HitchHike: Practical Backscatter Using Commodity WiFi

Pengyu Zhang[1*], Dinesh Bharadia[2*], Kiran Joshi[1], Sachin Katti[1]
Stanford University[1], MIT[2]
Co-primary Student Authors[*]
{pyzhang, krjoshi, skatti}@stanford.edu, dineshb@csail.mit.edu

ABSTRACT

We present HitchHike, a low power backscatter system that can be deployed entirely using commodity WiFi infrastructure. With HitchHike, a low power tag reflects existing 802.11b transmissions from a commodity WiFi transmitter, and the backscattered signals can then be decoded as a standard WiFi packet by a commodity 802.11b receiver. HitchHike's key invention is a novel technique called **codeword translation**, which allows a backscatter tag to embed its information on standard 802.11b packets by just translating the original transmitted 802.11b codeword to another valid 802.11b codeword. This allows any 802.11b receiver to decode the backscattered packet, thus opening the doors for widespread deployment of low-power backscatter communication using widely available WiFi infrastructure. We show experimentally that HitchHike can achieve an uplink throughput of up to 300Kbps at ranges of up to 34m and ranges of up to 54m where it achieves a throughput of around 200Kbps.

CCS Concepts

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

Keywords

Backscatter; WiFi; Wireless

1. INTRODUCTION

Backscatter communication has recently attracted interest for applications such as implantable sensors, wearables, and smart home sensing because of its ability to offer near zero-power connecitivity to these sensors. These applications have severe power constraints, implantable sensors for example have to last for decades, while even more traditional smart home monitoring applications will benefit from sensors and actuators that can last several years. Backscatter communication can satisfy the connectivity requirements while consuming so little power that it could be powered by harvesting alone, or with batteries that could last several years.

Traditional backscatter systems however require specialized hardware to generate the excitation RF signals that backscatter radios can reflect, as well as to decode the backscattered signals. Recent research such as WiFi backscatter [18], BackFi [2] and Passive WiFi [20] has reduced the need for specialized hardware. Passive WiFi for example can decode backscattered signal using standard WiFi radios. However, it still requires a dedicated continuous wave signal generator as the excitation RF signal source. BackFi needs a proprietary full duplex hardware add-on to WiFi radios to enable backscatter communication. Inter-Technology Backscatter [14] is a system that enables backscatter communication from a commercial Bluetooth radio to a commercial WiFi radio. Despite its novelty, it does not enable backscatter communication among WiFi radios. Consequently, a backscatter system that can be deployed using commodity WiFi radios on access points, smartphones, watches and tablets, does not exist.

Further, recent work such as Passive WiFi also requires twice the spectrum compared to other approaches such as BackFi, since they generate the RF excitation tone signal on the center of two WiFi channels and the backscatter tag frequency shifts and reflects the excitation signal on both the two WiFi channels. Due to the frequency shifting, the WiFi device (the reader) that is decoding the backscattered signal does not need full duplex hardware. However the tradeoff is that for a single backscatter communication link, we end up using two 20Mhz WiFi channels. For example, the continuous wave signal generator sends out a tone at the center of Channel 3 and the backscatter occupies both Channel 1 and 6, rendering them unusable for any other WiFi communication. Given the paucity of spectrum in the unlicensed band, solutions that do not waste spectrum would be desirable.

We introduce HitchHike, the first backscatter communication system that works using only commodity 802.11b WiFi devices for both generating the RF excitation signal as well as decoding the backscattered signal. We observe that most users are surrounded by multiple WiFi radios, either in APs or on their phones and tablets or even on their smartwatches. HitchHike utilizes these commodity WiFi radios as both an RF source for backscatter and a receiver which decodes the backscattered signals. This allows HitchHike to be very cost effective and widely deployable since it can benefit from the ubiquity and low-cost nature of WiFi

SenSys '16, November 14-16, 2016, Stanford, CA, USA
© 2016 ACM. ISBN 978-1-4503-4263-6/16/11...$15.00
DOI: http://dx.doi.org/10.1145/2994551.2994565

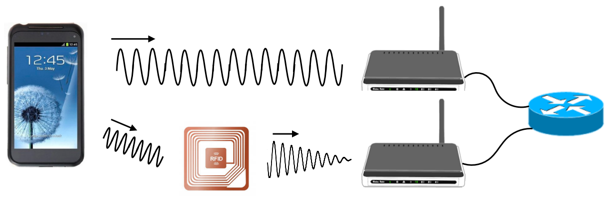


Figure 1: HitchHike concept: HitchHike enables backscatter communication between commodity 802.11b WiFi radios.

and eliminate the need for a specialized, dedicated reader or receiver. Second, HitchHike does not waste spectrum, it piggybacks backscattered signals on WiFi packets that are being used for productive communication. Hence, HitchHike can be efficiently deployed with current WiFi infrastructure and unlicensed spectrum.

HitchHike's deployment is best explained via the example in Figure 1. The excitation device is a smartphone with a standard WiFi radio. The smartphone transmits an 802.11b packet to the first AP to which it is connected on Channel 1. To backscatter, the tag receives the WiFi packet, frequency shifts it to Channel 6, modulates its information and then reflects the WiFi signal. The second AP, which is tuned to listen on Channel 6, then receives and decodes the backscatter packet as a standard WiFi packet. As shown in Figure 1, deploying HitchHike is very simple. It does not require any specialized hardware nor does it waste any spectrum. Both WiFi channels are used for productive communication, one for standard WiFi and the other for the backscatter link.

The key challenge in realizing HitchHike is: how can a backscatter tag produce a WiFi compliant packet by backscattering another WiFi compliant packet and also modulate its data on the resulting packet? HitchHike's key conceptual contribution here is **codeword translation**. Specifically, every 802.11b WiFi packet is a sequence of codewords that are picked from a codebook to represent different bits that are being transmitted. HitchHike's backscatter tags act as codeword translators, in other words they take a valid codeword in the transmitted 802.11b packet and translate it into a different valid codeword from the 802.11b codebook. The specific translation encodes the bit that the backscatter tag itself wants to communicate. The backscattered packet is therefore like any other 802.11b packet, albeit with a sequence of translated codewords depending on the data that backscatter tag wants to communicate. Consequently it can be decoded by any standard 802.11b receiver.

HitchHike's practical design makes two contributions in realizing the concept of codeword translation:

- **Efficient XOR Decoder**: HitchHike's decoding operation at the receiver is a simple XOR of the transmitted 802.11b packet and the backscattered 802.11b packet. The decoder is efficient because it allows the original 802.11b packet to be used for productive communication and simply piggybacks backscatter data on that transmission.

- **Spectrally efficient frequency shifting**: If the backscatter tag simply reflects the transmitted 802.11b packet, the receiver cannot likely decode the backscattered packet since its received simultaneously on the same channel as the original transmission. This leads to strong self-interference from the original 802.11b transmission leading to decoding failure. To tackle this challenge, like recent work [20, 35], HitchHike uses frequency shifting where the backscattered packet is shifted and transmitted on an adjacent non-overlapping WiFi channel. However these frequency shifting techniques produce two sidebands, one is the desired adjacent channel backscatter transmission whereas the other is an unwanted and wasted sideband that can also interfere with other communication. Both HitchHike and Inter-Technology Backscatter [14] describe a single sideband frequency shifting technique that ensures that only the desired backscatter sideband is produced and other spurious sidebands are eliminated.

The two techniques listed above together ensure that HitchHike is spectrally efficient and does not waste spectrum unlike prior work based on frequency shifting. Both WiFi channels are used for productive communication, and no spurious sidebands are produced.

We also note that the focus of this paper is on the uplink from the HitchHike tag to the HitchHike AP. The reason is that the IoT applications that we are designing for are bottle-necked on the uplink. These gadgets (such as fitness trackers, home sensors, wearables, etc) are collecting a lot of sensor data and need to upload them to the cloud and downlink often isn't needed, or if it is, very low throughput of a few Kbps suffice [24]. Hence in the rest of the paper we will focus on the uplink, but note that prior work has already demonstrated WiFi backscatter designs (which can be used with HitchHike too) for the downlink that can provide upto 20 Kbps [18].

We implement HitchHike using an FPGA and a customized backscatter analog front end board. Our empirical evaluation shows the following:

- HitchHike is able to decode backscattered data even though the HitchHike decoder (an MacBook Pro laptop) is 54m away from a tag in the line-of-sight (LOS) deployment, 1.5× longer than the maximum range reported by Passive WiFi. In non-line-of-sight (NLOS) deployment, HitchHike is able to decode the tag data at 32m.

- HitchHike is able to achieve close to 300Kbps throughput when the receiver is less than 34m from the tag in LOS deployment. At farther distances in LOS and NLOS deployments, HitchHike is able to achieve an average of 222kbps and 50kbps throughput respectively.

- We demonstrate that our HitchHike tag only consumes power on the order of 33μW despite the fact that it moves the backscattered signal into another WiFi channel. We also show that our system is able to coexist gracefully with existing WiFi infrastructure.

2. 802.11B PRIMER

HitchHike backscatters 802.11b packets from commodity WiFi devices. Here we give a brief description of how 802.11b packets are encoded and decoded to provide context to the underlying HitchHike's design.

1 Mbps and 2 Mbps DSSS transmission: An 802.11b radio uses a finite set of codewords to encode packets. For example, 1 Mbps 802.11b transmission uses only two codewords, *code*$_0$ and *code*$_1$, as shown in equation 1. Data zero and one are encoded as *code*$_0$ and *code*$_1$ respectively. The only difference between the two codewords is a 180^o phase offset, which indicates whether data zero or one is transmitted. The barker code used by the two codewords is a sequence similar to the PN sequence used in the CDMA system. It is designed to significantly increase the SNR at the decoder. For example, 802.11b decoder can decode 1 Mbps at -95 dBm.

$$\begin{aligned} code_0 &= \text{barker} \\ code_1 &= \text{barker} \times e^{j\pi} \end{aligned} \tag{1}$$

Instead of using only two codewords, 2 Mbps 802.11b uses four codewords in its codebook to encode packets as shown in equation 2. Data 00, 01, 11, and 10 are encoded as *code*$_0$, *code*$_1$, *code*$_2$, and *code*$_3$ respectively. Again, the data are embedded in the phase of the codewords.

$$\begin{aligned} code_0 &= \text{barker} \\ code_1 &= \text{barker} \times e^{j\frac{\pi}{2}} \\ code_2 &= \text{barker} \times e^{j\pi} \\ code_3 &= \text{barker} \times e^{j\frac{3\pi}{2}} \end{aligned} \tag{2}$$

5.5 Mbps and 11 Mbps CCK transmission: 5.5 Mbps and 11 Mbps CCKs use a larger set of codewords compared to the 1 Mbps and 2 Mbps cases. Equation 3 shows the codewords used by the 5.5 Mbps transmission. To transmit at 5.5 Mbps, CCK divides the bit stream into blocks of four bits. The first two bits are used to determine the phase θ of the codeword, which varies among 0, $\frac{\pi}{2}$, π, and $\frac{3\pi}{2}$. The last two bits are used to choose one of the four barker codes. 11 Mbps CCK transmission uses a similar strategy where the data stream is divided into blocks of 8 bits. Then, the first two bits are used to select the phase, and the last six bits are used to choose one of the 64 Barker codes.

$$\begin{aligned} code_0 &= \text{barker}_0 \times e^{j\theta_0} \\ &\cdots \\ code_{15} &= \text{barker}_{15} \times e^{j\theta_{15}} \end{aligned} \tag{3}$$

In summary, 802.11b WiFi protocol uses a finite set of codewords to encode packets. Our backscatter system leverages the fact that only a finite set of codewords are used, and if the tag can translate the codeword *code*$_i$ used by the 802.11b transmitter to another codeword *code*$_j$ within the same set, then any 802.11b receiver can decode the backscattered packet. We now turn to discuss our design, which is based on this observation.

3. DESIGN

Figure 2 shows an overview of our system. A commodity WiFi radio transmits a normal 802.11b WiFi packet, the backscatter tag reflects the packet to another 802.11b WiFi radio while modulating its information. When the tag backscatters the packet, it shifts the frequency of the reflected signal to an adjacent WiFi channel. The 802.11b receiver listening on the adjacent WiFi channel receives the reflected WiFi packet, decodes the packet using the normal WiFi decoding chain, and then extracts the backscattered information from the decoded bit stream. Next, we discuss the key components of our system which enable this capability, including 802.11b code word translator, XOR decoder and spectrally efficient frequency shifter.

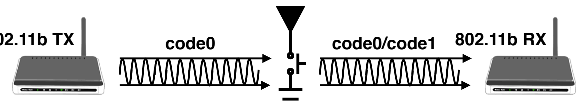


Figure 2: 802.11b code word translator at the tag.

3.1 HitchHike's Codeword Translator

The key idea underlying HitchHike is the concept of codeword translation. Conceptually, any modulation scheme (including WiFi's) is a mapping between bits and codewords from a discrete codebook. Decoding is the inverse operation, mapping from a received codeword to the actual bit. For a commodity WiFi receiver to decode the backscattered packet, its codewords need to come from the same codebook as that of WiFi. In other words, if the backscatter tag can act as a **codeword translator**, i.e. translate the codewords from the original 802.11b packet to other codewords in the 802.11b codebook, then a standard 802.11b receiver will be able to decode the packet, and a standard 802.11b transmitter can transmit original data. For example, equation 4 shows how the codeword for 1 and codeword for 0 are related, or in otherwords codeword 0 can be translated into codeword 1. The trick, of course, is to implement the translation based on what bits the tag wants to backscatter and such that the 802.11b receiver can recover the applied translation and therefore recover what bits the backscatter tag wanted to communicate.

$$\begin{aligned} &\text{codeword } 0 = 1 \times \text{barker} \\ &\text{codeword } 1 = -1 \times \text{barker} = \text{codeword } 0 \times e^{j\pi} \end{aligned} \tag{4}$$

For backscattering 802.11b 1Mbps signals, HitchHike's tag implements a simple translation. If it wants to backscatter the bit zero, then it does no translation and simply reflects the original codeword. If it wants to backscatter bit one, then it translates the received codeword to the only other valid codeword in the 802.11b 1Mbps codebook. To do so, it simply shifts the phase of the received codeword by 180 degrees as shown in equation 5. If the original 802.11b bit was zero, this will translate to a one being backscattered, and vice versa if the original 802.11b bit was one. This is shown via equation 5. Figure 2 shows an example of such translation. Section 3.5 describes how the tag physically implements the codeword translation.

$$\begin{aligned} &\text{Tag data } 0 = \text{802.11b data} \\ &\text{Tag data } 1 = \text{802.11b data} \times e^{j\pi} \end{aligned} \tag{5}$$

3.2 HitchHike's XOR Decoder

The 802.11b receiver can now decode the backscattered packet since all the codewords are valid codewords from the 802.11b 1Mbps codebook. But to recover the bits that the backscatter tag wanted to send, it needs to figure out what translations were applied to the original 802.11b bits. Our key observation here is that recovering the translation is equivalent to XOR-ing the decoded packet with the original 802.11b packet. To see why this is true, consider the codeword translation for 802.11b outlined above. If the original bit was one and the backscatter tag wanted to send one,

Table 1: Logic table between the decoded data, 802.11b bit, and backscatter bit.

decoded data	802.11b data	backscatter bits
1	1	0
1	0	1
0	1	1
0	0	0

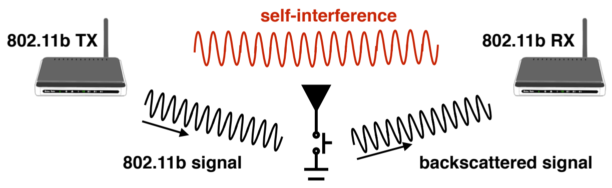


Figure 3: Self-interference from the 802.11b transmitter.

then translation was applied. Practically, this means that the phase shift by 180 degrees was applied twice (once at the transmitter and again at the backscatter tag), thus resulting into codeword corresponding to zero was backscattered. If the original bit was one and the tag wanted to send zero, then no translation was applied (no phase shift) and this resulted in one being backscattered. Similarly the other two combinations are shown in Table 1. As we can see, that simply gives us the logic table for a XOR operation. Thus, the decoded packet is basically original data XORed with the backscatter bit.

$$decoded\ packet = original\ data \oplus\ backscatter\ bits \quad (6)$$

Hence, to recover the backscatter bits, the receiver simply has to invert the XOR with original 802.11b packet. To do so however, it needs to know the original 802.11b packet. Our current design assumes that HitchHike will be deployed in a setting similar to the one shown in Figure 1. Here, a smartphone is transmitting the original 802.11b packet to WiFi AP1 which is listening on Channel 1. The tag implements codeword translation as described above, shifts the backscatter signal to channel 6 and the backscattered packet is decoded by AP2 which is listening on Channel 6. AP2 sends the decoded backscattered packet to AP1 which then implements the XOR operation to recover the backscatter bits from the tag. We expect such a deployment scenario to be quite typical in most enterprise or home scenarios, where often there are several APs listening on adjacent channels.

However, if two APs are not available, the smartphone can transmit a known 802.11b packet (e.g a packet of all 1s) and then any WiFi radio receiving the backscattered packet can XOR with the known 802.11b packet and recover the backscatter data. Finally, the above description was specific to the 1Mbps 802.11b transmission protocol. However the same concept of codeword translation can be applied to any of the 802.11b transmissions rates up to 2Mbps since codeword translation for all of them can be implemented by phase shifting techniques. Our implementation in this paper focuses on the 1Mbps bitrate, the other bitrates are the focus of our future work.

3.3 Spectrally Efficient Frequency Shifter

Once a tag has the ability to translate codewords between an 802.11b transmitter and an 802.11b receiver, it can modulate information on top of an 802.11b signal. However, in order to decode the backscattered tag data, we have to deal with another factor — self-interference from the 802.11b transmitter. Figure 3 shows an example of such interference. When an 802.11b receiver receives the backscattered signal, it also receives the signal from the 802.11b transmitter. This 802.11b signal acts as a strong interference because it shares the same frequency band as the backscattered signal and to make matters worse, the exciting signal is usually ∼30dB higher than the backscattered signal. Figure 4 shows the received signal strength of backscatter and the 802.11b self-interference. In this experiment, the backscatter tag is 1m away from the transmitter and we move the receiver away from the tag. We measured the backscattered and self-interference signal strength. At 10m, the self-interference is 40dB higher than the backscattered signal and makes the backscattered signal almost impossible to decode.

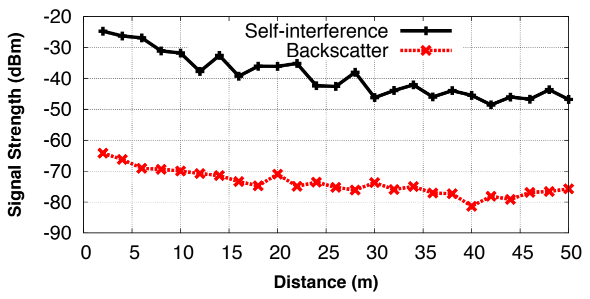


Figure 4: Signal strength of self-interference and backscattered signal.

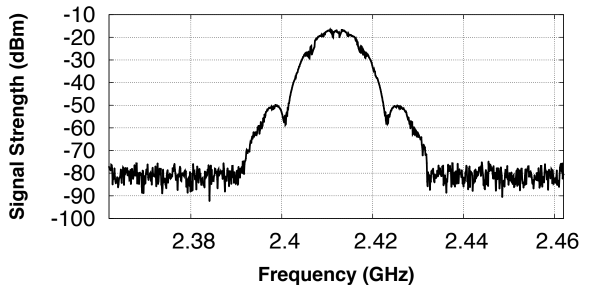


Figure 5: Empirically measured 802.11b transmission spectrum.

[2] addresses this problem by exploiting full-duplex communication techniques. However, [2] requires hardware modification on existing 802.11b radios, which is not preferred. Our system design sidesteps the self-interference problem by enabling the backscatter tag to frequency shift the backscattered signal to an adjacent, non-overlapping WiFi channel. However a non-overlapping WiFi channel does not imply that we will not see any self-interference, the WiFi transmission emits energy in adjacent channels too. Figure 5 shows the spectral profile of an 802.11b WiFi transmission. It shows that there is still a signal leaking into the adjacent band 11Mhz away from the center of the channel, albeit 30dB lower. However, the leaked signal strength degrades when we are further away, for example, 50dB signal degradation when we are 22MHz away from the center.

To implement such frequency shifting, HitchHike multi-

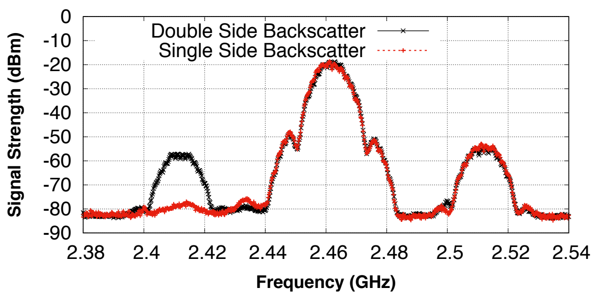


Figure 6: The spectrum of single side-band and double side-band backscatter.

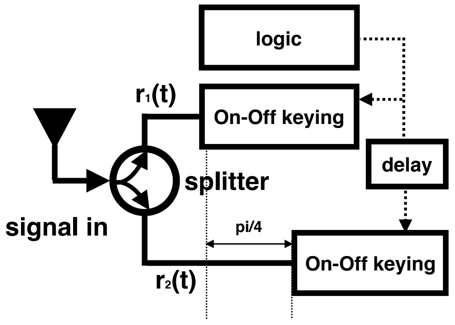
plies the 802.11b incident signal with a square wave produced by the tag as $S_{802.11b} \times S_{\text{tag}}$. When the frequency of the 802.11b incident signal is f_c and the tag square wave frequency is f_t, the backscattered signal will be moved to the $f_c \pm f_t$ band, which is f_t away from the original 802.11b signal. We can choose f_t large enough such that the self-interference from the incident 802.11b signal is small. In our implementation, our tag generates a square wave at 30MHz in order to move the backscattered signal 30MHz away from the incident 802.11b signal.

3.4 Producing single side-band backscatter

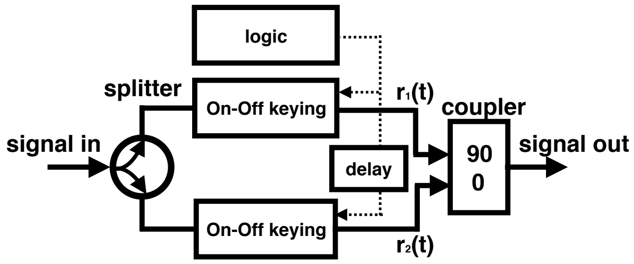
The simple frequency shifting design described above has a drawback: it creates copies of the 802.11b signal on both sides of the main signal as shown in Figure 6. In this experiment, the 802.11b transmitter transmits with a central frequency of 2.462GHz. The tag shifts the backscattered signal 50MHz away from the 802.11b signal. We see that such frequency shifting creates copies on both $2.462GHz \pm 50MHz$ (the red signal). The double sides backscatter signal creates unwanted interference in other band and would hurt any transmission happening in that band. As a result, the tag needs to create a single side band backscatter signal, with its data only on that band. Before we eliminate the other side-band, let's see why we get double sidebands.

When the tag toggles the RF switch at f_t frequency, it essentially uses a square wave $S_{tag}(f_t t)$ to modulate the incident 802.11b signal, this is basically a multiplication operation. The square wave signal can be presented using Fourier series as shown in equation 7 and the signal received at the tag by the 802.11b transmitter can be represented using $S_{802.11b}$ and $r(t)$ in equation 8 shows the signal backscatter by the tag. We assume the 802.11b signal as a sinsouid $sin(2\pi f_c t)$ for simplicity of anlaysis. From equation 8, we can observe sidebands on both sides of the spectrum $cos(2\pi(f_c - nf_t)t$ and $cos(2\pi(f_c + nf_t)t)$ with center frequencies $f_c + nf_t$ and $f_c - nf_t$ are created in the backscatter signal. Both HitchHike and Inter-Technology Backscatter [14] describe a technique to eliminate the unwanted side-band, which we explain next.

$$S_{tag}(f_t t) = \frac{4}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} sin(2\pi n f_t t) \tag{7}$$



(a) Single side-band backscatter with a single antenna



(b) Equivalent conceptual design

Figure 7: Figure 7(a) presents the backscatter design with an antenna for single side-band design. Figure 7(b) shows the equivalent design via unfolding the backscatter design for ease of analysis.

$$\begin{aligned} r(t) &= S_{802.11b} \times S_{tag}(f_t t) \\ &= sin(2\pi f_c t) \times S_{tag}(f_t t) \\ &= \frac{4}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} sin(2\pi f_c t) \times sin(2\pi n f_t t) \\ &= \frac{2}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} \{cos(2\pi(f_c - nf_t)t) \\ &- cos(2\pi(f_c + nf_t)t)\} \end{aligned} \tag{8}$$

We present a design that achieves single side band backscatter, while having low power consumption and introducing negligible loss on the backscattered signal strength. Conceptually, our design is similar to Inter-Technology Backscatter [14] despite the differences in the implementation details. Figure 7(b) shows the concept of this design. The key idea is that the tag receives the 802.11b signal from the transmitter and then splits it into two copies on two paths. Both copies pass through the on-off keying (square wave multiplier) on each path, and as a result, we create double side-band signal on each path. The trick we make here is that we make the signal on one path has a negative copy on the one side-band and has the same copy on the other side-band. Then, when we sum the signals from the two paths together, we end up with eliminating the signal on one side-band and increasing the signal on the other side-band. Let us now look at details to understand how we can make the signals on the two paths have different polarization.

The Tag first receives the 802.11b signal from the transmitter, and then splits it into two paths, where each contains the same copy of the 802.11b signal. We then pass both the signal with square wave multipler, however we delay the square wave signal on the second path by $\frac{1}{4f_t}$ in the time domain (which is equivalent to $\frac{pi}{2}$ phase shift) relative to the first path $r_1(t)$, and refer to it as the delayed copy $r_2(t)$. Both paths are multiplied by the square wave to shift their frequency. The mathematical formulation of $r_2(t)$ is shown in equation 10 while the non-delayed version $r_1(t)$ on the first path is shown in equation 9. The key observation here is that $r_2(t)$ has a $\frac{+\pi}{2}$ and $\frac{-\pi}{2}$ phase offset compared to the first path ($r_1(t)$) on $f_c - nf_t$ and $f_c + nf_t$, respectively because of the delay on $r_2(t)$. We can exploit these different phase offset on $f_c - nf_t$ and $f_c + nf_t$ to eliminate one of them, let us see how.

$$\begin{aligned} r_1(t) &= sin(2\pi f_c t) \times S_{tag}(f_o t) \\ &= \frac{4}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} sin(2\pi f_c t) sin(2\pi n f_t t) \\ &= \frac{2}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} \{cos(2\pi(f_c - nf_t)t) \\ &- cos(2\pi(f_c + nf_t)t)\} \end{aligned} \tag{9}$$

$$\begin{aligned} r_2(t) &= sin(2\pi f_c t) \times S_{tag}(f_t(t - \frac{1}{4f_t})) \\ &= \frac{4}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} sin(2\pi f_c t) sin(2\pi n f_t t - \frac{n\pi}{2}) \\ &= \frac{2}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} \{cos(2\pi(f_c - nf_t)t + \frac{n\pi}{2}) \\ &- cos(2\pi(f_c + nf_t)t - \frac{n\pi}{2})\} \end{aligned} \tag{10}$$

The idea is to add both the copies $r_1(t)$ and $r_2(t)$ in a unique way such that it cancels $f_c - nf_t$. The technique is, we phase shift $r_2(t)$, the delayed path by $\frac{\pi}{2}$ in RF domain and then add it back to $r_1(t)$ to get backscattered signal $r(t)$ (output signal). Observe that we actually introduced another $\frac{+\pi}{2}$ phase offset on the $f_c - nf_t$ frequency component on the delayed path signal $r_2(t)$, thus in total we have created π radians of phase shift on the $f_c - nf_t$ frequency for the delayed path $r_2(t)$ relative to the first path $r_1(t)$, thus $f_c - nf_t$ frequency component gets canceled upon addition with $r_1(t)$. The backscattered signal $r(t)$ can be represented by equation 12. On the other hand, the frequency component at $f_c + f_t$ has phase shift $\frac{-\pi}{2}$ at the delayed path $r_2(t)$, which undergoes $\frac{+\pi}{2}$ in RF domain before addition, thus the resulting in the phase shift of 0^o relative to the $r_1(t)$, thus $f_c + f_t$ gets added constructively and is the only component left un-canceled. As shown in equation 12, the backscattered signal $r(t)$ only contains frequency components in one side band $f_c + nf_t$. Figure 6 shows the empirically measured backscatter signal strength. Clearly we see that the single side band design eliminates at least 20 dB of other side band.

$$\begin{aligned} r(t) &= r_1(t) + r_2(t) \angle \frac{\pi}{2} \\ &= \frac{2}{\pi} \sum_{n=1,3,5...odd}^{\infty} \{\cancel{cos(2\pi(f_c - nf_t)t)} - cos(2\pi(f_c + nf_t)t)\} \\ &+ \{\cancel{cos(2\pi(f_c - nf_t)t + \frac{n\pi}{2} + \frac{\pi}{2})} \\ &- cos(2\pi(f_c + nf_t)t - \frac{n\pi}{2} + \frac{\pi}{2})\} \\ &= \frac{4}{\pi} \sum_{n=1,3,5...odd}^{\infty} \frac{1}{n} \{-cos(2\pi(f_c + nf_t)t)\} \end{aligned} \tag{11}$$

$$\begin{aligned} r(t) &= r_1(t) + r_2(t) \angle \frac{\pi}{2} \\ &= \frac{2}{\pi} \{\cancel{cos(2\pi(f_c - f_t)t)} - cos(2\pi(f_c + f_t)t)\} \\ &+ \{\cancel{cos(2\pi(f_c - f_t)t + \frac{\pi}{2} + \frac{\pi}{2})} - cos(2\pi(f_c + f_t)t - \frac{\pi}{2} + \frac{\pi}{2})\} \\ &= -\frac{4}{\pi} cos(2\pi(f_c + f_t)t) \end{aligned} \tag{12}$$

So far we have shown the input and output model. However, that is not the case with the backscatter tag. A tag only has one antenna, which is both input and output. Figure 7 on the left shows the backscatter tag design which is folded version of the desgin on the right side of the same figure. For simplicity, this section conducted tone analysis, which can very easily be extended to the 11b transmission, in the interest of space we omit it. Before we finish this section, we conduct an experiment, where we transmit 11b signal from the transmitter with the tag deployed at half a meter from the transmitter and the receiver is deployed at the 1 meter from the tag. We deploy the tags with and without single side band and show the spectrum plot as shown in Fig. 5. Clearly we see that SSB design eliminates atleast 30 dB of other side band.

3.5 Putting everything together

Let us now turn to look at how we put every piece of our system together and run a protocol that is able to identify when the 802.11b transmitter starts transmission and determine when the tag starts backscattering its data. Figure 8 shows the timing diagram of different events in our system. The 802.11b transmitter first occupies the wireless channel by issuing RTS-to-CTS messages. These messages will reserve two WiFi channels, one for the normal WiFi transmission and another for backscatter. We do not ask the tag to sense and occupy the wireless channel because the tag cannot afford the power consumed by spectrum sensing. Once successful, the 802.11b transmitter can proceed to synchronize with the tag for upcoming backscatter transmission.

In order to synchronize with the tag for its uplink transmission, the 802.11b transmitter sends a sequence of short 802.11b packets in the predetermined packet slots marked as P_1, P_2, ..., P_n in Figure 8. The presence and absence of packets in these predefined slots indicate the data transmitted to the tag using On-Off keying (OOK) modulation. In order to send data one, the 802.11b transmitter sends a packet in one time slot. In contrast, the 802.11b transmitter does not send a packet to encode data zero. This one and zero sequence transmitted by the 802.11b transmitter can be identified by the tag using an analog envelop detector. This

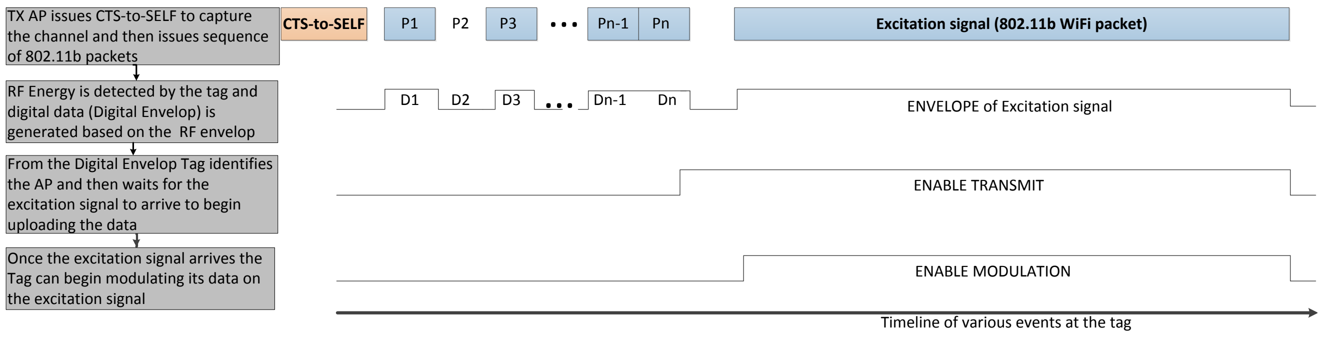

TX AP issues CTS-to-SELF to capture the channel and then issues sequence of 802.11b packets
RF Energy is detected by the tag and digital data (Digital Envelop) is generated based on the RF envelop
From the Digital Envelop Tag identifies the AP and then waits for the excitation signal to arrive to begin uploading the data
Once the excitation signal arrives the Tag can begin modulating its data on the excitation signal
CTS-to-SELF
P1
P2
P3
Pn-1
Pn
Excitation signal (802.11b WiFi packet)
D1
D2
D3
Dn-1
Dn
ENVELOPE of Excitation signal
ENABLE TRANSMIT
ENABLE MODULATION
Timeline of various events at the tag

Figure 8: Putting everything together: timing diagram of events in HitchHike.

envelop detector runs continuously on the tag which outputs a signal when the amplitude of the monitored signal is larger or smaller than a threshold. Once the tag decodes a predefined sequence from the 802.11b transmitter, it knows that it can embed backscattered bits on the next incoming WiFi packet, which is named excitation packet.

After sending the sequence of short packets for synchronizing with the tag, the 802.11b transmitter will start sending excitation packets, which are normal 802.11b packets. The tag captures the rising edge of each excitation packet and embeds the backscattered information on the packet. One important factor that we have to consider is that the backscattered bit stream should not corrupt the preamble of the 802.11b packet. Otherwise, 802.11b packet decoder will fail. In order to prevent such corruption, the tag waits for a deterministic amount of time after the tag detects the starting point of an excitation packet, which is 400μs in our implementation. This is because the 802.11b packet header is 384 bits and take 384μs to transmit it. In the end of a tag transmission, the tag embeds an end of packet message in the backscattered data to inform both the 802.11b transmitter and the receiver that the backscatter communication is done.

4. IMPLEMENTATION

We build a prototype of our system using commodity WiFi transceivers and a customized backscatter tag. We describe their implementation details below.

4.1 802.11b transceiver and tag hardware

802.11b transceiver: Our 802.11b receiver is a Macbook Pro laptop, which has a WiFi card that runs the 802.11a/b/g-/n/ac protocols. We use the sniffer tool embedded in the Wireless Diagnostics application to set the desired channel where we want to receive the backscattered signal. Then, we use tcpdump to analyze the received WiFi packets and extract the backscattered information.

We use an Intel 5300 WiFi card on an Intel NUC as the standard 802.11b transmitter. It supports 802.11 a/b/g/n data transmission. We use the firmware provided by [9] to control the rate of 802.11b packets transmission.

Backscatter tag: We now describe the implementation details of our backscatter tag. Figure 9 shows the diagram of the key components of our tag. The clock source of our system is a ring oscillator, which comprises odd number of inverters and is able to provide a 30MHz clock for the rest of the system. The 30MHz clock is divided into two paths

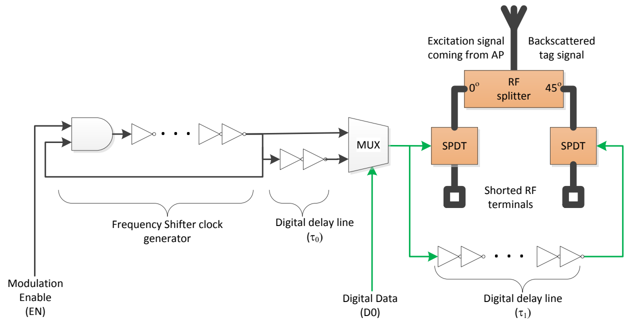

Excitation signal coming from AP
Backscattered tag signal
0°
RF splitter
45°
MUX
SPDT
SPDT
Shorted RF terminals
Frequency Shifter clock generator
Digital delay line (τ_0)
Modulation Enable (EN)
Digital Data (D0)
Digital delay line (τ_1)

Figure 9: HitchHike tag hardware diagram.

where one is directly connected to a multiplexer and the other goes through an inverter before being fed into the multiplexer. The reason we feed the clock on the second path into an inverter comes from the fact that the tag should modulate the 802.11b packets with 0^o or 180^o degree phase shifting. The 30MHz clock on the first path is able to shift the 802.11b signal by 30MHz with 0^o degree phase offset. The clock on the second path is also able to shift the 802.11b signal by 30MHz. However, it introduces a 180^o phase offset.

The state of the multiplexer is controlled by an Igloo Nano AGLN250 FPGA in our implementation. When the FPGA wants to transmit zero, it chooses the first path clock where 30MHz clock with 0^o phase offset is used. Similarly, when the FPGA wants to transmit one, it chooses the second path clock where 30MHz clock with 180^o phase offset is used.

The chosen clock is then again divided into two paths for creating the single side-band backscatter. The signal on the first path is connected to one RF switch without any delay. The signal on the second path is connected to another RF switch with a delay that is able to produce a 90^o phase offset. We implement this delay using inverters where 8.3ns delay is introduced because the signal frequency is 30MHz. The output of the second RF switch is passed through an RF delay line that is able to produce a 45^o phase offset from the antenna to the RF switch. At the last stage, the signals on the two paths are combined by the RF splitter before backscattering out.

In our implementation, we choose ADG902 RF switch and toggle it at 30MHz to move the backscattered signal away from the incident 802.11b signal. We choose this RF switch because of its low return loss. We use a Vector Network Analyzer to empirically measure the return loss of this RF

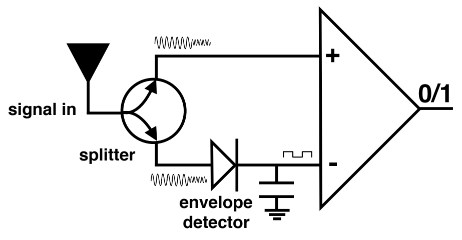


Figure 10: 802.11b analog envelop detector.

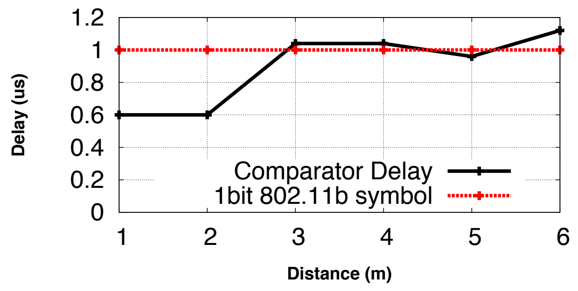


Figure 11: The delay between comparator signaling and 802.11b transmission.

switch. Only 3dB loss is observed, which suggests that we do not lose a significant amount of RF power during reflection.

Open source HitchHike platform: Both software and hardware of HitchHike platform will be available on Stanford website under an academic license to ensure reproducibility of results.

4.2 How to synchronize with 802.11b packets?

When the backscatter tag injects 1 bit of backscatter data on top of 1 bit of 802.11b data, we need to address one problem — how should we synchronize the tag bits stream with the 802.11b bits stream? and how does the synchronization error impact the performance of decoding?

We achieve such synchronization by using an energy detector at the tag side. Figure 10 shows the block diagram of our energy detector. It takes the 2.4GHz wireless signal as input and splits into two paths. The signal on one path is directly connected to a NCS2200SQ2T2G comparator and the signal on the other path is passed through a diode and a small capacitor before being connected to the comparator. Once the 802.11b transmitter starts transmission, the signal on the first path will become high immediately. However, the signal on the second path will remain low for a while. Since there is a difference between the two inputs of the comparator, the comparator is able to identify that the 802.11b transmission started.This comparator does not output jitter signals during an 802.11b packet because an 802.11b packet has a constant envelop in the time domain.

Figure 11 shows the delay between the comparator signaling and the start of an 802.11b transmission. We measure this delay across distance when an 802.11b transmitter sends at 30dBm. At 1m, our synchronization circuit introduces a 0.6μs delay. This delay increases when the tag is further away from the 802.11b transmitter because the tag receives less energy at further distance. At 6m, we experience 1.1μs delay. A natural question to ask is that how does this delay impact the backscatter decoding?

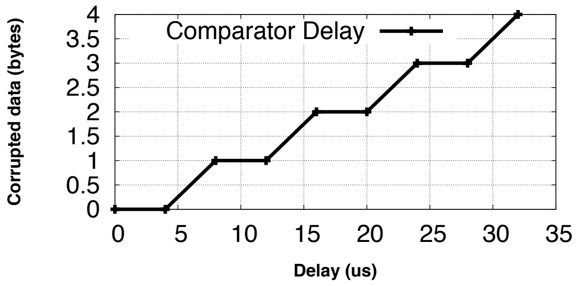


Figure 12: Corrupted backscatter data per packet due to the delay of our synchronization circuit.

Figure 12 shows the number of corrupted data bits per packet at the 802.11b receiver when we increase the delay introduced by our synchronization circuit. We introduce this artificial and controlled delay by sending 802.11b packets via an SMBV100A signal generator, which allows us to use a wire to inform the tag the starting point of an 802.11b transmission with a deterministic delay. We observe that when the delay is smaller than 8μs, zero bytes of data are corrupted. However, when the delay is between 8μs and 12μs, 1 byte of data is corrupted. The number of corrupted data becomes larger when the delay is larger. However, since our empirically measured synchronization delay shown in Figure 11 is smaller than 2μs, the delay introduced by the synchronization circuit does not impact the performance of our backscatter decoding at the 802.11b receiver.

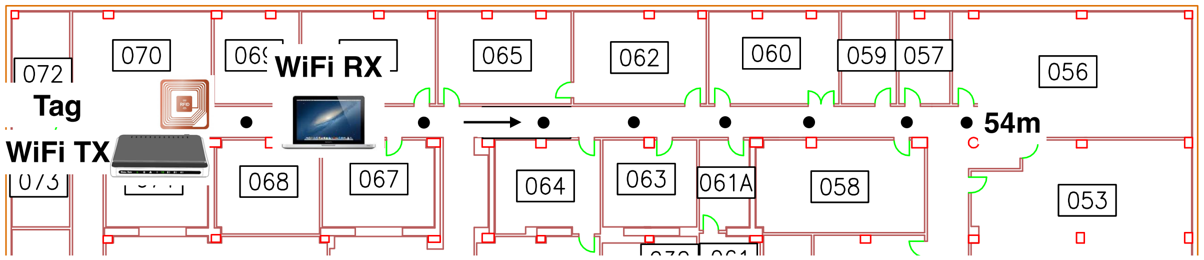
5. EVALUATION

We describe the experimental evaluation of our system to understand how our system performs in diverse deployments. Our experiments show the following:

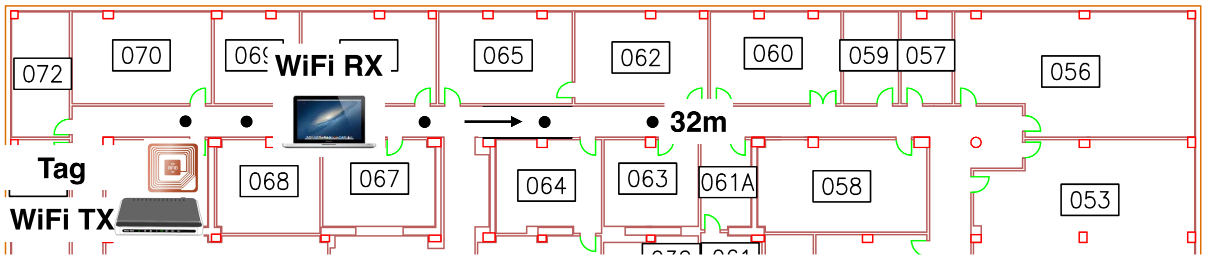
- The HitchHike prototype achieves an uplink backscatter range of 50m in line-of-sight scenarios, which is twice as better compared to prior backscatter systems such as Passive WiFi [20]. In non-line-of-sight (NLOS) deployment, our prototype achieves a range of 16m even when the backscatter signal has to pass through two walls.
- Our system is able to achieve close to 300Kbps throughput when the receiver is less than 34m from the tag in LOS deployment. At farther distance and NLOS deployment, our system is able to achieve an average of 222kbps and 50kbps throughput respectively.
- We also benchmark the operational regime of our system and shows that backscatter decoding works even when the tag is 6m away from the 802.11b transmitter.
- Finally we demonstrate that our low-power tag only consumes power on the order of 33μW despite the fact that it moves the backscattered signal into another band. We also show that our system is able to co-exist with existing WiFi infrastructure well.

5.1 HitchHike's Performance

First, we investigate HitchHike's range. We quantify performance using three metrics: throughput, bit error rate

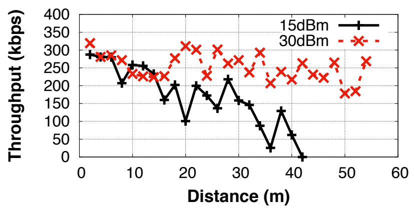


(a) LOS deployment

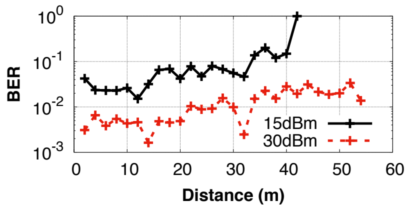


(b) NLOS deployment

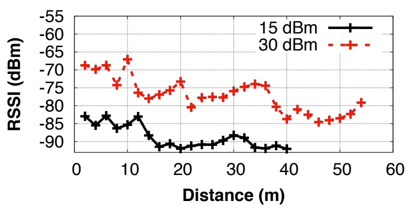
Figure 13: Floor plan and experiment setup of our backscatter system in LOS and NLOS deployment.



(a) Througphut



(b) BER



(c) RSSI

Figure 14: Backscatter throughput, BER, and RSSI across distance in line-of-sight deployment where WiFi TX-to-tag distance is 1m.

(BER), and received signal strength indicator (RSSI). Figure 13(a) and Figure 13(b) show the floor plan and the experiment setup of our system in both LOS and NLOS deployment. For the LOS deployment, all devices are deployed in a hallway. For the NLOS deployment, the 802.11b transmitter and the tag are deployed in a room while the 802.11b receiver is deployed in the hallway, which is separated from the tag by one or two walls depending on distance. We deploy the tag 1m away from the 802.11b transmitter. Then, we move the 802.11b receiver away from the tag and measure the achieved throughput, BER and RSSI across distances.

5.1.1 *Line-of-sight performance*

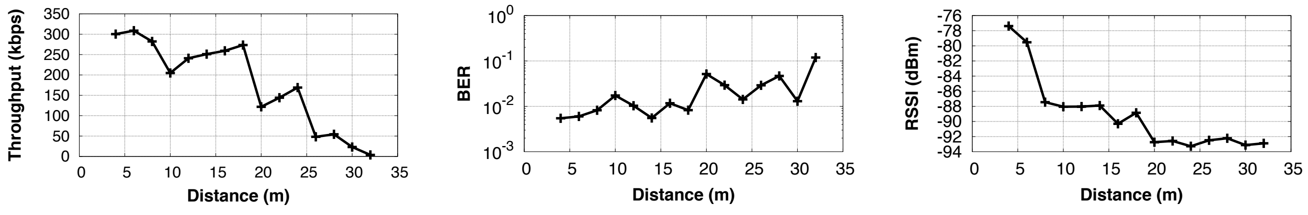
Figure 14(a) shows the throughput of our backscatter system with increasing range. As we can see, the maximal operational distance of our system is 54m, 1.5×∼2× longer than the range reported by Passive WiFi [20] that leverages a single tone using dedicated hardware as the carrier signal for generating the backscattered 802.11b packets. Second, we can achieve close to 300Kbps throughput when the 802.11b receiver is 34m from the tag. Such throughput can meet the requirement of many applications of wearables and Internet of Things where sensor data are collected at a rate of ∼100kbps.

When we move the 802.11b receiver away from the tag, backscatter throughput decreases because the backscattered signal strength decreases as shown in Figure 14(c). When the 802.11b receiver is more than 40m away from the tag, the received backscatter signal strength is below -80dBm and makes decoding harder. Figure 14(b) shows that bit error rate increases from 10^{-2} to 10^{-1} at longer distances and as a result, throughput decreases.

5.1.2 *Non-line-of-sight Performance*

Figure 15(a) shows the backscatter throughput in the NLOS deployment. We find two observations. First, the maximum communication distance achieved is 32m, longer than the maximum distance reported by Passive WiFi [20]. Second, we are able to achieve 100∼200kbps throughput when the 802.11b receiver is within 25m from the tag.

As expected, the throughput and range of our backscatter system degrades compared with the LOS deployment. For example, the maximum communication distance achieved is 32m, much shorter than the 54m of the LOS deployment. Such performance degradation is caused by the lower backscattered signal strength as shown in Figure 15(c). The backscattered signal strength is -76dBm when the 802.11b receiver is 1m away from the tag, which is 10dB lower than



(a) Througphut (b) BER (c) RSSI

Figure 15: Backscatter throughput, BER, and RSSI across distance in non-line-of-sight deployment where WiFi TX-to-tag distance is 1m.

the LOS deployment because there is a wall between the tag and the 802.11b receiver in the NLOS deployment. When the distance between the tag and the 802.11b receiver increases, the backscattered signal strength drops sharply. When the 802.11b receiver is more than 10m away, the backscattered signal strength is around -88dBm, close to the noise floor. As a result, decoding the backscattered signal becomes much harder.

5.1.3 *Impact of WiFi Transmitter Power*

We also evaluate the performance of our system when the 802.11b transmitter sends at different power levels. On the 2.4GHz unlicensed band, the maximum peak power allowed by FCC is 30dBm. However, many WiFi transmitters choose to send at a lower power level. Figure 14(a) also shows the throughput of a tag when the 802.11b transmitter sends at 15dBm and 30dBm. For 15dBm transmission, we can see the maximum range achieved is 42m and throughput degrades significantly when the 11b decoder is more than 16m away. As expected, the 30dBm transmission does improve the communication range to 54m and significantly reduce the corresponding bit error rate.

5.1.4 *Impact of Transmitter-Tag Distance*

Since backscatter performance depends on both of the 802.11b TX-to-tag distance and the 802.11b RX-to-tag distance, we first fix the 802.11b TX-to-tag distance and measure the maximum 802.11b RX-to-tag distance where backscatter decoding succeeds. Then, we change the 802.11b TX-to-tag distance and do the measurement again. Figure 17 shows the empirically measured communication range of our system. Backscatter communication still succeeds when the tag is 50m away from the 802.11b receiver or 6m away from the 802.11b transmitter. The backscatter tag cannot operate at a long distance from the 802.11b transmitter because the tag cannot identify the excitation packet sent by the transmitter. In addition, the tag cannot be far away from both of the 802.11b transmitter and receiver either. For example, when the tag is 6m away from the 802.11b transmitter, the maximum distance between the 802.11b receiver and the tag is 8m. We also calculate the theoretical communication range of our system using the Friis model and the backscattered signal strength measured when the tag is 50m away from the 802.11b transmitter. As shown, the theoretical curve matches the empirically measured curve well except that the empirically measured communication distance is a bit shorter.

5.1.5 *Impact of Synchronization Signal Delay*

One factor that potentially impacts the performance of our system is the time-domain delay between the tag bits stream and the 802.11b bits stream. In this experiment, we use an SMBV100A signal generator to transmit 802.11b packets. This signal generator is able to output a signal that indicates the start of an 802.11b packet. We use a wire to feed this signal to the tag and inform the tag when the 802.11b packet transmission starts. More importantly, we can introduce a deterministic delay on this signal to emulate the time-domain jitters experienced by the tag.

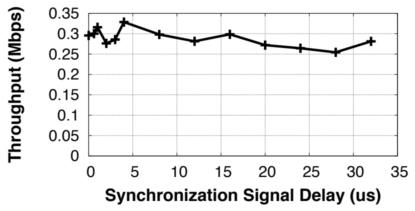
Figure 16(a), 16(b), and 16(c) show the throughput, BER, and RSSI of the backscattered signal when we introduce $0.6\mu s$ to $32\mu s$ delay. We can see the backscatter throughput does decrease a bit when the delay value increases. However, the throughput degradation is not significant. The reason is that only the first several bits of a backscatter packet are corrupted. The rest of the backscatter packet can still be decoded correctly. As a result, throughput degradation is not significant. In addition, as we show in section 3, the empirically measured delay is less than $2\mu s$ across distances. Therefore, the throughput degradation is even smaller. Figure 16(b) and Figure 16(c) also show similar conclusion where we can observe similar bit error rate and received backscattered signal strength when the delay value changes from $0.6\mu s$ to $32\mu s$.

5.1.6 *Impact of Packet Checksum*

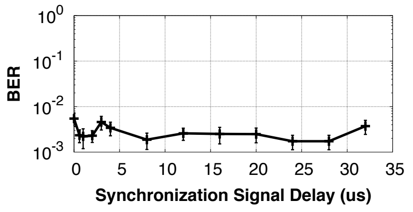
When a HitchHike tag converts an incoming WiFi packet into a backscattered packet, it does not know the content of the WiFi packet. Therefore, although the backscattered packet is a valid WiFi packet on the physical layer, its packet checksum could be wrong. Can we receive a packet when its checksum is wrong? The answer depends on the WiFi receiver hardware. On some receivers, such as Macbook Pro, we are able to receive and decode packets with bad checksum as long as we configure these receivers into monitor mode. However, some WiFi receivers do not provide this capability and they drop packets with bad checksum in hardware. For those radios, they cannot be used to decode the backscatter packets generated by a HitchHike tag.

5.2 Tag Power Consumption

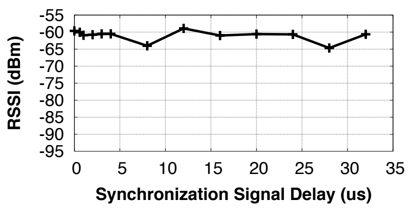
Figure 18 shows a breakdown of the power consumed by different components of our tag, which is obtained by using a 45nm power analysis tool. The clock module, which produces a 30MHz clock for the rest of the system, actually only consumes $4\mu W$ of power. The data modulator, which embeds backscattered bits on top of the 30MHz clock, con-



(a) Througphut



(b) BER



(c) RSSI

Figure 16: Benchmark backscatter throughput, BER, and RSSI when the synchronization signal is delayed.

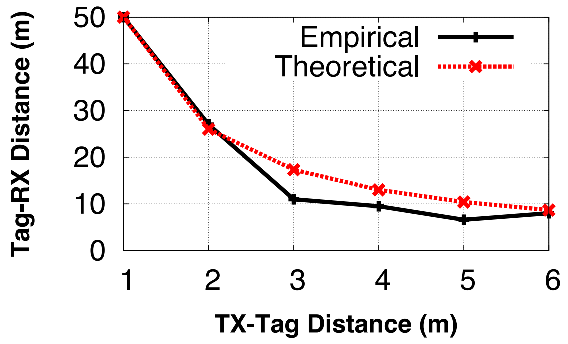


Figure 17: Communication range of our system.

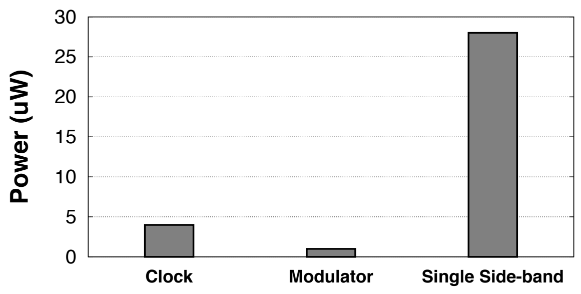


Figure 18: HitchHike tag power consumption.

sumes 1μW of power. The module which is responsible for generating the single side-band backscatter consumes 28μW of power. The total power consumed by our system is only 33μW, comparable with the 59.2μW consumed by Passive WiFi [20]. In addition, many energy harvesting system, such as small solar panel, are able to provide continuous power above 33μW even in indoor environment. Therefore, our tag has the potential of being deployed without batteries.

5.3 Co-existence with WiFi Networks

Finally, we investigate how well backscatter communication co-exists with concurrent WiFi communication. In this experiment, we deploy a backscatter tag 4m away from an 802.11b transmitter. The 802.11b transmitter is 3m away from a laptop, which transmits continuous WiFi packets to another laptop that is 5m away from the the 802.11b transmitter. The 802.11b transmitter sends 802.11b packets on channel 7 (2.442GHz), the tag shifts the backscattered signal by 30MHz to channel 13 (2.472GHz), and we run the WiFi stream between the two laptops on channel 1 (2.412GHz).

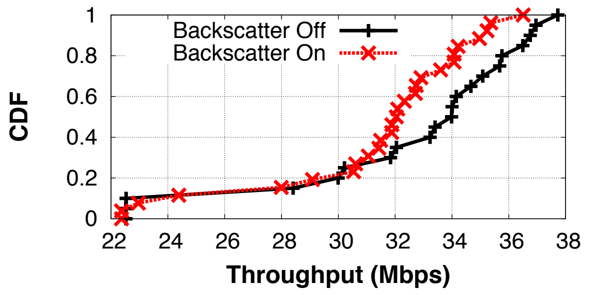


Figure 19: WiFi throughput when backscatter is present and absent.

We look at the throughput of the WiFi transmission between the two laptops as well as our backscatter system, and understand how they impact each other.

5.3.1 *How does backscatter impact WiFi?*

Figure 19 shows the WiFi throughput between the two laptops when backscatter is present and when it is absent. When we turn off the backscatter transmission, the median WiFi throughput achieved is 33.9Mbps. The WiFi throughput varies between 22Mbps and 38Mbps because of human movement nearby. When backscatter is present, the median WiFi throughput drops a bit to 32Mbps, 5% smaller. As a result, backscatter does not cause severe interference to the WiFi streams. The reason is that the backscattered signal does not have an overlapping spectrum with the active WiFi transmission. Further, the backscattered signal strength is usually below -70dBm, much lower than the signal strength of the active WiFi transmission. As a result, backscatter communication does not severely impact active WiFi transmission.

5.3.2 *How does WiFi impact backscatter?*

We now turn to look at the other case — how does active WiFi transmission between the two laptops impact the throughput of backscatter? Figure 20 shows the backscatter throughput when the WiFi stream between the two laptops is present and absent. When we turn off the WiFi stream on the laptops, the median backscatter throughput achieved is 236Kbps. When the WiFi stream is present, the median backscatter throughput drops to 220Kbps, 7% smaller. The impact is small because the backscattered signal is 60MHz away from the WiFi stream on the frequency domain and any interference leaking in the frequency domain is quite small.

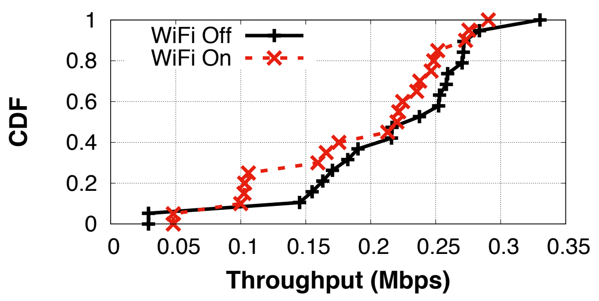


Figure 20: Backscatter throughput when WiFi is present and absent.

6. RELATED WORK

Our system is the first one that enables backscatter communication between two commodity 802.11b WiFi radios without any hardware modification. Our work is inspired by recent progress on backscatter systems, interference cancellation and network coding. Let us discuss related work and highlight our differences from the previous work.

Backscatter systems: In recent years, we have seen a significant throughput and communication range improvement of backscatter systems [30, 7, 1, 8, 10, 11, 12, 13, 19, 25, 29, 26, 28, 33, 23, 27, 31, 34, 32]. [21] leverages TV signals and enables backscatter communication between two battery-less devices. [18] looks at the opportunity of backscattering information on top of a WiFi signal. [2] leverages full-duplex technique to enable backscatter communication with a WiFi AP. [6] emulates BLE transmission by backscattering a BLE baseband signal on top of a single tone. [20] leverages similar ideas by emulating an 802.11b transmission on top of a single tone signal. Inter-Technology Backscatter [14] enables backscatter communication from a commercial Bluetooth radio to a commercial WiFi radio. However, none of these previous work enables backscatter communication between two 802.11b WiFi radios without any hardware modification or deploying a specific single tone emitter. Since WiFi radios and access points are ubiquitous and have been integrated in almost every laptop, smartphone, etc. More importantly, all these WiFi radios and access points are backward compatible in terms of being able to transmit and decode 802.11b packets even though they primarily run 802.11g/n/ac protocols. Therefore, the capability of doing backscatter communication between two 802.11b radios will enable a rich set of applications in the wearables and Internet of Things fields.

Interference cancellation: One key problem that we have to address is the strong self-interference experienced by a backscatter system when backscattering between two commercial 802.11b radios. [15] [5] [3] [4] [22] develop a set of full-duplex cancellation techniques to reduce self-interference in WiFi and MIMO systems. However, we cannot use these techniques in our system for two reasons. First, we need to do hardware modification on existing radios, which is not preferred. Second, more importantly, full duplex techniques cannot be used when the 802.11b transmitter and receiver are physically separated and mobile. Our system addresses this problem by moving the backscattered signal away from the incident signal in the frequency domain. In other words, we allocate another clean channel for backscatter communication.

Network coding: The WiFi bits stream XOR used by our system for improving spectrum efficiency is inspired by previous work on network coding. [16] and [17] look at the opportunity of mixing multiple data streams using network coding to improve the wireless network performance. Our system takes a similar idea by XOR-ing the original 802.11b stream with the backscattered 802.11b stream before feeding it into the backscatter decoder. This operation enables the 802.11b transmitter to send arbitrary data, which is different from previous systems [6] [20] where one of the 2.4GHz channel is wasted due to transmitting a single tone.

7. CONCLUSION

HitchHike is the first backscatter communication system that can be deployed completely using commodity WiFi infrastructure. Further, it is spectrally efficient, unlike prior solutions it does not waste precious unlicensed spectrum. We plan to investigate how to extend HitchHike to use other infrastructure such as GSM, 3G and LTE as well as other flavors of WiFi such as 802.11g/n/ac.

8. ACKNOWLEDGMENTS

We thank the shepherd Joshua R. Smith, the anonymous reviewers, Eyal Cidon, and Manikanta Kotaru for their insightful comments.

9. REFERENCES

[1] O. Abari, D. Vasisht, D. Katabi, and A. Chandrakasan. Caraoke: An e-toll transponder network for smart cities. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 297–310, New York, NY, USA, 2015. ACM.

[2] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti. Backfi: High throughput wifi backscatter. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 283–296. ACM, 2015.

[3] D. Bharadia and S. Katti. Fastforward: fast and constructive full duplex relays. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 199–210. ACM, 2014.

[4] D. Bharadia and S. Katti. Full duplex mimo radios. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 359–372, Seattle, WA, Apr. 2014. USENIX Association.

[5] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti. Achieving single channel, full duplex wireless communication. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 1–12. ACM, 2010.

[6] J. F. Ensworth and M. S. Reynolds. 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*, pages 78–85. IEEE, 2015.

[7] S. Gollakota, M. S. Reynolds, J. R. Smith, and D. J. Wetherall. The emergence of rf-powered computing. *Computer*, 47(1):32–39, 2014.

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

[9] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool release: gathering 802.11 n traces with channel state information. *ACM SIGCOMM Computer Communication Review*, 41(1):53–53, 2011.

[10] H. Hassanieh, J. Wang, D. Katabi, and T. Kohno. Securing rfids by randomizing the modulation and channel. NSDI, 2015.

[11] P. Hu, P. Zhang, and D. Ganesan. Leveraging interleaved signal edges for concurrent backscatter. In *Proceedings of the 1st ACM workshop on Hot topics in wireless*, pages 13–18. ACM, 2014.

[12] P. Hu, P. Zhang, and D. Ganesan. Laissez-faire: Fully asymmetric backscatter communication. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 255–267. ACM, 2015.

[13] P. Hu, P. Zhang, M. Rostami, and D. Ganesan. Braidio: An integrated active-passive radio for mobile devices with asymmetric energy budgets. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 384–397. ACM, 2016.

[14] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 356–369. ACM, 2016.

[15] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha. Practical, real-time, full duplex wireless. In *Proceedings of the 17th annual international conference on Mobile computing and networking*, pages 301–312. ACM, 2011.

[16] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: analog network coding. In *ACM SIGCOMM Computer Communication Review*, volume 37, pages 397–408. ACM, 2007.

[17] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. *IEEE/ACM Transactions on Networking (ToN)*, 16(3):497–510, 2008.

[18] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: internet connectivity for rf-powered devices. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 607–618. ACM, 2014.

[19] B. Kellogg, V. Talla, and S. Gollakota. Bringing gesture recognition to all devices. In *Usenix NSDI*, volume 14, 2014.

[20] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith. Passive wi-fi: Bringing low power to wi-fi transmissions.

[21] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: wireless communication out of thin air. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 39–50. ACM, 2013.

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

[23] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 619–630. ACM, 2014.

[24] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers. A review of wearable sensors and systems with application in rehabilitation. *Journal of neuroengineering and rehabilitation*, 9(1):1, 2012.

[25] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith. Powering the next billion devices with wi-fi. *arXiv preprint arXiv:1505.06815*, 2015.

[26] J. Wang, F. Adib, R. Knepper, D. Katabi, and D. Rus. Rf-compass: robot object manipulation using rfids. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 3–14. ACM, 2013.

[27] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 61–72. ACM, 2012.

[28] J. Wang and D. Katabi. Dude, where's my card?: Rfid positioning that works with multipath and non-line of sight. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 51–62. ACM, 2013.

[29] J. Wang, D. Vasisht, and D. Katabi. Rf-idraw: virtual touch screen in the air using rf signals. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 235–246. ACM, 2014.

[30] P. ZHANG, D. Bharadia, K. Joshi, and S. Katti. Enabling backscatter communication among commodity wifi radios. In *Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference*, SIGCOMM '16, pages 611–612, New York, NY, USA, 2016. ACM.

[31] P. Zhang and D. Ganesan. Enabling bit-by-bit backscatter communication in severe energy harvesting environments. *NSDI, Berkeley, CA*, 2014.

[32] P. Zhang, D. Ganesan, and B. Lu. Quarkos: Pushing the operating limits of micro-powered sensors. In *Proceedings of the 14th USENIX conference on Hot Topics in Operating Systems*, pages 7–7. USENIX Association, 2013.

[33] P. Zhang, J. Gummeson, and D. Ganesan. Blink: A high throughput link layer for backscatter communication. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 99–112. ACM, 2012.

[34] P. Zhang, P. Hu, V. Pasikanti, and D. Ganesan. Ekhonet: high speed ultra low-power backscatter for next generation sensors. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 557–568. ACM, 2014.

[35] P. Zhang, M. Rostami, P. Hu, and D. Ganesan. Enabling practical backscatter communication for on-body sensors. In *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference*, pages 370–383. ACM, 2016.