitation signal into another valid codeword from the same codebook during backscattering. In other words, the backscattered signal is still a valid WiFi, ZigBee, or Bluetooth signal. Therefore, commodity radios decode the backscattered signal and extract the tag's embedded information. More importantly, FreeRider does codeword translation regardless of the data transmitted by these radios. Therefore, these radios can still do productive data communication. FreeRider accomplishes codeword translation by modifying one or more of the three dimensions of a wireless signal — amplitude, phase and frequency. A tag ensures that the modified signal is still comprised of valid codewords that come the same codebook as the original excitation signal. We built a hardware prototype of FreeRider, and our empirical evaluations show a data rate of ~60kbps in single tag mode, 15kbps in multi-tag mode, and a backscatter communication distance up to 42m when operating on 802.11g/n WiFi.

CCS CONCEPTS

• **Networks → Network architectures**; **Wireless access networks**;

CoNEXT '17, Incheon, Republic of Korea

DOI: 10.1145/3143361.3143374

KEYWORDS

Backscatter; WiFi; ZigBee; Bluetooth; Wireless

1 INTRODUCTION

Backscatter communication is well-known for providing an ultra-low power wireless link for connecting Internet-of-Things devices. Backscatter radios only consume microwatts of power during data transmission because they passively reflect and modify wireless signals to embed information instead of doing active transmission like WiFi. Typically there are three components in a backscatter communication system: an excitation signal generator, a backscatter device (tag), and a receiver. RFID is a popular example of backscatter. In RFID, the reader acts as both the excitation signal generator *and* the receiver. This requires a dedicated infrastructure of expensive custom hardware (such as RFID readers). Hence, despite their power efficiency, we actually do not see a wide deployment of backscatter-based systems. As a result, researchers have recently been exploring using already-deployed commodity radios, such as WiFi and Bluetooth, to act as the excitation radio and/or receiver.

There are a number of research projects along these lines. Passive WiFi [16] is a pioneer which enables a tag to talk to an 802.11b WiFi receiver. A special tag transforms a sine wave signal to an 802.11b signal during backscattering. Therefore, the backscattered signal can be decoded by a commodity 802.11b WiFi receiver. One inconvenience of using [16] is that you have to buy dedicated hardware which emits the sine wave signal. A followup work, Interscatter [13], addresses this problem by hacking Bluetooth on a smartwatch. [13] forces the Bluetooth radio to transmit a specific sequence of data bits such that the signal that is emitted is a sine wave. [13] made a significant step toward using commodity radios for backscatter. However, the channel occupied by Bluetooth cannot be used for productive data communication because the Bluetooth radio itself is generating the excitation signal with no data (e.g., all zeros). This is **non-productive communication**. In summary, most recent research emulates the excitation signal either with a tone generator or commodity hardware with non productive communication [13, 16].

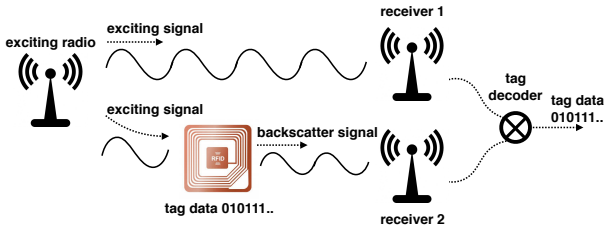


Figure 1: FreeRider system overview. An example scenario could be office setting where our smart-phone can act as exciting radio, and the WiFi APs as receiver 1 and receiver 2. The WiFi APs are connected by Ethernet backhaul which performs the decoding operation and sends tag data to the Internet.

The ISM band is crowded with numerous devices; deploying backscatter systems that rely on non-productive communication results in decreased data rates and increased congestion for wireless connectivity. Ideally, we would like the excitation signal transmitter to communicate productively with normal non-backscatter clients while simultaneously providing a signal that tags can backscatter.

To this end, a recent work HitchHike [25] enables backscatter communication using commodity 802.11b WiFi radios while the 802.11b radio itself does productive communication. However, [25] only works with 802.11b WiFi. Most modern WiFi clients use 802.11g/n where OFDM signals are transmitted. This means HitchHike devices will see little WiFi traffic they can use to backscatter. Therefore, there is a strong need to design a system that can do backscatter communication with widely deployed commodity radios (such as 802.11g/n WiFi, Bluetooth, and ZigBee) while these radios continue communicating normally with their intended clients. Since a specific type of radio wave might not be available or has bad signal quality at one particular location, the technique we invent should be general enough such that the tag can rely on multiple types of radios for backscattering its information.

This paper presents the design and implementation of FreeRider, the first system that enables backscatter communication using commodity 802.11g/n WiFi, Bluetooth, and ZigBee radios while these radios themselves are doing productive communication. Furthermore, we are the first, to our knowledge, to implement and evaluate a multi-tag system that works with commodity radios.

We overcome two key challenges in designing FreeRider. The first challenge is designing a system that can do backscatter communication with various commodity radios while they are still performing productive communication. To address this, we use a technique called codeword translation that originated in HitchHike [25] (which only works with 802.11b WiFi), and extend the technique such that it works with OFDM and other various commodity radios. The key observation here is that any wireless signal on the ISM band is generated using a set of known codewords from a fixed codebook. For example, Bluetooth uses FSK modulation and has two codewords in its codebook: it transmits a tone at one frequency to send data one, and a different frequency to send zero. Similarly, WiFi and ZigBee also have finite sets of codewords that vary in combinations of phase, amplitude or frequency. To do codeword translation, a tag transforms the ongoing excitation signal's codeword into another valid codeword in the same codebook during backscattering. This is done by modifying one or more of the amplitude, phase, and frequency of the excitation signal. The specific translation depends on the data that the tag wants to communicate and the type of the excitation signal. Because the codeword in the backscattered signal is a valid codeword from the same codebook as the original excitation signal, we can use a commodity radio to receive the backscattered signal. Figure 1 shows an overview of FreeRider. An existing ISM band radio (WiFi, Bluetooth or ZigBee) transmits data to clients as it normally would. The IoT device (which we will refer to as a tag to be consistent with prior backscatter literature) backscatters this signal and embeds the information it wants to communicate. The backscattered signal arrives at a commodity receiver that uses the same technology (WiFi, Bluetooth or ZigBee) as the transmitting radio, but sits on an adjacent channel. The intended client of the original communication decodes the original signal, and the second receiver decodes the backscattered signal. The decoded bits streams from the two receivers are compared to obtain the tag data.

The second challenge is gracefully supporting multiple tags on the same wireless channel. FreeRider addresses this issue by leveraging a technique called packet length modulation to transmit necessary information to the tags for coordination. Its core idea is using the length of the excitation packets to encode 0s and 1s, which can be arranged to form messages to the tags that implement a *backscatter MAC protocol*. This protocol sends control messages to the tags that coordinate tag transmissions to avoid collisions.

We prototyped our FreeRider tag, and used a MacBook Pro laptop as the backscatter decoder. Our empirical evaluation shows the following results:

- We can decode backscattered OFDM WiFi signals from 42m in a line-of-sight (LOS) deployment, and 22m in a non-line-of-sight (NLOS) deployment.
- We achieve a maximum throughput of ~60kbps from a backscattered OFDM WiFi signal when a LOS receiver is 18m or closer. For further distances, we achieve an average of 32kbps (LOS) and 20kbps (NLOS).
- We demonstrate that our FreeRider tag can backscatter ZigBee signals from up to 22m, achieving 15kbps. It can backscatter Bluetooth up to 12m, achieving

55kbps. We also show that our system co-exists peacefully with WiFi networks no matter what type of excitation signal the tag is backscattering.
- We evaluate FreeRider's performance with up to twenty tags and show that our MAC scheme can communicate successfully with each of the twenty tags and ensure uplink fairness among them.

2 DESIGN

2.1 Understanding backscattered signals

When a tag backscatters an excitation signal, modifies the three degrees of freedom of a signal: amplitude, phase, and frequency. Such modification is shown below where S(t) is the excitation signal, T(t) is the tag signal, and B(t) is the backscattered signal. The backscattered signal B(t) is the time domain product between the excitation signal S(t) and the tag signal T(t). Therefore, a tag changes its signal T(t) to modify the amplitude, phase, and frequency of the backscattered signal B(t).

$$\begin{aligned} S(t) &= A_s e^{j(2\pi f_s t+\theta_s)} \\ T(t) &= A_t e^{j(2\pi f_t t+\theta_t)} \\ B(t) &= S(t)T(t) \\ &= A_s e^{j(2\pi f_s t+\theta_s)} A_t e^{j(2\pi f_t t+\theta_t)} \\ &= A_s A_t e^{j(2\pi(f_s+f_t)t+(\theta_s+\theta_t))} \end{aligned} \quad (1)$$

A tag modifies the amplitude of the backscattered signal by tuning the terminating impedance of the tag antenna. The backscattered signal strength is a function of $\Gamma = \frac{Z_T - Z_A^*}{Z_A + Z_T}$ where Z_A is the tag antenna impedance and Z_T is the impedance across tag antenna terminals. The exact formulation between the backscattered signal strength and Γ can be found in [21]. In traditional backscatter systems, a tag switches between $Z_{T_1} = Z_A$ and $Z_{T_2} = 0$ to encode information. Therefore, we observe two levels of amplitude on a backscattered signal. Instead of switching between two impedances, as [21] does for creating the analog backscatter signal, our tag switches across multiple impedances to fine tune the amplitude of the backscattered signal.

A tag changes the phase of the backscattered signal by delaying the tag signal in the time domain. In order to introduce an additional phase offset $\Delta\theta$ at the tag, we delay the tag signal by $\frac{\Delta\theta}{2\pi f_t}$. Such $\Delta\theta$ phase offset introduced at the tag leads to a $\Delta\theta$ phase offset on the backscattered signal. Changing the frequency of the backscattered signal is even simpler. The tag just changes the frequency of toggling its RF transistor. Therefore, a tag is able to modify the amplitude, phase, and frequency of the backscattered signal by changing the tag signal. We next explain how a tag leverages this ability to enable backscatter communication between commodity radios.

decoded codeword	excitation signal codeword	tag bits
C_2	C_1	1
C_1	C_2	1
C_1	C_1	0
C_2	C_2	0

Table 1: Logic table between the backscatter signal codeword, excitation signal codeword, and tag data bits.

2.2 Codeword translation

The key technique for backscatter communication between a tag and commodity radios is *codeword translation*. In this section, we describe what codeword translation is and how a tag does it.

2.2.1 **Codeword translation overview.** We first define two key concepts in our system — codewords and codebooks. A codeword C_i is a signal symbol on the physical layer that represents specific data transmitted. For example, Bluetooth uses binary FSK modulation to embed information. Therefore, it only uses two codewords, $C_1 = e^{j2\pi f_1 t}$ and $C_2 = e^{j2\pi f_2 t}$, to represent data one and data zero respectively. A codebook B is the set of valid codewords used by a radio. The codebook of Bluetooth is $B = \{C_1, C_2\}$ because only two codewords are used. Similarly, 802.11g/n WiFi uses a codebook $B = \{C_1, C_2, ..., C_n\}$ where C_i is an OFDM symbol. WiFi, ZigBee and Bluetooth all use different sets of codewords and codebooks.

Our key observation is that different codewords in the same codebook are related to each other by shifts in phase, amplitude, frequency or a combination of them. For example, the codeword C_1 used by Bluetooth only differs from C_2 in the frequency domain, where a frequency difference $f_2 - f_1$ is observed.

Codeword translation is transforming a valid codeword C_i to another valid codeword C_j where both codewords belong to the same codebook, meaning $C_i \in B$ and $C_j \in B$. If such a transformation can be done by a backscatter tag in a low power fashion, then we are able to leverage WiFi, ZigBee, and Bluetooth signals for backscatter. Because the transformed codeword is still a valid codeword in the same codebook, a commodity WiFi, ZigBee, or Bluetooth radio can be used to decode the backscatter signal. The tag data is encoded by the specific codeword translations described below:

An simple example codeword translation protocol is tag summarized by Equation 2, where the codeword of the excitation signal is C_i. To encode data one, the tag translates the excitation codeword C_i to C_j before reflection. To encode data zero, the tag leaves the codeword untranslated, which means the backscatter signal has the same codeword as the exciting signal.

$$\text{backscatter codeword} = \begin{cases} C_j & \text{Tag data one} \\ C_i & \text{Tag data zero} \end{cases} \quad (2)$$

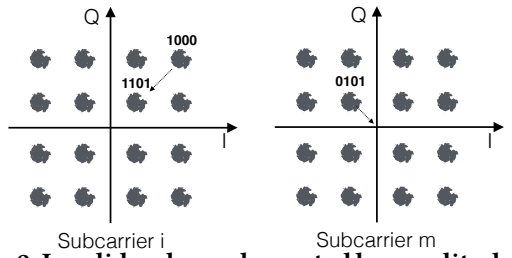


Figure 2: Invalid codewords created by amplitude modification on an OFDM signal.

By using codeword translation, we can decode the backscatter signal using commodity WiFi, ZigBee, or Bluetooth radios to extract the tag data. Table 1 shows the logic table for decoding a backscatter signal. We see that in this case the tag bits are the XOR of the backscattered codeword and the original codeword. Therefore, we can extract the tag data simply by computing the XOR of the original excitation bit-stream and the backscatter bit-stream. More complicated schemes yielding faster data rates are described in later sections.

2.2.2 **How the tag does codeword translation.** As we described before, a tag does the codeword translation by modifying the amplitude, phase, or frequency of the excitation signal. Such modification transforms the excitation codeword from C_i to C_j in the backscattered signal. When a tag does codeword translation it is frequency agnostic, meaning that the tag will apply the same modification on signals across all frequencies. This is not a problem for a signal that uses a single carrier wave, such as Bluetooth, ZigBee, and 802.11b WiFi. However, introduces issues in 802.11n WiFi because the OFDM signal used by 802.11n has multiple subcarriers. When a tag changes the amplitude of a signal on subcarrier i, it will introduce the same amplitude modification on another subcarrier m. However, the modified signal on subcarrier m might not be a valid codeword. An example of this is shown in Figure 2 where the data modulated on subcarrier i is 1000 while the data modulated on subcarrier m is 0101. When the tag transforms the signal on subcarrier i from 1000 to 1101 by reducing the signal amplitude, the tag applies the same operation on subcarrier m, reducing the strength of the signal that represents 0101. As a result, the tag creates an invalid codeword on subcarrier m. Therefore, when a tag does codeword translation, it has to look for a dimension (either amplitude, phase, or frequency) to do the signal modification such that the modified signal still has a valid codeword. We next explain how to do codeword translation for WiFi, Bluetooth, and ZigBee radios.

2.3 Backscatter with commodity radios

2.3.1 **Backscatter with OFDM WiFi.** Equation 3 shows the mathematical formulation of an OFDM modulated signal where $\{X_k\}$ are the data symbols modulated on subcarriers, N is the number of sub-carriers, and T is the OFDM symbol time. For 802.11g/n WiFi, an OFDM symbol lasts for 4μs and contains 64 subcarriers. The data symbols $\{X_k\}$ are generated using BPSK, QPSK, 16-QAM, or 64-QAM modulation depending on the WiFi bit rate.

$$S(t) = \sum_{k=0}^{N-1} X_k e^{\frac{2\pi k t}{T}} \tag{3}$$

When backscattering with OFDM WiFi, a tag cannot modify the amplitude or frequency of the excitation OFDM signal because such modification creates an invalid codeword in the backscattered signal. Therefore, the tag modifies only the phase of the backscattered signal. A binary example is shown in equation 4. The tag introduces phase offset $\Delta\theta$ to transmit data one. It introduces no offset to transmit data zero. The value of $\Delta\theta$ depends on the tag bit rate. For example, if the tag transmits at lower data rate, it uses the binary scheme where $\Delta\theta$ is 180^o. If the tag wants to transmit at higher data rate, it can choose $\Delta\theta$ as 90^o and use equation 5 to encode its information.

$$B(t) = \begin{cases} S(t)e^{j0} & \text{Tag data 0} \\ S(t)e^{j\Delta\theta} & \text{Tag data 1} \end{cases} \tag{4}$$

$$B(t) = \begin{cases} S(t)e^{j0} & \text{Tag data 00} \\ S(t)e^{j\Delta\theta} & \text{Tag data 01} \\ S(t)e^{j2\Delta\theta} & \text{Tag data 10} \\ S(t)e^{j3\Delta\theta} & \text{Tag data 11} \end{cases} \tag{5}$$

2.3.2 **Backscatter with ZigBee.** We now describe how a tag does backscatter communication with ZigBee radios. ZigBee radios use Offset QPSK (OQPSK) modulation. Similar to QPSK modulation, data is encoded in the phase of the transmitted signal. Therefore, a tag embeds data in an OQPSK signal by modifying the phase during reflection. When the tag transmits data one, it introduces a $\Delta\theta$ phase offset on the reflected signal. When the tag transmits data zero, it does not change the phase. Note that the formula for embedding tag bits in ZigBee is the same for an 802.11g/n WiFi signal (equation 4 or equation 5 depending on the tag bit rate).

2.3.3 **Backscatter with Bluetooth.** We now discuss how to backscatter with Bluetooth. A Bluetooth radio modulates information by changing the carrier signal frequency between f_1 and f_0 depending on the codeword transmitted. When the radio transmits data one, it sends a sine wave with frequency f_1. When transmitting data zero, it sends a sine wave with frequency f_0.

A FreeRider tag uses the formula shown below to embed its information. When transmitting data one, it produces an additional frequency offset Δf in the backscattered signal by toggling its RF transistor at frequency Δf. When transmitting data zero, it does not produce the additional frequency

offset. If we select Δf carefully, we can ensure that B(t) is still a valid Bluetooth signal and can be decoded by a commercial Bluetooth radio.

$$B(t) = S(t)T(t) = \begin{cases} S(t)e^{j(2\pi\Delta f t)} & \text{tag data one} \\ S(t) & \text{tag data zero} \end{cases} \quad (6)$$

One possible Δf is $|f_1 - f_0|$. Let the Bluetooth radio transmit data one with frequency f_1. For the tag to transmit data one, it shifts the signal by Δf so the backscattered codeword becomes $e^{j(2\pi f_0 t + \theta_s)}$. This is still a valid Bluetooth FSK codeword because it is a sine wave with frequency f_0. However, a commercial Bluetooth radio will decode it as zero rather than one. Conversely, to encode a data zero the tag does not frequency-shift the Bluetooth signal. The case is symmetric when the Bluetooth radio transmits data zero with frequency f_0 instead. In summary, to transmit data one a FreeRider tag transforms a Bluetooth codeword with frequency f_1/f_0 to a backscattered codeword with frequency f_0/f_1. To transmit data zero, the tag produces a backscattered codeword with the same frequency as the original Bluetooth codeword. Therefore, by choosing Δf properly, a FreeRider tag can produce a backscattered signal that is a valid Bluetooth signal while still embedding its information.

2.3.4 Avoiding interference from active radios. When a tag reflects the excitation signal to a receiver, the receiver will see severe interference from the exciting signal because both the backscattered signal and the exciting signal share the same channel [27]. To avoid such interference, FreeRider leverages techniques developed by [27] and [13] where the backscattered signal is shifted in the frequency domain to ensure that it occupies a different channel than the exciting signal. Such frequency shifting can be done by toggling the RF transistor at the desired frequency offset. For example, if we want to move the backscattered signal 20MHz away from the exciting signal, we toggle the RF transistor at 20MHz.

When backscattering a WiFi signal, our tag does the frequency shifting such that the backscattered signal sits on channel 13, which is the least used channel in the 2.4GHz ISM band. This reduces interference to and from other active radios. When backscattering Bluetooth or ZigBee, tags backscatter on channels close to 2.48GHz because these channels experience less interference from WiFi.

2.4 MAC protocol

To facilitate effective sharing of the wireless medium between multiple tags, a media access (MAC) scheme is needed. Our MAC protocol serves two purposes: it tells the tag what signals to backscatter with and allows our system to support multiple tags. Below we discuss the design in more detail; we evaluate the performance of the MAC layer in Section 4.5.

2.4.1 **Coordinating tags.**

Determining when to backscatter: If the incorrect signal is backscattered, data cannot be recovered. The tags need a way to distinguish when to start backscattering signals. To do this, the transmitter sends a *preamble* containing a predetermined sequence of 0s and 1s (we describe this further in Sec. 2.4.2). The tag maintains a circular buffer of received bits. If the beginning of the buffer matches the preamble, the tag know the buffer contains a backscatter initiate command from the transmitter and not random packets.

Communicating with multiple tags: Since a tag does not have sufficient power to do carrier sensing, we designed a random access scheme based on Framed Slotted Aloha where the transmitter acts as a central coordinator, similar to the one implemented in RFID backscatter systems[7]. Communication is done in *rounds* with a fixed number of slots per round. Each round, the tags choose a random slot to transmit. If two tags choose the same slot, there is a collision and no data is successfully transmitted. At the end of a round the transmitter processes data from the tags and adjusts the number of slots before proceeding to the next round.

Compared to a stochastically-allocated time-division scheme, random access allows the set of tags to grow and shrink without a specific association process. The number of slots is inferred by the receiver from how many packets it receives, as well as any collisions. The receiver passes this information to the transmitter. If the transmitter sees many collisions, it adds slots. It decreases the number of slots if there are many un-utilized. To avoid collisions from other users on the same channel, the transmitter uses carrier sensing before sending messages to the tags. Each round can have an arbitrary amount of delay before the next. This ensures that the backscatter system does not hog the channel.

The use of rounds allows for fairness between the backscatter system and other users of the channel. The use of slots within the backscatter system allows for fairness between tags. Below we present the low-power transmitter-to-tag communication system needed to operate our MAC.

2.4.2 **Transmitting coordination messages to tags.** One simple solution to enable communication from the transmitter to the tags is using a commodity 802.11 decoder in each tag. However, the decoder implements complex signal processing, which is power-intensive. The challenge is designing a transmitter-to-tag communication system that is low power, i.e. does not require the tags to decode packets. Our solution is a low-power overlay scheme that sits on top of an existing system and can be implemented with commodity hardware.

Envelope detectors are a low-power component that can be exploited to implement a transmitter-to-tag communication system. Low-power envelope detectors consumes less

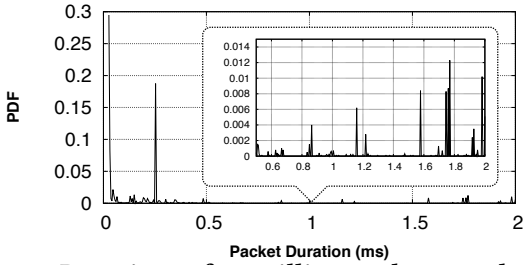


Figure 3: Durations of 30 million packets on ch. 6 collected in a lecture hall. We see a bimodal distribution where ~78% of packets last less than 500μs and ~18% last 1500μs-2700μs. With a pulse-width error bound of 25μs, the probability an ambient packet having the same length as our pulses is about 0.03%.

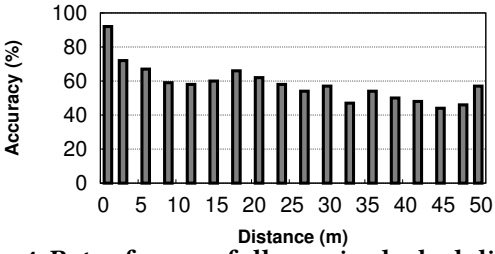


Figure 4: Rate of successfully received scheduling messages vs distance transmitting at 15dBm.

than 1μW [20]. To create our transmitter-to-tag communication system, we need parameters that can be easily measured using the envelope detector, and can also be modulated at the transmitter using commodity hardware. Two possibilities are implementing amplitude modulation by varying transmission strength, or doing on on-off keying. Packet amplitudes cannot be easily controlled on a per-packet basis with commodity hardware, and furthermore is not robust at long distances due to low SNR. On-off keying works at longer distances, but does not work well with the presence of ambient network traffic.

Packet length modulation (PLM): To overcome those limitations, we used packet length modulation: packet duration is easy for the transmitter to control, works well at a range of distances, and is robust in the presence of ambient network traffic. PLM is similar to the modulation scheme used in [13]. In our scheme, a 0 bit is represented by packets of duration L_0 and a 1 bit by L_1. To control the length of the packet, our transmitter sends packets of pre-defined durations. The tag uses an envelope detector to identify the presence of a packet and measure the duration. If a pulse duration equals L_0 or L_1 (within an error bound) a bit is recorded to a buffer. This is much more robust in low-SNR environments than amplitude modulation because the length of a packet does not degrade with distance. If a pulse has a duration besides L_0 or L_1 it is treated as noise and ignored. This allows bits to be received successfully in the presence of other transmissions. Our prototype implementation on WiFi operates at approximately 500bps, which is sufficient for operating the MAC layer.

To send the scheduling messages, the transmitter could generate dummy packets, but a better way is to buffer existing traffic before sending it to the NIC, and then re-order or re-packetize to get the necessary sequence of L_0s and L_1s. This way, as long as the network is busy, the backscatter messages impose negligible overhead on the rest of the channel.

Figure 4 shows the rate of decoding success vs distance. The experiments were performed in a long hallway inside an office building. For a reference voltage of 1.8v, the system is able to successfully decode the scheduling messages with over 70% accuracy when the tag is less than 4m away from the transmitter, and can successfully decode preambles with about 50% a distance of 50m. Due to increased SNR, higher accuracy is possible at close proximity by increasing the reference voltage in the comparator.

3 IMPLEMENTATION

We built a prototype of our system using off-the-shelf commodity 802.11g/n WiFi, ZigBee, and Bluetooth transceivers and a customized backscatter tag. We describe the implementation details below.

3.1 Hardware platform

802.11g/n WiFi transceiver: Our 802.11g/n receiver is a MacBook Pro laptop with a Broadcom BCM43xx WiFi card that supports 802.11a/b/g/n/ac. We put the WiFi card into monitor mode to report packets with bad checksums to our software. After receiving the packets, we use tcpdump to parse them and extract the tag bits.

We use an Intel 5300 WiFi card on an Intel NUC as the standard 802.11g/n OFDM transmitter (15 dBm). We use the firmware provided by [10] to control the rate of 802.11g/n packets transmission.

ZigBee transceiver: We use TI CC2650 [4] as the ZigBee transceiver and set the transmission power to 5dBm, which is the maximum power allowed by this radio. The CC2650 radio dev board CC2650EM-7ID supports two types of antennas: a PCB on-board antenna and an antenna with an SMA interface. We use VERT2450 antenna because the beam width is wide; it is mounted [2] on the SMA interface.

Bluetooth transceiver: We use TI CC2541 [3] as the Bluetooth transceiver. This radio transmits at 1Mbps and 0 dBm using FSK modulation with a frequency deviation of 250kHz and a bandwidth of 1MHz. The modulation index used is 0.5 ± 0.01.

FreeRider tag: Figure 5 shows a hardware prototype of the FreeRider tag. The tag has two VERT2450 [2] antennas,

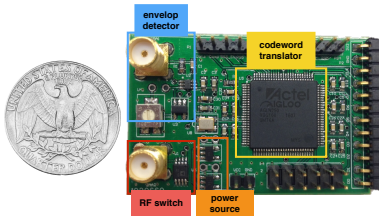


Figure 5: FreeRider tag hardware prototype.

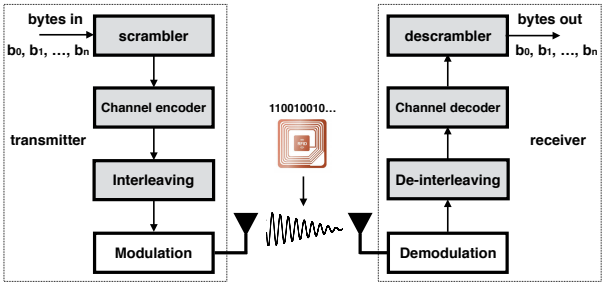


Figure 6: Transmission and reception block diagrams used in 802.11g/n.

one for reception and one for transmission. The reception antenna is connected to an LT5534 envelope detector, which measures when an incoming signal starts. We measured a 0.35us delay between the starting point of an exciting signal and the indicator signal from the envelope detector. In other words, 0.35μs after the excitation signal actually arrives, the envelope detector will tell the processor that the excitation signal has begun. In our evaluations, the performance does not degrade when experiencing a 0.35μs delay.

The other antenna is controlled by an ADG902 RF switch, which decides when and how to backscatter the exciting signal. The codeword translation module is implemented in a low-power FPGA AGLN250. We also have a power management module on the tag which provides 1.5V and 3.3V to the rest of the system.

Open source FreeRider platform: The source code of the FreeRider platform is available on [**?**] under an academic license to ensure reproducibility of results.

3.2 Implementation challenges

Each radio comes with its own physical layer stack with a specific set of channel codes, interleaving techniques and scrambling algorithms, all of which can interfere with codeword translation and render it ineffective. We discuss how to enable codeword translation despite these challenges.

3.2.1 **Challenges backscattering OFDM WiFi.** Figure 6 shows three factors that could cause trouble when decoding a backscattered WiFi signal: the scrambler, the convolutional channel encoder, and interleaving. The scrambler is a data whitening engine where it takes the input data and XORs it

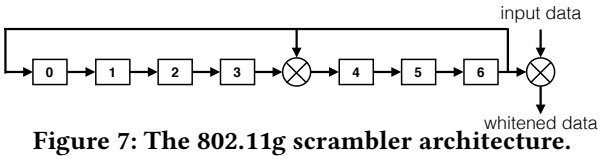


Figure 7: The 802.11g scrambler architecture.

with a pseudorandom sequence. A scrambler ensures that the data transmitted is not all zeros nor all ones, which causes a bad peak-to-average ratio. The channel encoder uses convolutional encoding to improve its robustness over wireless transmission. The interleaving engine re-orders the transmitted bits sequence to ensure that even a bursty error on wireless channel does not cause a burst of continuous errors on the received data. We consider these three modules because they are placed before the modulator in an 802.11g/n transmitter. We explain why such placement could cause unsuccessful backscatter decoding next.

For any input sequence $b_0, b_1, ..., b_n$, the transmitted signal S(t) can be formulated as $S(t) = f(b_0, b_1, ..., b_n)$ where $f()$ represents the operations introduced by the scrambler, channel encoder, interleaver, and modulator. Since the corresponding demodulator, de-interleaver, channel decoder, and descrambler in the receiver provide the reverse operations $f^{-1}()$, the receiver is able to decode and output the transmitted sequence.

Unfortunately, when a FreeRider tag is present and produces a signal $g(t_0, t_1, ..., t_n)$ using tag bits $t_0, t_1, ..., t_n$, the backscattered signal $B(t)$ becomes the time-domain product between the tag signal and the excitation signal. To see why this is a problem, let us examine the binary case shown in equation 7. This does not look like a signal that is generated by XOR of the exciting signal bits and the tag bits and passed through $f()$. Therefore, decoding the tag bits becomes hard.

$$\begin{aligned} B(t) &= S(t)T(t) \\ &= f(b_0, b_1, ..., b_n) \times g(t_0, t_1, ..., t_n) \\ &! = f(b_0 \oplus t_0, b_1 \oplus t_1, ..., b_n \oplus t_n) \end{aligned} \tag{7}$$

One insight to solve this problem is redundancy, i.e map one tag bit to multiple 802.11g/n bits. Instead of directly transmitting $t_0, t_1, ..., t_n$, the tag actually transmits a sequence where a tag repeats each bit multiple times before switching to the next one. We look at each module in the TX/RX chain and understand why redundancy helps solve this problem.

The first module is the interleaving engine which interleaves the data assigned to each subcarrier. Interleaving is done per OFDM symbol [1]. In other words, the interleaving engine will not interleave data belonging to two OFDM symbols. Therefore, as long as we can ensure that the tag bit duration is longer than an OFDM symbol, the interleaving engine will not cause troubles.

Now we turn to the other two modules, the scrambler and channel encoder. Both modules produce and maintain a deterministic structure of the data fed into the modulator.

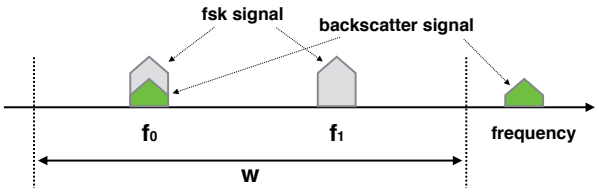


Figure 8: A FreeRider tag embeds its information on a Bluetooth signal.

The scrambler uses the structure shown in Figure 7 to do data whitening. Even when the input is all zeros, the actual data transmitted is a non-zero sequence. Such data whitening reduces the peak-to-average power ratio in the RF front end. The mathematical expression of the scrambler is shown in equation 8.

$$C[k] = b[k] \oplus b[k-3] \oplus b[k-7] \tag{8}$$

The channel encoder uses Equation 9 to encode the data at 6Mbps where $b[k]$ is the input bit and $C_1[k]$ and $C_2[k]$ are the codewords generated using a 1/2 coding rate. For other bit rates, the channel encoder is different. The data injected by the tag will corrupt the structures created by both modules and make backscatter decoding hard. How do we decode the tag data given the presence of these two modules?

$$\begin{aligned} C_1[k] &= b[k] \oplus b[k-2] \oplus b[k-3] \oplus b[k-5] \oplus b[k-6] \\ C_2[k] &= b[k] \oplus b[k-1] \oplus b[k-2] \oplus b[k-3] \oplus b[k-6] \end{aligned} \tag{9}$$

We simulated the two modules in Matlab and found that as long as a tag injects one bit tag data on four OFDM symbols (96 WiFi bits in 6Mbps data rate), we are able to obtain around $1e^{-3}$ bit error rate in decoding the tag bits. This is because there is a one-to-one mapping between the input sequence $b[k]$ and the output of the two modules $C[k]$ or $\{C_1[k], C_2[k]\}$. Equation 8 and equation 9 show that the sequence of $\{b[k] \oplus 1, b[k-1] \oplus 1, ..., b[k-7] \oplus 1\}$ can generate $C[k] \oplus 1$ and $\{C_1[k] \oplus 1, C_2[k] \oplus 1\}$. Therefore, when the tag does codeword translation and converts $C[k]$ and $\{C_1[k], C_2[k]\}$ to $C[k] \oplus 1$ and $\{C_1[k] \oplus 1, C_2[k] \oplus 1\}$, the corresponding modules at the receiver should output $\{b[k] \oplus 1, b[k-1] \oplus 1, ..., b[k-7] \oplus 1\}$. We prove this via empirical Matlab simulation and real system implementation with a MacBook Pro laptop as the backscatter decoder.

The last factor that could impact backscatter decoding is the pilot tone. Pilot tones in an OFDM symbol are used for correcting the phase error. Such phase error correction could remove the additional phase offset introduced by a tag, and render incorrect tag data decoding. Fortunately, many WiFi chips, such as Broadcom BCM43xx, do not use pilot tones for phase error correction. Therefore, we are often still able to decode the backscattered tag data.

3.2.2 **Challenges backscattering ZigBee.** ZigBee uses OQPSK modulation where there is a constant time-domain offset (half a bit) between the in-phase signal and the quadrature signal. Such offset is introduced for reducing the signal Peak-to-Average Power Ratio (PAPR) by avoiding the 180^o phase transition between neighboring bits. If the tag introduces a 180^o phase transition between neighboring bits in the backscattered ZigBee, it could damage the OQPSK signal structure and cause trouble decoding.

One approach of solving this problem is embedding one tag bit to multiple (N) OQPSK symbols. When a tag transmits data one, instead of introducing the 180^ophase offset on a *single* OQPSK symbol, the tag actually introduces the *same* 180^o additional phase offset on N OQPSK symbols. The first tag-modified OQPSK symbol might be incorrectly decoded by a commercial ZigBee decoder because of the potential OQPSK signal structure violation discussed above. However, the following $N-1$ tag-modified OQPSK symbols can be correctly decoded because the structure of OQPSK signals is maintained. Therefore, as long as we choose a large N, we can embed information in ZigBee traffic. We found that in practice a value of $N = 8$ is sufficient.

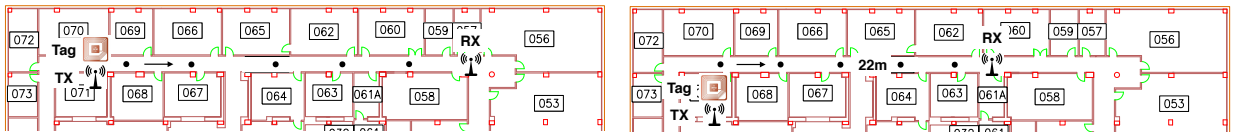
3.2.3 **Challenges backscattering Bluetooth.** There are two factors, modulation index i, and channel bandwidth w, that could cause trouble when decoding the backscattered Bluetooth signal. Modulation index i is defined as $\frac{f_1 - f_0}{w}$ and represents the ratio between the frequency deviation of an FSK signal and the bandwidth it occupies. A commercial Bluetooth radio, such as [3], usually uses a modulation index 0.5. When a FreeRider tag toggles its RF transistor at Δf, while producing the desired backscattered signal, it also produces an undesired signal on the other side of the spectrum as shown in Figure 8. This is because the backscattered signal is the time-domain product between the Bluetooth signal and the tag signal. Therefore a double-sideband backscatter signal is produced. We cannot use the single-sideband backscatter technique proposed by [13] to eliminate the undesired signal because we do not know which side (left or right) of the backscattered signal is undesired.

Fortunately, we can eliminate the undesired backscattered signal by leveraging the fact that a Bluetooth radio treats signals outside of a channel as interference and is able to eliminate them. Therefore, the selection of Δf also needs to satisfy the following two conditions, which ensure that the undesired signal sits outside of the backscatter channel and will be eliminated:

$$\begin{cases} f_1 + \Delta f > f_1 + (1-i)\frac{w}{2} & \text{FSK radio data one} \\ f_0 - \Delta f < f_0 - (1-i)\frac{w}{2} & \text{FSK radio data zero} \end{cases} \tag{10}$$

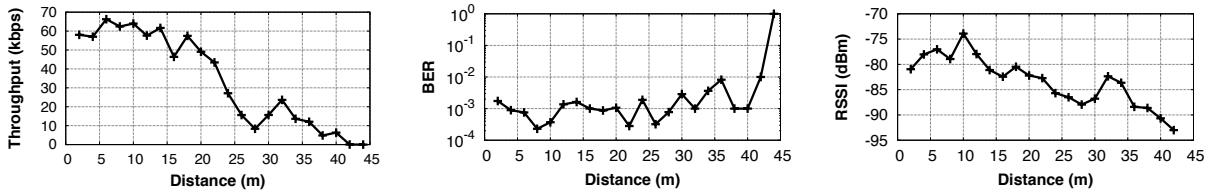
3.3 Low-power tag design

Another challenge is tag power consumption. Traditional RFID tags run a low speed clock and consume several microwatts of power. When we incorporate 20MHz frequency shifting on the tag, how much power does it consume? If we



(a) LOS deployment (b) NLOS deployment

Figure 9: Floor plan and experimental setup of our system for LOS and NLOS deployments.



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

Figure 10: Backscatter throughput, BER, and RSSI across distances in the WiFi LOS deployment.

directly take an off-the-shelf 20MHz oscillator to drive the system, the tag consumes several milliwatts of power [13] and is not low power. [27] and [13] propose the design of a low-power tag that is able to do such frequency shifting. [27] uses a ring oscillator to produce a 20MHz frequency shifting while only consuming 20μW of power. [13] designs a low-power phase-lock loop that is able to produce a 36MHz frequency shifting while only consuming 28μW of power. FreeRider takes the same ring oscillator design as [27] and produces the square wave signal needed for achieving the desired frequency shifting. We simulated using the TSMC 65nm technology, and the overall power consumption of the FreeRider tag is around 30μW depending on the type of the excitation signal. Most of the power (19μW) is consumed by the 20MHz clock needed for frequency shifting. 12μW is needed for operating the RF switch and 1~3μW is needed for running the control logic which determines the type of codeword translator to run. In the next section we present an empirical evaluation and analyze the results.

4 EVALUATION

4.1 Experimental setup

Figure 9(a) and Figure 9(b) show the experimental setup of our system. We deployed a FreeRider tag 1m away from an exciting signal transmitter (802.11g/n WiFi, ZigBee, or Bluetooth). We use these commodity radios with no hardware modifications or amplifiers. Then, we move the receiver away from the tag and measure FreeRider's throughput, bit error rate (BER), and received signal strength indicator (RSSI). We conducted both line-of-sight (LOS) and non-line-of-sight (NLOS) experiments. In the LOS experiments, all devices are placed in a hallway. In the NLOS experiments, the transmitter and the tag are deployed in a room while the receiver is deployed in a hallway. In the NLOS deployment, the backscattered signal passes through multiple walls.

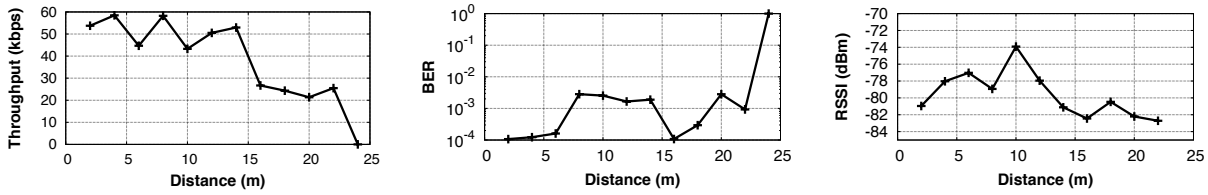
4.2 FreeRider's performance

4.2.1 Backscatter with 802.11g/n WiFi.

LOS deployment: Figure 10(a) shows the throughput of our system with increasing distance in LOS deployment. The 802.11g/n WiFi transmitter sends its OFDM signal at 11dBm, which is the maximum allowed by our hardware. We see that the receiver is still able to decode the backscattered signal at 42m, 1.4× longer than the maximum distance reported by Passive WiFi [16] and Inter-Technology Backscatter [13], and 8.4× longer than the maximum distance achieved by FS-Backscatter [27]. Such long communication distance is sufficient for many Internet-of-Things applications.

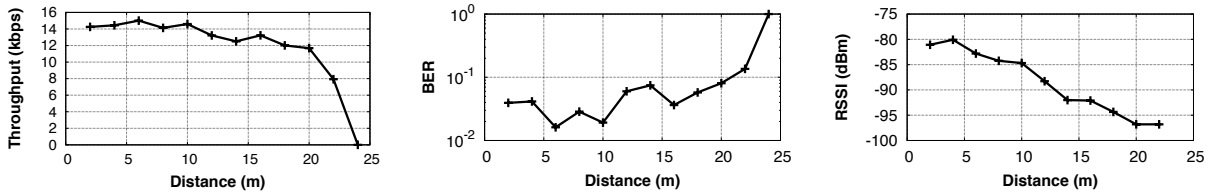
Our system achieves ~60kbps data rate when the receiver is less than 18m away from the tag. When the receiver moves farther to ~26m-36m, throughput degrades to ~15kbps. This is a lower data rate than [25] because OFDM symbols are longer in duration than DSSS symbols. One interesting observation here is that the bit error rate remains low even at longer distances as shown in Figure 10(b) despite the fact that RSSI does degrade across distance as shown in Figure 10(c). For example, we still obtain $1e^{-3}$ BER when the receiver is 40m away from the tag. This observation tells us that at longer distances, if a backscattered packet reaches the receiver, then it is very likely that we are able to extract the tag bits with low BER. If the header itself is not decoded, then we observe packet loss and lower throughput.

NLOS deployment: We also evaluated the performance of our system in a non-line-of-sight deployment where both the 802.11g/n transmitter and the tag are deployed in a room while the receiver moves away in a hallway. Figure 11(a) shows the throughput of our system in the NLOS deployment. The receiver is still able to receive the backscattered packets when it is 22m away from the tag. Similar to the LOS deployment, we achieve ~60kbps data rate when the



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

Figure 11: Backscatter throughput, BER, and RSSI across distances in the WiFi NLOS deployment.



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

Figure 12: Backscatter throughput, BER, and RSSI across distances in the ZigBee LOS deployment.

receiver is less than 14m away from the tag. At longer distances, backscatter throughput degrades to ∼20kbps.

Figure 11(b) shows the BER of our system in the NLOS deployment. Similar to the LOS deployment, we achieve a low BER across distances. However, backscatter communication stops at 22m even though we obtain an RSSI of -84dBm at 22m as shown in Figure 11(c). We find that when the receiver is more than 22m away from the tag, the backscattered signal actually needs to pass one more wall before reaching the receiver as shown in Figure 9(b). As a result, the signal becomes too weak and the packet header cannot be detected.

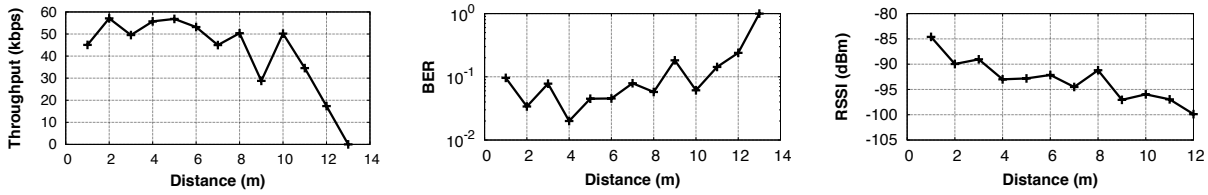
4.2.2 **Backscatter with ZigBee.** We also evaluated the performance of our system when the exciting signal is a 5dBm ZigBee signal. Figure 12(a) shows the throughput of our system when the ZigBee receiver moves away from the tag. The receiver receives backscattered packets from up to 22m away. Figure 12(c) shows that the received signal strength degrades to -97dBm at 22m, close to the noise floor of the ZigBee radio. Therefore, receiving the backscattered packets at longer distances becomes hard. We achieve a ∼14kbps backscatter data rate when the receiver is less than 12m away from the tag. At farther distances, the throughput degradation is not severe. We still obtain a 12kbps data rate at a distance of 20m. The bit error rate we achieve is $\sim 5e^{-2}$ across all distances, higher than the case when the exciting signal is 802.11g/n WiFi.

4.2.3 **Backscatter with Bluetooth.** Finally, we evaluated the performance of our system when the exciting signal is a 0dBm Bluetooth transmitter. Figure 13(a) shows the throughput of our system when the receiver moves away from the tag. We see that the receiver decodes backscatter packets up to 12m. Figure 13(c) shows that the backscatter signal has a strength of -100dBm at 12m, close to the noise floor. Therefore, decoding the backscattered packets at farther distance becomes hard. When the receiver is less than 10m away from the tag, we achieve a ∼50kbps data rate. Throughput degrades to 19kbps at 12m and the BER increases to 0.23.

4.3 Impact of TX-to-tag distance

We also evaluated how the transmitter-to-tag distance impacts the communication range of our system. In this experiment, we varied the transmitter-to-tag distance and measured the maximum receiver-to-tag distance where backscatter communication can be sustained. Figure 14 shows the results. When backscattering 802.11g/n WiFi, at a transmitter-to-tag distance of 4m, the maximum receiver-to-tag distance is 8m. This is less than the 42m achievable when the transmitter-to-tag distance is 1m. Decreasing the receiver-to-tag communication distance yields a slight increase in the achievable transmitter-to-tag distance. The operational regime of our system is shown in the grey area in Figure 14.

When a ZigBee or Bluetooth radio is used, both the transmitter-to-tag distance and the receiver-to-tag distance become shorter. The maximum transmitter-to-tag distance is 2m and 1.5m for ZigBee and Bluetooth radios respectively, and the corresponding operational regime of our system is marked with green and purple in Figure 14. Both regimes are smaller compared to the case when an 802.11g/n WiFi is used primarily because the transmission power of the ZigBee and Bluetooth radios is lower (15dBm vs 5dBm and0dBm).



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

Figure 13: Backscatter throughput, BER, and RSSI across distances in the Bluetooth LOS deployment.

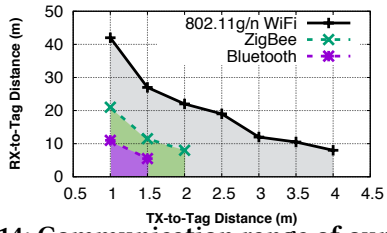


Figure 14: Communication range of our system.

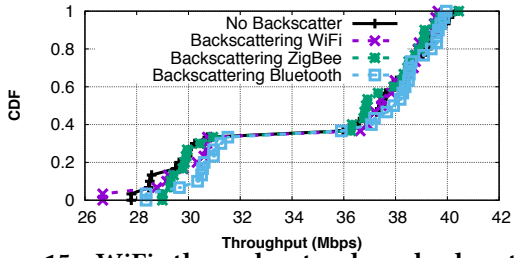


Figure 15: WiFi throughput when backscatter is present and absent.

4.4 Co-existence with WiFi networks

Next, we see if the FreeRider system can co-exist with existing WiFi networks. In this experiment, we generated artificial WiFi traffic where a laptop transfers files via WiFi on channel 6 (2.437GHz). Then, we run backscatter on ~2.472-2.48GHz (the exact frequency depends on the type of the exciting signal). We measure how the WiFi traffic and backscatter impact each other when the backscatter channel does not conflict with the WiFi channel.

4.4.1 **Does backscatter impact WiFi?** Figure 15 shows the WiFi throughput when the backscatter tag is present or absent. When backscatter is absent, WiFi is able to transmit with a 37.4Mbps median data rate. Then, we place a backscatter tag 1m away from the WiFi receiver and measure the WiFi throughput. The tag runs three codeword translators sequentially, one for backscattering 802.11g/n WiFi, one for ZigBee, and one for Bluetooth. The median WiFi throughput measured is 37Mbps, 37.9Mbps, and 36.8Mbps respectively, close to the WiFi throughput when the backscatter tag is not present. Therefore, a backscatter tag does not cause interference on an existing WiFi traffic.

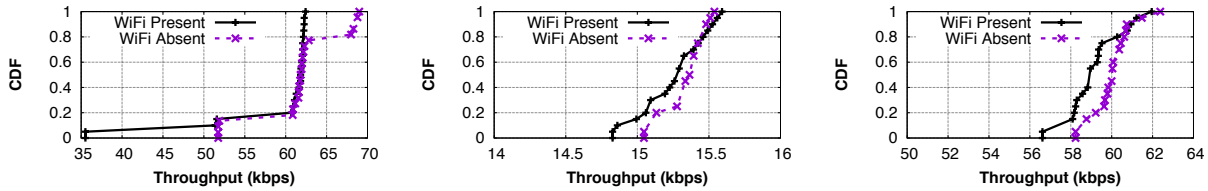
4.4.2 **Does WiFi impact backscatter?** Now we turn to the other case to see whether or not concurrent WiFi traffic impacts backscatter decoding. When a tag backscatters its data in a channel that is occupied by WiFi traffic, backscatter throughput degrades to zero because the WiFi traffic is usually ~30dB higher than the backscattered signal [27]. Therefore, backscatter suffers. In this experiment, we focus on the case where existing WiFi traffic does not share the same channel as backscatter, and try to understand how backscatter performs in the presence of WiFi traffic on adjacent channels.

Figure 16(a) shows the backscatter throughput when an 802.11g/n WiFi is used as the exciting signal, the tag backscatters on channel 13 (2.472GHz), and the WiFi traffic runs on channel 6 (2.437GHz). When the WiFi traffic is absent, we achieve 61.8kbps median backscatter throughput. When the WiFi traffic is present, the backscatter throughput is still 61.8kbps. However, we can see that backscatter is able to reach 68kbps for 20% of the time when the WiFi traffic is absent, and degrades to 35kbps for 10% of the time when the WiFi traffic is present. Therefore, the presence of WiFi traffic does impact the backscatter throughput. To minimize this impact, we can use a technique similar to the one in [25] and use RTS-CTS to reserve the channel for backscatter.

Figure 16(b) and Figure 16(c) show the backscatter throughput when a tag backscatters a ZigBee signal and a Bluetooth signal respectively. In both experiments, the tag backscatters on channel 2.48GHz. We see that the backscatter throughput difference between WiFi traffic being present or not is only 1~2kbps. Therefore, the existing WiFi traffic does not impact the backscatter performance when a ZigBee or Bluetooth radio is leveraged. One reason is because both radios are narrowband, and therefore, have better performance in filtering out-of-band interference.

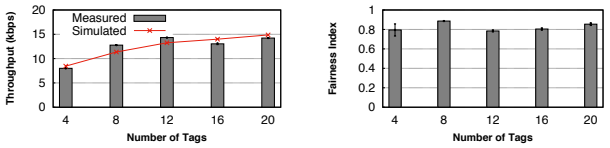
4.5 Evaluating MAC layer performance

We look at the performance of our system when communicating with multiple tags. Figure 17(a) shows the aggregated throughput when we place 4, 8, 12, 16, and 20 tags in front of the transmitter. The aggregate throughput is lower than the single tag case for two reasons: control overhead and collisions. As the number of tags increases the aggregated throughput increases. This is due to the relative ratio of control overhead decreasing when there are more transmission slots used. If we extend our simulation beyond the 20 tags



(a) Backscattering 802.11g/n WiFi signals (b) Backscattering ZigBee signals (c) Backscattering Bluetooth signals

Figure 16: FreeRider backscatter throughput when a WiFi traffic is present or absent.



(a) Aggregated throughput (b) Jain's fairness index

Figure 17: FreeRider throughput and Jain's fairness index when communicating with 4, 8, 12, 16, and 20 tags.

we had physically available, the throughput asymptotes at about 18kbps. If there are no collisions (i.e. a TDM scheme), the simulation throughput asymptotes at about 40kbps.

Framed Slotted Aloha is well-suited to applications that have low data needs and where the number of active tags can increase or decrease without warning, such as inventory tracking. More data-intensive applications would benefit from a time division scheme, which would be possible to implement in FreeRider, but we limited the analysis to a single MAC layer design for space reasons.

Figure 17(b) shows the Jain's fairness index [14] when 4, 8, 12, 16, and 20 tags are present. When the number of tags increases, the fairness index stays about the same because the scheduler dynamically allocates a larger number of slots when more tags are present. The averaged fairness index is 0.85 when 20 tags are present, which suggests that most of tags still obtain similar opportunities for data transmission.

5 RELATED WORK

The range and throughput of backscatter systems has improved in recent years [5, 6, 9, 11, 12, 17–19, 22–24, 26]. Despite the improvements, one remaining bottleneck of deploying backscatter systems is the lack of backscatter reader infrastructure. Therefore, researchers have been looking for opportunities to leverage commodity radios, such as WiFi and Bluetooth. [15] embeds backscatter information on top of WiFi traffic by changing the strength of the reflected signal. Due to the interference from the WiFi transmitter, [15] achieves relatively low data rate and short range. [8] and [16] modulate a Bluetooth and an 802.11b baseband signal respectively on a single tone signal sent by an emitter. Both enable backscatter communication with commodity WiFi and Bluetooth receivers at the cost of deploying a dedicated single tone emitter. [13] removes the single tone emitter and uses a Bluetooth radio to transmit a specific data sequence to generate the tone signal. Therefore, the Bluetooth radio itself cannot be used for productive data communication. [27] enables WiFi-to-WiFi and Bluetooth-to-Bluetooth backscatter communication within a short range. [25] enables backscatter communication between two 802.11b WiFi radios. However, none of these work with widely deployed commodity radios, such as 802.11n WiFi and Bluetooth, while the excitation signal transmitter is doing productive data communication. [25] relies on 802.11b signals for backscatter. However, such signals do not naturally exist in many locations because most WiFi clients run 802.11n/ac with OFDM signals whenever possible. Additionally, [25] does not support communication with multiple tags. [6] requires hardware modifications on commodity WiFi access points. [8] and [16] require an additional single tone emitter. [13] require specific excitation signals, and therefore, the excitation signal transmitter cannot be used for productive data communication. Our system is the first that works with popular 802.11n WiFi and Bluetooth radios while the excitation signal transmitter is used for productive data communication.

6 CONCLUSION

FreeRider, in our opinion, is the first system that enables us to leverage widely-used commodity wireless hardware for deploying backscatter. FreeRider is also the first system that implements and evaluates more than one tag communicating with the commodity receiver. Unlike previous work, the excitation signal can be used for productive data communication, which means that we can deploy backscatter devices at any location where these commodity radios are available. FreeRider does so by using a technique called codeword translation, which embeds a tag's information in the backscattered signal while ensuring that the backscattered signal can be decoded by a commodity wireless radio. FreeRider provides the opportunity to deploy backscatter systems using existing wireless infrastructure, and enables simple ultra-low power wireless connectivity for IoT devices.

ACKNOWLEDGMENT

We thank the shepherd Kyle Jamieson and the anonymous reviewers for their insightful comments.

REFERENCES

[1] [n. d.]. 802.11 Reference Design: PHY. https://warpproject.org/trac/wiki/802.11/PHY.

[2] [n. d.]. Ettus Research VERT2450 Antenna. https://www.ettus.com/product/details/VERT2450.

[3] [n. d.]. TI CC2541 radio. http://www.ti.com/product/CC2541.

[4] [n. d.]. TI CC2650 radio. http://www.ti.com/product/CC2650.

[5] Omid Abari, Deepak Vasisht, Dina Katabi, and Anantha Chandrakasan. 2015. Caraoke: An e-toll transponder network for smart cities. In *ACM SIGCOMM Computer Communication Review*, Vol. 45. ACM, 297–310.

[6] Dinesh Bharadia, Kiran Raj Joshi, Manikanta Kotaru, and Sachin Katti. 2015. BackFi: High Throughput WiFi Backscatter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 283–296.

[7] Michael Buettner and David Wetherall. 2008. An empirical study of UHF RFID performance. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*. ACM, 223–234.

[8] Joshua F Ensworth and Matthew S Reynolds. 2015. Every smart phone is a backscatter reader: Modulated backscatter compatibility with bluetooth 4.0 low energy (ble) devices. In *RFID (RFID), 2015 IEEE International Conference on*. IEEE, 78–85.

[9] Jeremy Gummeson, Pengyu Zhang, and Deepak Ganesan. 2012. Flit: a bulk transmission protocol for rfid-scale sensors. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 71–84.

[10] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool release: gathering 802.11 n traces with channel state information. *ACM SIGCOMM Computer Communication Review* 41, 1 (2011), 53–53.

[11] Pan Hu, Pengyu Zhang, and Deepak Ganesan. 2015. Laissez-Faire: Fully asymmetric backscatter communication. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 255–267.

[12] Pan Hu, Pengyu Zhang, Mohammad Rostami, and Deepak Ganesan. 2016. Braidio: An integrated active-passive radio for mobile devices with asymmetric energy budgets. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 384–397.

[13] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. 2016. Inter-Technology Backscatter: Towards Internet Connectivity for Implanted Devices. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 356–369.

[14] Raj Jain, Arjan Durresi, and Gojko Babic. 1999. *Throughput fairness index: An explanation*. Technical Report. Tech. rep., Department of CIS, The Ohio State University.

[15] Bryce Kellogg, Aaron Parks, Shyamnath Gollakota, Joshua R Smith, and David Wetherall. 2014. Wi-fi backscatter: internet connectivity for rf-powered devices. In *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 607–618.

[16] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R Smith. 2016. Passive Wi-Fi: bringing low power to Wi-Fi transmissions. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 151–164.

[17] Vincent Liu, Vamsi Talla, and Shyamnath Gollakota. 2014. Enabling instantaneous feedback with full-duplex backscatter. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 67–78.

[18] Jiajue Ou, Mo Li, and Yuanqing Zheng. 2015. Come and be served: Parallel decoding for cots rfid tags. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. ACM, 500–511.

[19] Aaron N Parks, Angli Liu, Shyamnath Gollakota, and Joshua R Smith. 2014. Turbocharging ambient backscatter communication. In *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 619–630.

[20] Nathan Pletcher. 2008. *Ultra-Low Power Wake-Up Receivers for Wireless Sensor Networks*. Ph.D. Dissertation. University of California Berkeley.

[21] Vamsi Talla and Joshua R Smith. 2013. Hybrid analog-digital backscatter: A new approach for battery-free sensing. In *RFID (RFID), 2013 IEEE International Conference on*. IEEE, 74–81.

[22] Jue Wang, Haitham Hassanieh, Dina Katabi, and Piotr Indyk. 2012. Efficient and reliable low-power backscatter networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 61–72.

[23] Jue Wang and Dina Katabi. 2013. Dude, where's my card?: RFID positioning that works with multipath and non-line of sight. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. ACM, 51–62.

[24] Lei Yang, Yekui Chen, Xiang-Yang Li, Chaowei Xiao, Mo Li, and Yunhao Liu. 2014. Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 237–248.

[25] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. 2016. Hitchhike: Practical backscatter using commodity wifi. In *ACM SENSYS*.

[26] Pengyu Zhang, Pan Hu, Vijay Pasikanti, and Deepak Ganesan. 2014. EkhoNet: high speed ultra low-power backscatter for next generation sensors. In *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM, 557–568.

[27] Pengyu Zhang, Mohammad Rostami, Pan Hu, and Deepak Ganesan. 2016. Enabling practical backscatter communication for on-body sensors. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*. ACM, 370–383.