SyncScatter: Enabling WiFi like synchronization and range for WiFi backscatter Communication

Manideep Dunna, Miao Meng, Po-Han Wang, Chi Zhang, Patrick Mercier, and Dinesh Bharadia, *University of California, San Diego*

https://www.usenix.org/conference/nsdi21/presentation/dunna

This paper is included in the
Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation.
April 12–14, 2021

978-1-939133-21-2

Open access to the Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation is sponsored by

SyncScatter: Enabling WiFi like synchronization and range for WiFi backscatter Communication

Manideep Dunna, Miao Meng, Po-Han Wang, Chi Zhang, Patrick Mercier, Dinesh Bharadia

University of California, San Diego

Abstract

WiFi backscattering can enable direct connectivity of IoT devices with commodity WiFi hardware at low power. However, most existing work in this area has overlooked the importance of synchronization and, as a result, accepted either limited range between the transmitter and the IoT device, reduced throughput via bit repetition, or both. In this paper, we present SyncScatter, which achieves accurate synchronization with incident signals at the IoT device level while realizing maximum possible sensitivity afforded by a backscatter link budget. SyncScatter creates a novel modeling framework and derives the maximal optimal range and synchronization error that the receiver can tolerate without significant performance compromises. Next, SyncScatter builds a novel hierarchical wake-up protocol, which, together with a custom ASIC, achieves a range of 30+ meters and the peak throughput of 500Kbps, with an average power consumption of 30μW.

1 Introduction

Ubiquitous wireless network coverage is required to enable the next-generation of Internet of Things(IoT) devices. In smart homes, offices, industrial environments, and more, WiFi is by far the most ubiquitous form of connectivity. However, enabling WiFi connectivity at the IoT device level requires high power consumption - to the point where most such IoT devices must be either plugged into wall power, must use large and frequently re-charged batteries, or simply cannot afford to transmit data at high average throughput [13, 22].

Recent work has proposed using backscatter communication techniques to reduce power, which forgoes direct WiFi signal generation by instead modulating data on top of ambient WiFi transmissions generated by existing access points (APs) [10, 40]. There are three components to backscattering systems: 1) the transmitting radio which generates the excitation signal; 2) the IoT device which reflects the incoming signal by encoding its information, and 3) a receiving radio (WiFi-compatible) which receives the packets and decodes the data from the IoT device. By avoiding any signal generation or amplification directly at RF, the power consumption of backscatter communication can be low - on the order of microwatts - such that small energy harvesters or batteries can directly power the IoT devices.

Existing work on WiFi-backscattering can be broadly classified into two major categories: the first set of WiFi-backscattering is inspired by the RFID style of backscatter communication systems [8, 21], wherein a tone-generator is deployed as the excitation radio. This gives the IoT device the freedom to begin backscattering at any time it likes, along with the freedom to create any backscattered waveform like WiFi to encode its information. This tone-based backscattering approach requires additional custom hardware - the tone generator - beyond commodity WiFi hardware, and thus deployment timelines and costs can be appreciable [21].

In contrast, the second set of works [40, 42] leverage existing WiFi infrastructure for both the excitation and receiving radios, therefore requiring no additional infrastructure deployment. In this approach, an existing WiFi radio's transmission is used as excitation signal, and the IoT device modulates the underlying data in the excitation signal in a specific manner according to the IoT device's data which is then reflected back to the environment. Properly backscatter modulated signals will retain a WiFi-compatible form, and therefore can be decoded by another WiFi radio. Ideally, the IoT device should synchronize itself to the incident signal, such that it knows precisely when to perform backscatter modulation. To ensure the superimposed data (i.e., the data from the incident source and the IoT device data) is readable by a commodity WiFi receiver, this synchronization should be down to the symbol level. Unfortunately, no prior work in WiFi backscatter synchronizes down to the symbol level [20, 21, 40].

Synchronizing to the symbol level with high-accuracy is challenging, as all ISM, including WiFi transmissions, are made up of complex digital waveforms at fairly high data rates. A large literature has worked on optimizing the power consumption of the synchronization routines in WiFi transceivers, which still takes considerable power [5]. Furthermore, the power consumption of synchronization increases exponentially to make it work at lower incoming signal power. Therefore, prior WiFi backscattering work such as Hitchhike [40] employs a very simply energy-detecting synchronization scheme that consumes low power but can only synchronize to an accuracy of 2μsec at an input power of $-20dBm$. Since the symbol rate in 802.11b WiFi is 1μsec, this means Hitchhike will effectively begin backscatter modulation at a random location within a symbol, which ultimately limits its achievable distance to be no more than 6m from the transmitter. Unfortunately, this comes directly at the cost of decreased achievable throughput, increased inter-symbol interference(ISI), and ultimately reduced communication range.

In this paper, we present SyncScatter, which is the first

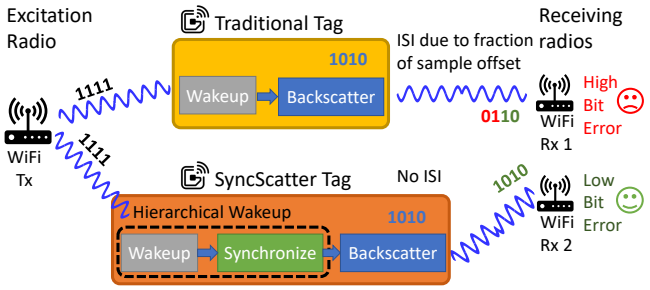


Figure 1: Shows a traditional tag form [40] which backscatter signal without accurate synchronization, leading to higher BER. In contrast the SyncScatter tag accurately synchronizes to the incoming signal.

integrated circuit-based backscattering platform that can enable symbol-level synchronization through a hierarchical wakeup and synchronization protocol, shown in Fig. 1, which works up to theoretical sensitivity levels. Such symbol synchronization enables longer range, higher-throughput, and more reliable backscatter communication than prior art for all forms of communication (not just WiFi). Here, we specifically built and prototyped SyncScatter to demonstrate the first fully-WiFi-compatible symbol-level synchronized, long-distance, extremely low-powered backscatter system. Furthermore, SyncScatter can support multiple IoT devices to co-exist without interfering with each other. SyncScatter is designed on a custom ASIC, enabling ultra-low-power consumption.

In order to bound the design space, we begin the paper by asking some fundamental questions: how accurate does the synchronization need to be? How far away from the transmitter can we work? What sensitivity level should we target at that range? How do we do all of this while keeping power low? To answer the first question, we present analysis wherein we add increasing amount of synchronization error and observe the performance (SNR vs BER), and choose point where the gains are incremental beyond it. Then, we leverage FCC specifications on maximum available transmit power along with the minimum SNR required to decode a certain ISM backscatter, accounting for loss in backscattering and the noise figure of the corresponding receiver, to derive the maximum path loss, and therefore distance, a backscatter system could potentially work at. This analysis also gives us the sensitivity level needed at the backscattering tag, which as we will show turns out to be $-35dBm$. Importantly, we discuss how improving sensitivity beyond this level is wasteful. This is the first analysis of its kind that combines path loss, synchronization accuracy, and sensitivity.

The next natural question for SyncScatter is how to achieve the required tight synchronization accuracy and desired high sensitivity while consuming microwatts of power? To keep power low, a direct envelope-detector (ED) approach is typically used; however, such a system's sensitivity typically reduces with increasing bandwidth. Since achieving a high synchronization accuracy requires high bandwidth, this poses a direct trade-off between accuracy and sensitivity. The only way to break this trade-off is to add RF amplification before the ED. However, this can cost significant power - upwards of 100s of microwatts.

To break this trade-off, our key insight is to create a two-stage, hierarchical wake-up and synchronization protocol, wherein a first stage (the wake-up receiver) is designed with single-digit microwatt power and leverages low-bandwidth energy detection to simply wake-up the tag at approximately the right time, at which point a second stage (the synchronization receiver) uses higher-power active RF amplification to enable the desired sensitivity at the desired bandwidth, but is turned on only for a short time to synchronize, and is powered down immediately post synchronization. SyncScatter creates a new protocol where two packets with controlled length are sent apriori to backscattering. The time duration of the two packets encodes the tag's identity, which results in an enable signal from the first stage wake-up receiver. The second stage turns on just before the start of the backscatter payload packet, samples the incoming signal at high bandwidth, looking for the beginning of the packet and the symbol boundary, and then promptly goes to sleep. Once symbol-level synchronization is achieved, the backscatter modulation logic reflects the incoming signal by overlaying its data in a synchronized fashion.

SyncScatter specifically builds an RF integrated circuit and hardware design for the entire hierarchical wake-up protocol, along with single-sideband backscattering circuits, which can backscatter any ISM 2.4 GHz signals. SyncScatter's WiFi transmitter and receiver are implemented using open-wrt [25] on TP-Link devices. SyncScatter is evaluated in indoor office environments to achieve the following results:

- SyncScatter achieves a sensitivity of up to -35 dBm via the custom integrated circuit, with a synchronization accuracy of 150 ns, which enables a 30+ meter link operation as measured in a regular office environment. As a result, the longer wake-up distance offered by SyncScatter allows the use of WiFi APs deployed in a typical home or office environment without requiring additional smartphones, unlike in HitchHike [40].
- SyncScatter tag enables symbol-level synchronization at very low power consumption by utilizing a hierarchical wake-up receiver with a false negative rate of 10^{-3}.
- SyncScatter tag supports backscatter communication over a wide range of the transmitter(Tx) to tag and receiver(Rx) to tag distances whose product is $\leq 169\ m^2$, i.e., 13m from Tx and 13m from Rx or 33m from Tx and 5m from Rx.
- SyncScatter supports multiple tags running concurrently and supports 802.11b waveforms, modulating at symbol level providing peak bit-rates of 500 Kbps.

2 Synchronization & Sensitivity Requirements

This section introduces a model that helps scope out requirements for the custom IC designed for WiFi backscattering. The methodology presented herein could also be used to

design any ISM backscatter systems.

2.1 Synchronization Requirements for WiFi

Synchronization is at the heart of all communication systems, and must be thought through carefully, even for backscatter systems (at least, ones that do not use tone generators, like fully WiFi-compatible systems). In this sub-section, we specifically discuss the need for synchronization and establish the minimum required synchronization accuracy, which is needed for minimal performance reduction in a fully WiFi-compatible backscatter system.

To describe the synchronization problem we are solving, we will start with a brief discussion of the problem with a figure, specifically on a representative 802.11b signal in Fig. 2. To backscatter a valid 802.11b signal, the tag performs code-word translation on the ambient 802.11b packets, similar to [40]. The tag embeds its data on the ambient 802.11b WiFi packet by changing the phase of each of the 802.11b symbols in the packet. The resulting backscattered signal is a product of the incident signal and the tag's phase modulation. Therefore, changing each symbol's phase ensures that the chip sequence on each symbol retains the 11-bit barker code and backscatters a valid 802.11b signal, which is then decodable by the WiFi receiver without inter-symbol interference shown on the right.

As observed in [40], the HitchHike tag, and in fact all past work that backscatters ambient signals, do not accurately synchronize with the incoming transmitter signal and therefore applies code-word translation with incorrect boundaries shown in the Fig. 2 left. Misalignment between the backscatter symbol timing and the original symbols in the 11b packet will start to change the barker code, which hurts the signal to interference ratio and, therefore, hurts the receiver's ability to decode the backscatter packet and can result in errors.

To quantitatively understand the impact of synchronization, we emulate the backscatter system wherein we transmit the reflected packet using an SDR while intentionally adding an increasing amount of synchronization error from 0 ns to 625 ns as shown in Fig. 3a. The synchronization errors are added after the header of the transmission, as the backscattered payload is applied only after the header. The transmission is received using an off-the-shelf WiFi AP. We measure 10000 packets with backscatter payload in a total of 1 million bits to generate the SNR to BER plot. As we can observe, synchronization errors up to 150ns do not lead to significant BER degradation. However, errors beyond this can lead to 7dB or more degradation.

A natural question is how did past work resolve this issue. Due to power-related issues that we will discuss shortly, prior work such as [40] achieved synchronization accuracies of anywhere from 1 to 10 μsec, which is well beyond the symbol period. This means that the backscattered signal modulated on top of the existing WiFi packet will have a random phase offset for each packet. To combat this problem, [40] uses a

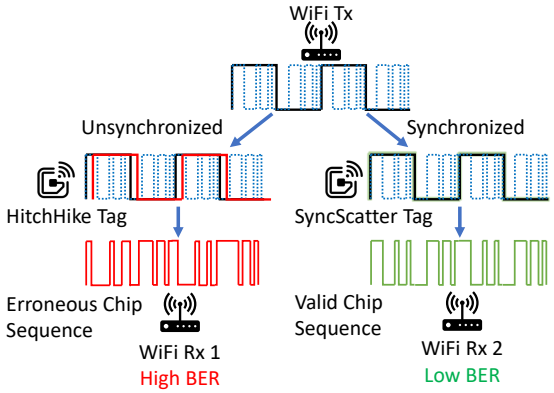
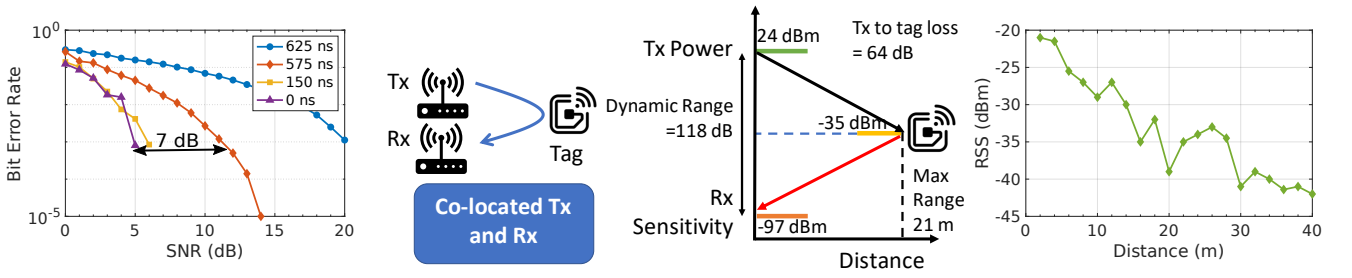


Figure 2: Impact of in-accurate synchronization, and resulting loss in signal quality at the receiver.

preamble to help the receiver find the start of backscatter data in the packet and decode the tag data. Further, to ensure proper decoding of tag data, each tag data bit is repeated multiple times(repetition coding), reducing the available throughput. Said differently, it would lose significant SNR gain due to lack of synchronization. A more severe consequence is that the CRC(cyclic redundancy check) of the packet often fails with past work. Instead, we can ensure CRC checks can be met with proper synchronization, enabling more WiFi cards as receivers while simultaneously enabling higher average channel throughput.

Generalization of Synchronization requirements to other wireless standards: An obvious next step is to know the synchronization requirements for backscatter systems based on different wireless standards like BLE(Bluetooth low energy) and 802.11g WiFi. In BLE transmissions, a data bit 0/1 is encoded as different frequency modulated sine tone signals. Backscatter tag modifies the frequency of sine tones present in the BLE symbols to encode backscatter data on top of BLE packets. The backscatter encoder must know the symbol boundaries to ensure that backscattering is successful. Otherwise, the backscatter data is spread on two consecutive symbols, and the receiver fails to decode the packet. Similarly, 802.11g WiFi backscatter systems encode data on top of an OFDM symbol by changing the signal phase. Hence the backscatter encoder must be aware of the OFDM symbol boundaries. To understand the synchronization delay's impact, we perform Matlab simulations of backscattering BLE and 20MHz standard OFDM signals as given in [42] and introduce synchronization(sync) delay while finding symbol boundaries. Our simulations reveal that BLE backscatter loses 4 dB SNR at 10^{-3} BER for 100 ns sync delay and can tolerate up to 50 ns sync delay(More details in Appendix 9). On the other hand, OFDM backscatter can handle up to 1000 ns sync delay and lose more than 10 dB SNR at 10^{-3} BER for 1200 ns sync delay. These observations suggest that symbol-level synchronization is essential for backscatter systems based on other protocols as well.



(a) BER-SNR curve

(b) Co-located WiFi Tx and WiFi Rx with a SyncScatter tag

(c) Rx Signal Strength with Tx-Tag distance

Figure 3: **(a)** BER-SNR curve for different amount of synchronization errors, **(b)**Maximum distance analysis and operating point for IoT device sensitivity assuming a co-located Tx and Rx, **(c)** Received Signal Strength (RSS) measured at the SyncScatter tag vs. distance.

2.2 Sensitivity of the backscatter tags

For an IoT device to backscatter its data, it needs to detect the incoming signal from the excitation radio. Prior work has shown very short distance backscatter from excitation radio to the tag, for example, less than 6 meters for WiFi backscattering [20, 40]. Given the flexibility in the design parameters with integrated circuit design, we would like to understand better the sensitivity for which the backscatter tag should be designed to optimize for range while minimizing power consumption.

At the outset, the above question looks ill-posed. To simplify the above question and perhaps get a more coherent answer, let us take an example of the same 11b signals transmitted at 1 Mbps. To simplify the above analysis, we would assume the worst-case mono-static backscatter communication scenario, i.e., the transmitting radio and the receiving radio are co-located. The challenge with the backscatter system is that it suffers from two-way path loss. Specifically, the transmitter to the tag suffers a path loss of $\frac{1}{d^2}$. Then the signal is re-radiated back from tag to the receiver, which suffers a multiplicative path loss of $\frac{1}{d^2}$, which implies $\frac{1}{d^4}$ path loss (or additive in dB). Said differently, it would suffer the same loss as traveling quadratic distance.

With that analysis in place, we can leverage the maximum possible dynamic range by operating at the highest possible transmit power and minimum receiver sensitivity at which we can decode the packet. For WiFi at 2.4GHz, the maximum average transmit power is 24 dBm (11b has a peak power of 30 dBm and 6 dB of PAPR), while a receiver sensitivity to decode 1 Mbps is at -97 dBm as shown in Fig. 3b. Given a tag can have ideal insertion loss of 3dB, this provides a dynamic range of 24 dBm $-(-97$ dBm$) - 3$ dB $= 118$ dB for decode-ability. Assuming a worst-case range condition where the tag communicates with a co-located transmitter and receiver AP, this suggests operation with 59 dB of one-way path loss, for a worst-case incident signal power 24 dBm -59 dB $= -35$ dBm.

To find the path loss in indoor environments, we measure the received signal strength with increasing distance between the transmitter and IoT device, up to a point where the received signal strength at the IoT device goes below the threshold of -35 dBm, providing us the range up to which we expect the backscatter to work. Fig. 3c plots the received power with increasing distance, and received power or sensitivity of -35 dBm would provide the maximum range benefit, i.e., from 15 – 21 meters, one-sided range. Note that prior systems have achieved -15 dBm sensitivity, with 6 meters working range [40].

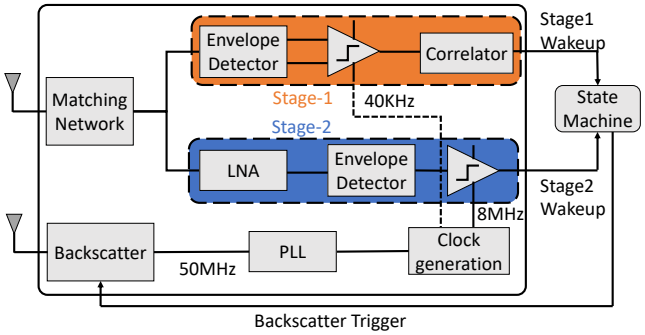


Figure 4: Design Overview of SyncScatter.

In summary, we need to achieve a sensitivity of -35 dBm beyond which the gains are incremental and concurrently achieve synchronization accuracy of 150 ns, all while keeping low power. Achieving this specification at the micro-watt level is extremely challenging.

3 SyncScatter Tag Design Overview

In this section, we describe the architecture of SyncScatter's tag and show how it can wake-up to properly-designed incident WiFi signals, perform symbol-level synchronization to said incident WiFi signals, and then perform backscatter modulation. This is accomplished by three separate subsystems: a wake-up receiver, a sync stage edge-detector, and a backscatter modulator as shown in Fig. 4.

In most prior works [40,42], the wake-up receiver and sync stage are combined into a single circuit. i.e., a single circuit is responsible for determining if an appropriate WiFi signal is incident on the tag and then indicating to the tag when to begin backscatter modulation. This is, however, a problematic approach if the tag desires low-power and sufficient sensitivity and synchronization accuracy, as these items all trade-off directly with one another. Thus, breaking wake-up functionality apart from synchronization functionality via the proposed hierarchical approach can serve to break this trade-off, enabling a low-power yet sufficiently sensitive and accurate design.

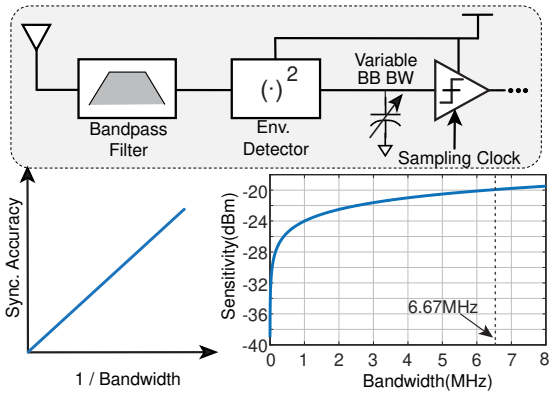


Figure 5: Analysis on Direct-Envelope Detector (ED) Structure

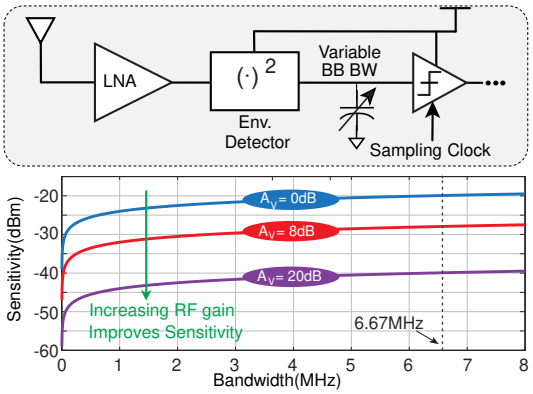


Figure 6: LNA Based ED Improvements

3.1 Sync Stage Receiver

As aforementioned, the best we can do given FCC WiFi transmitter power limitations and the best achievable WiFi receiver noise figure is to achieve a sensitivity at the tag of approximately −35 dBm. Achieving a sensitivity better than this does not improve the system's performance or range in any meaningful manner and is thus just wasteful. At the same time, we need to detect the symbol boundary of an incident WiFi packet with an accuracy of at least 150 ns. Again, doing better than this does not meaningfully improve performance. Thus, the design space here is: achieve these sensitivity and synchronization accuracy specifications while consuming as little power as possible.

To find the accurate symbol boundaries, we can use the fact that multiple symbols together constitute a packet. If we can somehow find the exact time instant at which the packet starts, we can determine the symbol boundaries by keeping track of the time elapsed from the beginning of the packet. The packet boundary is indicated by a change in the signal power received, and it can be used to find the start of the packet. Assuming we have already woken-up to a pre-specified WiFi signature (as described in the following section), we can then measure the instantaneous signal amplitude by passing the signal through an envelope detector (ED) and monitoring its envelope for a strong rising edge representing the beginning of the packet that will be backscatter modulated.

Before we present how we can perform necessary synchronization, it is first instructive to provide a brief overview of very low-power energy-detecting radio receivers. The simplest and lowest power receiver directly connects an antenna to an ED, whose output is low-pass filtered and then sampled. Fig. 5. shows a simplified block diagram of this approach. This approach's main benefit is that the ED can be passive and thus consumes zero power; this circuit's only power consumption is due to the sampler/comparator, which can be in the low single-digit microwatt regime. Note that this approach energy-detects everything at its input, and thus there is usually a filter at the input to ensure out-of-band interferers do not get demodulated.

The main design parameter in such a receiver is the baseband bandwidth, BW_{BB}, set by the combination of the effective resistance of the ED itself and its load capacitance as a simple first-order *RC* filter. The larger the baseband bandwidth, the more precise an ED will be able to detect a rising packet edge, and therefore the better the synchronization accuracy will be. To first order, this is a one-to-one trade-off, as shown in Fig. 5, assuming that the comparator is sampling at the Nyquist rate (i.e., 2× the baseband bandwidth). To achieve a synchronization accuracy of 150 ns, we would need a baseband bandwidth of at least 6.7 MHz.

However, the complication here is that the larger the baseband bandwidth, the larger the noise bandwidth becomes. As described in [18, 34], with no gain in front of the ED, the receiver's sensitivity is typically dominated by the noise of the ED itself. Interestingly, RF noise is immaterial in such a scenario because a passive ED's noise is so much larger than all downconverted RF noise. As a result, increasing the baseband bandwidth directly increases the noise, which degrades the sensitivity with a $5\log(BW_{BB})$ trade-off (where $5\log()$ instead of $10\log()$ is used to account for the squaring function of the ED). Specifically, the sensitivity of a direct-ED receiver is given by equation 1.

$$P_{sensitivity} = \frac{20}{k_{ED} * A_V^2} \sqrt{BW_{BB} * PSD_0 * SNR_{MIN}} \quad (1)$$

where k_{ED} is the scaling factor of envelope detector, A_V is the front-end voltage gain (equals to 1 for a direct-ED receiver), PSD_0 is the output-referred baseband noise, and SNR_{MIN} is the required minimum signal-to-noise ratio. The result of this equation is plotted in Fig. 6 for representative values of k_{ED}, PSD_0, and SNR_{MIN} to be 250/V, 300 nV^2/Hz and 6 dB, respectively. Achieving a synchronization accuracy of 150ns requires a baseband bandwidth of 6.7MHz, which, as shown per these numbers, permits a sensitivity of at best −20 dBm. This is obviously unacceptable.

Since we cannot further reduce the noise floor of a passive ED, the only recourse is to either provide more voltage gain before the ED or build an active ED to reduce its noise floor. It turns out that building an active ED with sufficiently low noise is not only not easy but also only helps with a $5\log()$ benefit. On the other hand, providing voltage gain before the ED helps

with a $20\log()$ benefit and is thus far more attractive.

In fact, it is possible, even at 2.4 GHz, to provide some amount of voltage gain completely passively through an impedance transformation network. For example, the π network (shown in Fig. 8a later) can take the 50 Ω antenna impedance and transform it to a 300 Ω impedance, theoretically giving $20\log(\sqrt{Z_o/Z_i}) = 8$ dB of "free" voltage gain without consuming power. As shown in Fig. 6, adding 8 dB of voltage gain improves the sensitivity by 8 dB for a net sensitivity of -28 dBm. This is still not good enough. Considering that the detector's impedance is of the order of $100k\Omega$, can we get more voltage gain if we build a better matching network? Unfortunately, the ability to do this is limited by the low-quality factor of components at high frequencies, along with parasitics - where for example, even 0.1 pF of parasitic capacitance presents as 600 Ω of impedance. Likewise, a 0.8 nH high-Q inductor from Coilcraft (0402DC-N80XR, Coilcraft, Illinois, USA) has $Q = 110$, which at 2.4 GHz is a series resistance of 25 mΩ, which limits Z_o to 300 Ω. Thus, it is very difficult to get much more than $\sim$8 dB of passive voltage gain at these frequencies.

As a result, the only remaining way to improve sensitivity is to add active RF gain. This is typically undesired by designers of backscatter tags, as the main purpose of doing backscatter modulation is to avoid having to build active circuits operating at RF since they tend to consume significant power. For example, in this work, we have built a custom RF amplifier into the integrated circuit, which provides a gain of 12dB for a power consumption of 240 μW. With the matching network, the provided gain improves the sensitivity by 20dB, which now meets the desired -40 dBm sensitivity specification (with some margin). However, while still significantly lower than the 10's to 100's of mW a typical WiFi transceiver would consume, the power consumption is still higher than desired.

We can now get to the key insight provided by SyncScatter: the high bandwidth needed for synchronization only needs to occur when we know we are about to backscatter - that is after we have already woken up. As a result, we only need to turn this RF amplifier on for a short amount of time to detect the rising edge of the packet to be backscatter modulated. We can shut-off the RF amplifier before and after this event. By duty-cycling the amplifier in this manner, we can cut down its average power consumption significantly. For example, the sync receiver needs to be turned on only for 50us throughout the 500 μs wakeup + 2000 μs data packet duration. The duty-cycled power, in this case, turns out to be $\frac{50}{2500} \times 240\mu\text{W} = 4.8\mu\text{W}$.

Again, the key insight in SyncScatter is that we can decouple the precise symbol level synchronization from the wake-up functionality, so that we can spend high power momentarily during symbol synchronization to achieve the desired bandwidth and sensitivity, while duty-cycling this down to low average power at times when we are not expecting

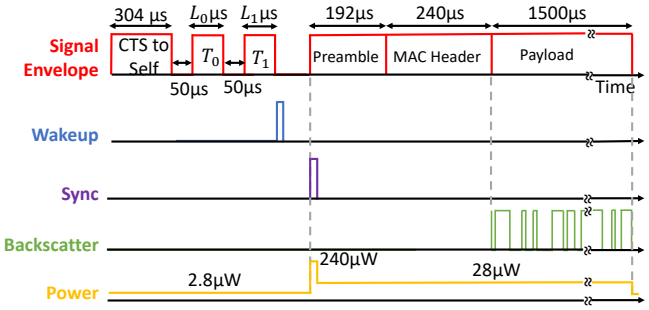


Figure 7: Timing Digram of SyncScatter.

the packet edge. Unlike other works which combine wake-up and synchronization in a single circuit, this work proposes a hierarchical method - where a low-power wake-up receiver is used to detect a WiFi compliant signature that indicates the next incident packet should be backscatter modulated, which then hierarchically turns on a sync stage receiver to provide the necessary high-bandwidth synchronization accuracy. This hierarchical approach, of course, only helps if we can design a wake-up path with low power and sufficient sensitivity, as will be described next.

3.2 Wake-up Receiver – First Stage

The wake-up stage's goal is to monitor the RF spectrum for a pre-specified set of packets that indicate the next packet is the one to be backscatter modulated. This should occur with the same sensitivity as the sync stage receiver, but ideally at much lower power since it must be on for potentially long durations of time while waiting to be triggered by a WiFi AP. If the wake-up stage is sufficiently low power, and the sync stage only needs to operate over a small duty-cycle, then the overall hierarchical approach can consume low average power.

The logical question is then: how can the wake-up path consume lower power and yet achieve the same sensitivity as the sync stage receiver? The answer is that the wake-up stage does not require the same amount of bandwidth since it is not being used to perform symbol-level synchronization. Reduced baseband bandwidth enables a reduction in the required amount of pre-ED RF gain to the point where no active RF amplification is needed, all while still meeting the desired -35dBm sensitivity level (with margin).

The wake-up pattern is constructed as follows. A WiFi-compatible identifier is transmitted by a WiFi AP first consisting of a CTS-to-self to temporarily reserve the channel, followed by the transmission of two packets, T0 and T1, with pre-determined lengths corresponding to the IoT tag that is supposed to be woken up. This sequence is illustrated in Fig. 7. Multiple tags can then be uniquely woken up by choosing different T0 and T1 packet lengths. At the tag level, the wake-up stage uses an 8 dB passive voltage gain network that is directly connected to an ED. The ED energy detects the entire packets and samples the energy with a comparator. Since the packet lengths are restricted to a minimum 50 μs, the required ED baseband bandwidth is 20 kHz. As shown

previously in Fig. 6, with 8 dB of passive voltage gain, this results in a sensitivity of −40 dBm - which is the desired level (with margin). The comparator's output passes into a counter that counts the number of logic '1's, given by the presence of a packet (vs. a logic '0', which would occur in the inter-packet interval), at a sampling rate of 40 kHz. If the expected number of 1's and 0's occur in the right order, the wake-up stage's output triggers the sync stage receiver. Importantly, this wake-stage is achieved with a purely passive ED that consumes zero power. As a result, the only power here is that of the comparator, correlator, and clock generator (Fig. 4), which consumes only 2.8 μW during active mode.

3.3 Backscatter Communication

Once the tag has woken up and the sync stage identifies the exact packet start instant, the system starts backscattering with zero data. This ensures that the incident WiFi packet's header is backscattered to a different WiFi channel for reception by another WiFi AP without any modification using a Single side-band (SSB) modulation technique similar to [40]. While this is occurring, the tag counts the number of clock cycles until the header is complete, after which it can begin to introduce its data into the backscatter modulator. The backscatter data is XORed with the incident 11b symbol data, also known as code-word translation similar to [40]. The backscatter data is recovered at the receiving AP by XORing the received data again with the original 11b symbol data.

3.4 Putting it all Together

In the subsection, we discuss the end-to-end life-cycle of data packet exchange from an IoT device to the WiFi AP. A WiFi AP with the firmware support to transmit an excitation signal transmits a CTS-to-self packet to reserve a slot of 5 milli-second. Next, the transmitting AP transmits the two packets T0 and T1, whose lengths are a multiple of 25μsec. The tag notices a special pattern of three packets using the wake-up stage receiver by measuring the duration of CTS-to-self, T0, and T1 packets.

Downlink to a Specific Tag: Each IoT device is pre-coded with the lengths for T0 and T1 (akin to a destination address), which is the tag's identity. The finite state machine (Figure 4) at the tag continuously runs the wake-up receiver to look and match the three packet durations consuming 3 μW, with the trigger level of the reference voltage for ED set to -40 dBm. Whenever the three measured lengths match with the precoded sequences, it enables the specific IoT device to receive the downlink data.

A fixed number of bits are allocated for downlink in the finite state machine. The AP transmits the packets with varying lengths to encode the downlink data with 25 μsec granularity. The wake-up receiver at the IoT device uses the packet length to decode the downlink message. Therefore, the downlink data-rate supported is 40 Kbps. We reserve 3 bits for downlink in our implementation, which are used to set the reflection side-band upper or lower.

Uplink from the Tag: Upon completing the downlink, the tag fires up the sync receiver at the IoT device to acquire synchronization to uplink the data. The AP transmits a longer packet which we use to uplink the data. The tag synchronizes to the receiving packet with 150 ns accuracy, assuming incoming power is higher than -40 dBm. The tag starts backscattering at 50 MHz without any data, as soon as it receives a trigger from the sync receiver. Back-scattering with no-data ensures the incoming packet is reflected on channel 11, assuming transmission was on channel 1. The receiving AP on channel 11 starts receiving the packet. It successfully receives the PHY and MAC header of a total of 432 μsec. Upon completion of 432 μsec, the IoT device starts backscattering data, which is compliant to WiFi standards, as discussed in the next section. The receiving AP decodes the packets successfully, with CRC matching ensuring the packet is reported to the higher layers. The receiving AP XORs its data with the transmitted data in the cloud to recover data from the IoT device, thus connecting the IoT device to the AP.

3.5 Working with COTS WiFi

In the previous sub-section, we assumed certain capabilities for COTS WiFi. We will present how our system is compatible with using commercial WiFi transceivers. Specifically, we explain our test setup to receive the backscatter packets and decode them. We also discuss how the physical layer modulation schemes in the 802.11b protocol affect the bit-data processing at the backscatter IC and the receiver end.

Generating the wakeup pattern: The wakeup pattern is an On/Off pattern made up of WiFi compliant packets. It contains two WiFi packets separated by a DCF interframe spacing(DIFS) gap between two successive packets. The WiFi packets are broadcast packets that are of 107us duration, each separated by 50us corresponding to the DIFS gap. We note that typical 802.11b data packet sizes are of the order of few milliseconds, and $500us$ (CTS-to-self + wakeup pattern) duration do not add significant overhead for the backscatter communication.

Scrambling and Differential encoding: 802.11b WiFi APs randomize the data by scrambling it before transmission. At the receiver end, a de-scrambler is used to de-randomize the data and obtain the original bits. So, the backscatter data bits have to be scrambled before they are transmitted. The backscatter bits are scrambled using a 7^{th} order polynomial implemented as a feedback shift register initialized with a fixed seed [3]. Since the seed is fixed in 11b transmissions, all the tags can be programmed with this seed to facilitate data scrambling. Following the scrambling operation, the bits have to be encoded in a differential manner following the 802.11b PHY differential modulation scheme so that the receiver can decode the backscatter bits correctly.

4 Hardware and IC Design

Each SyncScatter tag is built upon a custom integrated circuit that was fabricated in TSMC's 65nm GP process. Our

chip design is inspired by previous works [35, 36], which like other works HitchHike [40] lacks synchronization at the necessary sensitivity levels. Our IC design includes the proposed wake-up, sync receiver, and SSB backscatter modulator. The remainder of this section describes the chip design and operation, along with board-level integration efforts.

IoT Device: Daughterboard: The 65 nm die is directly mounted onto a custom printed circuit board, hereafter referred to as the daughterboard. The daughterboard(Figure 9b) routes all power and digital control traces to headers, which interface with a larger motherboard. Although not strictly necessary, the daughterboard is fabricated to utilize two RF antennas for ease of initial design: one for the wake-up path and one for the backscatter-path. However, we can use a simple switch to include both these paths to interact with a single antenna. The chip is clocked primarily from a 16 MHz crystal oscillator, with the oscillator circuit integrated on the chip, and the crystal soldered onto the daughterboard.

4.1 Wake-up receiver:

The wake up receiver consists of passive voltage gain directly feeding an ED and then to comparator which is sampled at 40kHz sampling rate as shown in Fig. 4.

Passive Voltage gain: At power levels of -40 to -30 dBm, the voltage seen at the antenna port are on the order of 5-20 mV. A passive voltage gain network is included to boost this according to the benefits outlined in the previous section. In this design, an *CLC*-based π matching network is employed, as shown in Fig. 8a, which provides 8 dB of voltage gain. The maximum achievable gain is limited by the quality factor, Q, of the constituent components at 2.4 GHz, along with the input impedance of the ED. In the current implementation, the employed inductor's Q is 110.

Envelope detector and comparator: Since the antenna is single-ended, it's easier to achieve high passive voltage gain with a single-ended matching network. Thus, the ED should also be single-ended. However, in general, it is better to perform baseband signal processing in a differential manner. Considering that the input impedance, output referred noise, and conversion gain are all important parameters to optimize in ED design, a multi-stage fully passive ED design is employed. In particular, a single-ended-to-differential Dickson-based topology is selected, thus acting as a pseudo-balun.

The ED's output bandwidth is controlled in part by the body bias voltage, which controls the ED's effective resistance, along with a variable capacitor. For the stage one design, a bandwidth of 200 kHz is targeted, which is sufficient to enable detection of the presence of T0 and T1 packets and their lengths, though without precise synchronization accuracy.

The pseudo-differential outputs of the stage one ED then feed into a differential comparator based on a Strong-ARM regenerative latch topology. This comparator effectively acts as a 1-bit analog-to-digital converter (ADC), and thus to extract useful timing information, it must be oversampled. As a result, it is clocked at 40 kHz. This clock is derived directly from the on-board crystal, after an on-chip division by a factor of 400. The comparison threshold voltage is tuned by externally controlling the bulk voltages of the input pair of the preamplifier implemented by a *gmC* integrator.

4.2 Sync Receiver

Once the first stage has determined that packets T0 and T1 have been transmitted, we turn on the sync receiver, and its purpose is to look for the rising edge of the subsequently transmitted packet header. It must do so with a bandwidth and sampling frequency greater than 6.67 MHz to meet the 150 *n*sec timing accuracy requirements. Since increasing the envelope detector's bandwidth will, without any other action, decrease sensitivity, this is countered by adding some active gain in front of the envelope detector.

The schematic of the sync receiver is shown in Fig. 4. Here, the stage two design shares the same antenna and passive voltage gain network as the first stage. But, instead of going directly into an envelope detector, it first goes into a low-noise amplifier (LNA). The LNA is designed to support a gain of 11 dB and an active-mode power consumption of 240 μW. If this second stage were on all of the time, this would be a significant power burden to the entire system. This is the beauty of the hierarchical wake-up feature: the LNA only needs to be turned on after the first stage wake-up receiver triggers reception of the appropriate signature. Otherwise, the LNA is nominally in a low-power sleep state. As a result, its average power consumption is extremely low - limited by the frequency of activation by stage one.

A schematic of the LNA is shown in Fig. 8c. A current reuse common source amplifier is implemented to achieve the desired gain with sufficiently low noise. Its output then feeds into a second envelope detector, optimized in a similar manner to wake-up receiver, though in this case for a bandwidth of 32 MHz to ensure a highly accurate rising edge timing. To compensate for the higher bandwidth, an active ED with decreased noise and increased conversion gain is employed in place of the passive ED in the wake-up receiver. A similar comparator as in the wake-up receiver with externally tunable reference voltage is used after the envelope detector, though in this case sampled at 8 MHz, which is sufficient to achieve a 150 ns synchronization time.

4.3 Clock generation:

The 16 MHz crystal oscillator clock is divided on chip into 8 MHz, 2 MHz and 40 kHz to be used by sampling clocks for sync receiver, PLL reference clock and sampling clock for wake up receiver, respectively. To drive the backscatter modulator, an integer-N PLL is adopted with a 2 MHz reference frequency and a dividing ratio of 25. The voltage controlled oscillator is implemented via a current starved ring oscillator with tunable current to ensure stable clock generation against PVT variations and it consumes 1.5 μW power.

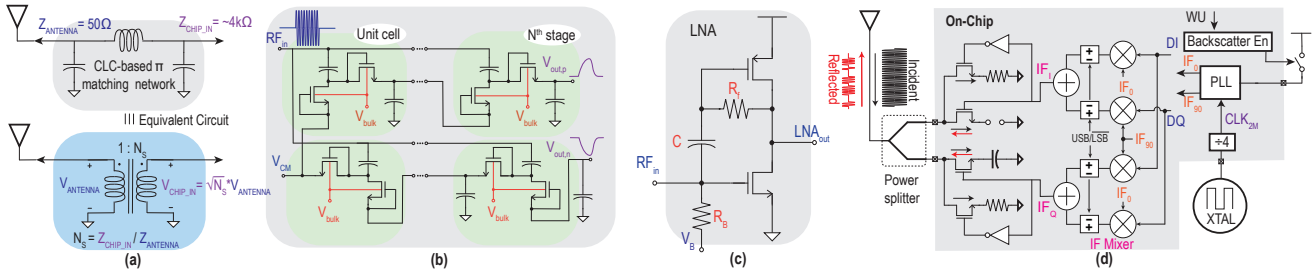


Figure 8: **(a)** CLC-based π matching network, **(b)** Schematic of passive pseudo-balun ED,**(c)** Schematic of LNA, **(d)** Transmission-line-less SSB QPSK Backscatter.

4.4 Backscatter circuit:

Single-side-band QPSK modulation is achieved by the backscatter circuit via the approach shown in Fig. 8d. Here, a power splitter/combiner breaks incident wireless power into two separate paths. The first path meets one of two possible termination conditions: 50 Ω or $Z_{L,0}$, depending on the state of $\text{IF}_{\text{OUT,I}}$. If it meets 50 Ω, then all of the incident RF energy is absorbed by this resistor, and no reflection is generated. However, if it meets $Z_{L,0}$, which is designed to be an open circuit in this implementation, then all of the incident RF power will be reflected back through the power combiner and to the antenna for re-radiation purposes. A similar situation occurs on the other path of the power splitter/combiner, which is terminated by either 50Ω or $Z_{L,90}$, depending on the status of $\text{IF}_{\text{OUT,Q}}$. If $Z_{L,90} = -j \times 50$, then all the signal is reflected, though in this case with a 90° phase shift. If I/Q mixers drive the two paths with 90° separated IF clocks, most of the energy will result in a single sideband, in the same way that a single-side-band mixer operates. Importantly, this approach only requires $Z_{L,90}$ to be a 1.2 pF capacitor - which is very easy to design on chip. This is in contrast to prior work, which required a transmission line to generate the requisite 90° phase shift [40].

4.5 Mother board

The prototype motherboard contains various voltage regulators for the chip and an ultra-low-power microcontroller for generating the data sequence with the correct timing. The voltage regulators generate all the different supply voltages required for the chip. To tune the LNA sensitivity and comparator thresholds, DACs are used to generate the variable voltages, which are controlled by the microcontroller through I2C/SPI buses. The chip's backscatter data input, wake-up signal, and synchronization signals are connected to the microcontroller(MCU) as well. The MCU turns on the sync receiver circuits once the wake-up pattern is detected, and it starts sending the data to the chip after an appropriate delay when the synchronization signal is asserted. The delay ensures the PHY header of the frame is preserved and stays valid. In a practical design, all the regulators and the microcontroller can be integrated onto the same chip, which would greatly reduce size and power consumption.

5 Evaluation

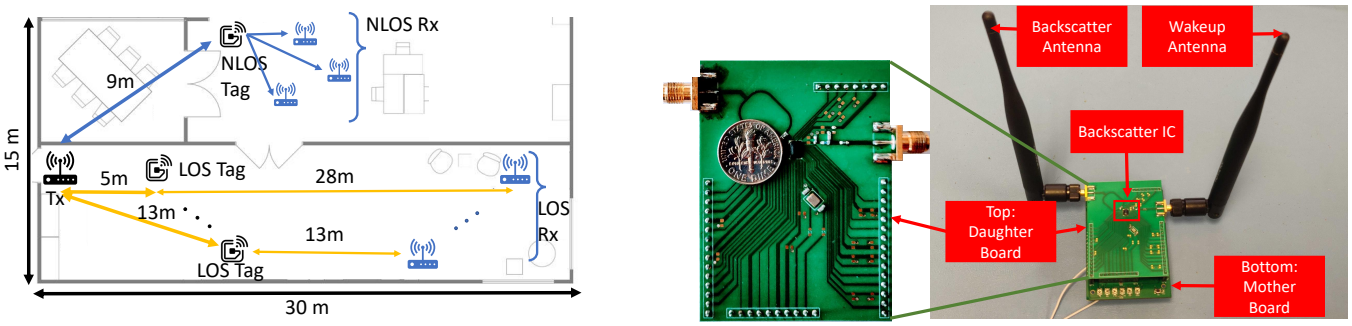
In this section, we first present micro-benchmarks to demonstrate the working of the individual modules in the SyncScatter tag. Then we evaluate the end-to-end performance of the SyncScatter in terms of Bit Error Rate (BER) and goodput by placing the tag in a large indoor environment (30m x 15m) as shown in Fig. 9a. We conduct the experiments by placing the Transmitter AP, Tag, and the receiver AP in both line of sight and non-line of sight conditions to demonstrate that SyncScatter tag is suitable for deployment in an office environment. We also conduct experiments to find SyncScatter's maximum range by co-locating the transmitter and receiver AP. To understand the impact of the co-existence of multiple tags, we perform goodput measurements by keeping two multiple tags in the same environment.

5.1 Micro-benchmarks

Here we verify the tag's functionality with micro-benchmarks to explain wake-up, synchronization, backscattering modules, and overall system design.

Tag Wake-up Accuracy: To test the robustness of the wake-up receiver, we measure the accuracy of wake-up with varying input power levels. To evaluate this, we conduct a wireless experiment to send a wake-up packet at different power levels. If the tag has woken up to the T0 and T1 packet, then it generates a trigger signal at the end of the wake-up packet. We send 1000 wake-up packets at varying power levels at the tag from -38 dBm to -30 dBm and monitor the number of triggers generated by the tag. The wake-up error rate is calculated as the number of successful triggers divided by the number of wake-up packets sent. Fig. 10a shows the wake-up accuracy for different power levels. SyncScatter's sensitivity is approximately -34 dBm, and for power levels above -34 dBm, the wake-up rate is very close to 1. Hence the tag responds well up to power levels of -34dBm, and SyncScatter will thus be able to wake up at a distance of up to 30 m.

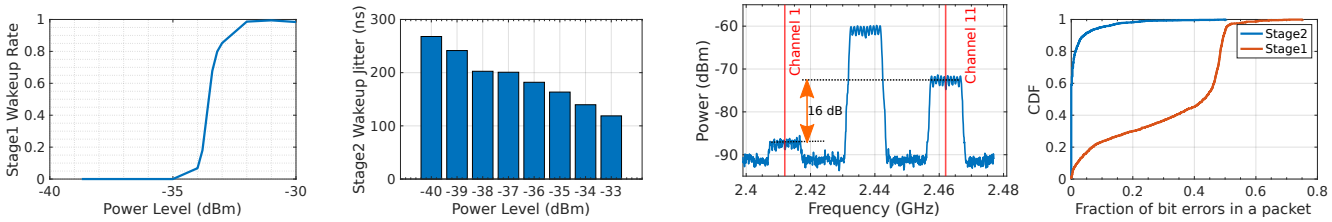
Synchronization Jitter: The synchronization stage is prone to have some timing jitter in detecting the variation of power level. Here we quantify the jitter at different input power levels by doing a wireless experiment similar to the previous setup. To measure the jitter, we send an 802.11b packet from an RF signal generator and monitor the output of the synchronization stage. Then, we measure the time elapsed from



(a) Experiment floor plan

(b) SyncScatter tag

Figure 9: (a) The floor plan shows our deployment of the transmitter, the backscatter tag and the receiver AP at eight different locations in an office environment. **(b)** The mother board and the daughter board with the wire-bonded backscatter IC.



(a) **(b)** **(c)** **(d)**

Figure 10: (a) Wakeup rate **(b)** Sync Jitter **(c)** SSB Backscattering **(d)** CDF of Bit error rates.

the moment the RF signal is applied and the instant at which the rising edge appears at the synchronization stage's output. We repeat this experiment by sending a thousand packets at different power levels and report the standard deviation of all the measurements at each power level. Figure 10b shows the synchronization jitter is below 150ns at -35 dBm power level and beyond, achieving desired optimal spec.

SSB Backscatter: Here we evaluate the performance of the single sideband backscatter in terms of the image rejection of SyncScatter tag. To measure the image rejection, we configure the tag to shift by 25MHz to the upper sideband and send a Wi-Fi signal on channel 6. Figure 10c shows the backscattered signal on channel 1 and channel 11. We observe that the left sideband signal i.e on channel 1, is ∼16 dB lower than the channel 11 signal.

5.2 Chip power consumption

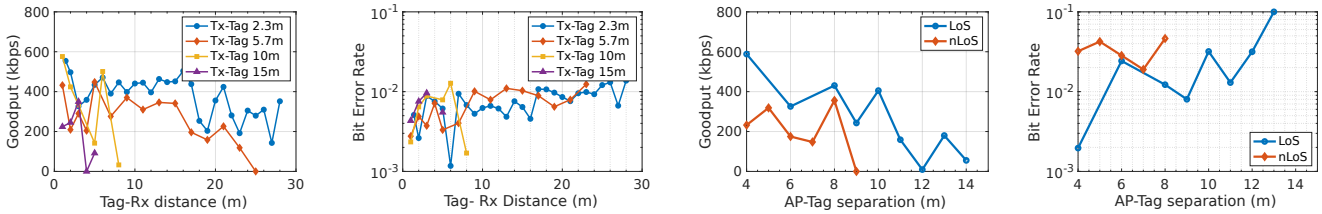
Here we evaluate the power consumption of the wake-up receiver, sync receiver, and backscatter stages of the chip. We report each stage's power consumption as the product of the supply voltage and the current drawn from the supply. The chip is powered from a 0.5v supply, and we use a source measurement unit [2] that can measure the current drawn by the circuit with a precision of 1μA. We observe that the on-board 16MHz oscillator and the wake-up receiver draw a current of 5.6 μA consuming 2.8μW. Since the chip spends most of its time in the wake-up state looking for the right time to backscatter, its average power consumption is dominated by the stage one wake-up receiver. Once the wake-up receiver detects T0 and T1 packets, then sync receiver is turned on, which draws 480 μA current and thus consumes 240μW, but only for an average of 50μs, to account for 40kHz sampling rate error. For a nominal wake-up duration of 500μs and data packet duration of 2ms, the duty-cycled power of the sync receiver is $\frac{50}{2500} \times 240\mu W = 4.8\ \mu W$. Thus, the sync receiver is controlled by a power switch and duty-cycled to conserve power. When the circuit is actually backscattering, the chip draws 56μA current consuming 28μW of power. The power consumption of the backscatter stage is dominated by the PLL to generate the 50MHz clock. Since the backscatter stage is powered on only during the data packet duration, the duty-cycled power turns out to be $\frac{2000}{2500} \times 28\mu W = 22.4\mu W$. So, the total average power from the three stages throughout the communication duration is 2.8 + 4.8 + 22.4 = 30μW, which is in the same range of Hitchhike [40] tag's power consumption.

5.3 End-to-End Results

Here we use WiFi radios as both the transmitter and the receiver. The transmitter transmits 802.11b signals at a peak power of +30dBm on WiFi channel 1, and the tag is configured to backscatter to channel 11. The receiver AP is set to monitor channel 11 using Wireshark [4], a packet capture software that allows us to collect the backscatter packets and compute bit error rate (BER) and the goodput.

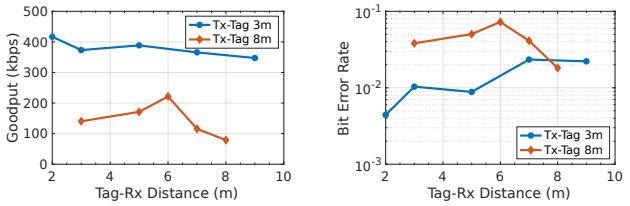
5.3.1 Impact of synchronization stage on BER

To understand the impact of synchronization on BER, we keep the transmitter, SyncScatter tag, and the receiver AP at 20m away from each other and capture the backscattered packets on channel 11. We perform this experiment in two ways, both with and without using the synchronization stage.



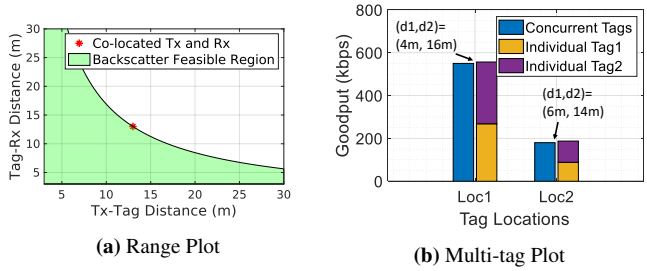
(a) Tx-Rx separated Goodput **(b)** Tx-Rx separated BER **(c)** Tx-Rx co-located Goodput **(d)** Tx-Rx co-located BER

Figure 11: Goodput and BER performance with various Tx-tag and tag-Rx distances for separated case and co-located case.



(a) NLOS Goodput **(b)** NLOS BER

Figure 12: Goodput and BER performance with various Tag-Rx distances for the NLOS case.



(a) Range Plot **(b)** Multi-tag Plot

Figure 13: Backscatter range plots.

We transmit 1000 data packets and plot the CDF of the bit error rates of the captured packets. As shown in Figure 10d, when the synchronization receiver is not used, more than 80 percent of the packets have BER higher than 0.1. But when the synchronization receiver is used to find the packet start, 70 percent of the packets have BER less than 10^{-3}, which is a substantial improvement over the other case.

5.3.2 Goodput/BER for a Single Tag

Here we compute the Goodput and the bit error rates by placing the transmitter, tag, and the AP in different Line of Sight(LoS) as well as non-line of sight(nLoS) configurations. We calculate goodput by taking into consideration the MAC and PHY layer overheads which account for approximately 40 percent of the packet data.

Line of Sight: We perform LoS experiments by placing the Tx, tag, and the Rx in a long corridor. We vary both the Tx to Tag distance and Tag to Rx distance while measuring the performance of SyncScatter tag. Figure 11a shows that when Tx to Tag distance is 2.3m, SyncScatter tag gives an average goodput of 450kbps at up to 30m from the tag. As expected, when we increase the transmitter to tag separation, we observe that the range of backscatter tag decreases due to higher path loss. For example, when tag and transmitter are separated by 10m, SyncScatter tag provides 400kbps goodput only until 9m tag to receiver separation. As shown in Figure 11b, the BER remains to be 10^{-2} over the whole range. We note that although the BER remains low over a wide range of Tx to tag and tag to Rx separations, there is a drop in the goodput values at certain locations. This could be attributed to the multipath nature of wireless channel causing deep fades at certain locations, impacting the number of successful packets decoded which in turn influences the goodput.

Non Line of Sight: In a real-world deployment, true line of sight conditions do not necessarily occur. So, we test our tag setup in a non-line of sight(nLoS) condition where the transmitter and tag are blocked by a wall between them. We vary the transmitter to tag separation and compute the bit error rate and goodput for various tag to receiver separations as plotted in Figure12a and Figure12b. Measurements show that the SyncScatter tag is able to be wake-up successfully even when blocked by a wall and has a backscatter range of 9m. Blockage due to the wall attenuates the signal, resulting in the throughput and BER being worse when compared to the line of sight case. When the Tx to tag separation is 3m, the tag offers an average goodput of 400kbps, and it falls to 100kbps when the Tx to tag separation is increased to 8m.

5.3.3 Co-located Tx and Rx set up

We also evaluate SyncScatter by placing the Transmitter and Receiver at the same location in both LoS and nLoS conditions. In LoS case, the backscatter range is limited to 13m which is smaller than the 30m range observed in the previous experiments where the Tx and Rx are at different separations from the tag. When the Tx and Rx are co-located, the backscatter signal suffers from the path loss twice due to the back and forth travel from the AP to the tag and back to the same location, impacting the maximum range. This argument also supports our observation that the rate of decrease in the goodput with separation is high as shown in Fig.11d and Fig. 11c. In the nLoS scenario, the tag offers an average throughput of 200kbps, with the maximum backscatter range reducing to 8m because of the wall blockage between the Tx and the tag.

5.3.4 Range of the SyncScatter Tag

Now we describe the maximum range until which SyncScatter tag works. In the co-located experimental setup, we observed that the backscatter packet decoding is successful up to 13m AP to tag Line of sight separation. Since the backscatter communication range is determined by the product of

Tx-tag (d_1), Tag-Rx separations(d_2). Any tag location satisfying the relation $d_1 \times d_2 \leq 13 \times 13 = 169m^2$ and $d_1 \leq$ maximum wake-up distance should be within the backscatter range. Fig. 13a shows the feasible backscatter range, which goes up to 30m in either direction.

5.3.5 Co-existence of Multiple Tags

Finally, we find out if multiple SyncScatter tags can exist together, specifically we want to know if the two tags wake-up simultaneously and cause interference to each other. In this experiment, we place the Tx and Receiver separated by 20m. The two tags are placed in line of sight(LoS) with the Tx and Rx and approximately at the same location. Fig. 13b shows the network goodput when both the tags are present and compare it against goodput when only one tag is present. We note that the concurrent goodput is slightly less than the averaged goodput of individual tags.

6 Related Work

SyncScatter is related to prior backscatter based networking with ambient signals [13, 22] that provides low-cost and energy-efficient communication [24, 31]. However, none of the past literature has shown the ability to synchronize at extremely low power with incoming WiFi signals. Furthermore, SyncScatter for the first time analyses the fundamental range limitation for WiFi backscattering, and provides a necessary synchronization specification to maximize the range. Using the analysis and a custom IC hardware design, SyncScatter improves the range, data-rate, and scales to multiple tags over existing literature, even prior IC implementations [35]. SyncScatter is related to the following topics:

Tone based WiFi backscatter communication: SyncScatter synchronizes with ambient commodity WiFi signals, therefore do not require deployment of new tone generators. In contrast, traditional backscatter system like RFID [12, 15, 16, 27, 29, 33, 39, 41] require reader device which act as both tone-generator and receiver. Recent work has shown the ability to backscatter WiFi from an incoming tone signal [19–21, 28, 32], which requires an excitation radio that can generate sine tone. For example, [19] uses a Bluetooth radio to generate a tone, but with the standards-limiting the maximum transmit power, the backscatter range is limited to smaller distances. In contrast, since SyncScatter leverages existing WiFi infrastructure for both excitation and receiving radio, it simplifies the deployment while improving range.

Backscatter communication with Ambient signals: Recently, there have been multiple works on backscattering ambient signals, wherein the excitation radio and receiving radio use existing infrastructure like WiFi [21, 40]. In general, the infrastructure-based backscatter that leverages WiFi [11, 20, 23, 38], LTE [13], Bluetooth [14, 19], ZigBee [22, 43], LoRa [17, 26, 28] or even visible light signal [37] can all benefit from SyncScatter's hierarchical protocol to improve the transmitter-to-tag range and throughput. For example, recent work which backscatters LTE signals [13] tag uses a preamble while backscattering due to the lack of synchronization with incident signals. In such scenarios, the hierarchical wake-up receiver can enable synchronization and improve the tag's performance.

Prior WiFi-based backscatter works focus on the core-principle of codeword translation: changing one OFDM symbol to another valid OFDM symbol [42] and [6] extends backscatter communication to leverage features of the MAC layer. Multi-hop backscatter [45] builds a mesh network of tags, and a few other works [9, 23, 44, 46] propose to leverage spatial multiplexing on the backscatter tag. However, none of the existing works provide extended coverage due to their fundamental inability to synchronize with the ambient signals.

7 Discussion and Future Work

SyncScatter presents a first integrated circuit to achieve synchronized backscatter communication with ambient signals.

802.11b and Beyond: Although 802.11b is an old technology, most modern APs come with dual-band radios supporting both 2.4GHz and 5 GHz radios. The recent 802.11ax standard is designed to be backward compatible with 11b/g devices and hence backscattering on 802.11b signals is still very applicable. Moreover, all the symbol-based backscatter systems such as FreeRider [42] which use 802.11g signals, requires synchronization making our design suitable for modern WiFi standards as well.

Adapting WakeUp Receivers to different protocols: Our hierarchical wake-up receiver design can be extended to backscatter systems based on BLE, OFDM, and LoRA protocols with some minor modifications. We observe that BLE and LoRA signals have a constant signal envelope similar to 11b signals, and they would work with our design. However, since LoRA networks demand long-range, the wake-up receiver has to be designed with more sensitivity at the expense of more power consumption on the tag. In the case of OFDM signals, the wake-up packets have to be engineered to have a small peak to average power ratio (PAPR) so that they appear like a constant envelope signal to the envelope detectors.

Integrated Power Management and Energy Harvesting: In the current implementation, we design the integrated circuit without integrated power management solutions, leading to higher power consumption. In future work, we would integrate the power management circuits such as voltage regulators, power switches within the integrated circuit itself. We note that our chip can be powered using a rechargeable coin cell such as CR2032 [1] and can have a continuous operation for more than 3 years. We can further enhance its life by exploring RF energy harvesting [7, 30] techniques to replenish the battery and integrate them onto the fabricated chip.

8 Acknowledgements

We thank anonymous reviewers and our shepherd, Prof. Haitham Hassanieh, and members of WCSNG, UCSD for their insightful feedback. This work was supported in part by the National Science Foundation under Grant No. 1923902.

References

[1] Cr2032 coin cell battery. https://data.energizer.com/pdfs/cr2032.pdf.

[2] Keithley source measurement unit. http://www.farnell.com/datasheets/2238744.pdf?_ga=2.15412083.450713776.1499842582-1530823304.1499842582.

[3] Vocal technologies: 802.11b white paper. https://www.vocal.com/wp-content/uploads/2012/05/802.11b_wp1pdf.pdf.

[4] Wireshark - packet captture and analysis tool. https://www.wireshark.org/.

[5] Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan networks - specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Higher speed physical layer (phy) extension in the 2.4 ghz band. *IEEE Std 802.11b-1999*, pages 1–96, 2000.

[6] A. Abedi, M. H. Mazaheri, O. Abari, and T. Brecht. Witag: Rethinking backscatter communication for wifi networks. In *Proceedings of the 17th ACM Workshop on Hot Topics in Networks*, pages 148–154, 2018.

[7] F. Alneyadi, M. Alkaabi, S. Alketbi, S. Hajraf, and R. Ramzan. 2.4 ghz wlan rf energy harvester for passive indoor sensor nodes. In *2014 IEEE International Conference on Semiconductor Electronics (ICSE2014)*, pages 471–474. IEEE, 2014.

[8] R. Barnett, G. Balachandran, S. Lazar, B. Kramer, G. Konnail, S. Rajasekhar, and V. Drobny. A passive uhf rfid transponder for epc gen 2 with-14dbm sensitivity in 0.13 μm cmos. In *2007 IEEE international solid-state circuits conference. digest of technical papers*, pages 582–623. IEEE, 2007.

[9] D. Bharadia, K. R. Joshi, and S. Katti. Full duplex backscatter. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, pages 1–7, 2013.

[10] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti. Backfi: High throughput wifi backscatter. *ACM SIGCOMM Computer Communication Review*, 45(4):283–296, 2015.

[11] D. Bharadia, K. R. Joshi, M. Kotaru, and S. Katti. Backfi: High throughput wifi backscatter. *ACM SIGCOMM Computer Communication Review*, 45(4):283–296, 2015.

[12] M. Buettner, B. Greenstein, and D. Wetherall. Dewdrop: an energy-aware runtime for computational rfid. In *Proc. USENIX NSDI*, pages 197–210, 2011.

[13] Z. Chi, X. Liu, W. Wang, Y. Yao, and T. Zhu. Leveraging ambient lte traffic for ubiquitous passive communication. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 172–185, 2020.

[14] 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 *2015 IEEE international conference on RFID (RFID)*, pages 78–85. IEEE, 2015.

[15] J. Gummeson, S. S. Clark, K. Fu, and D. Ganesan. On the limits of effective hybrid micro-energy harvesting on mobile crfid sensors. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 195–208, 2010.

[16] 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, 2012.

[17] M. Hessar, A. Najafi, and S. Gollakota. Netscatter: Enabling large-scale backscatter networks. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 271–284, 2019.

[18] X. Huang, G. Dolmans, H. d. Groot, and J. R. Long. Noise and sensitivity in rf envelope detection receivers. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(10):637–641, 2013.

[19] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *SIGCOMM*, 2016.

[20] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: Internet connectivity for rf-powered devices. In *ACM SIGCOMM Computer Communication Review*, 2014.

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

[22] Y. Li, Z. Chi, X. Liu, and T. Zhu. Passive-zigbee: enabling zigbee communication in iot networks with 1000x+ less power consumption. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, pages 159–171, 2018.

[23] X. Liu, Z. Chi, W. Wang, Y. Yao, and T. Zhu. Vmscatter: A versatile {MIMO} backscatter. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 895–909, 2020.

[24] S. Naderiparizi, M. Hessar, V. Talla, S. Gollakota, and J. R. Smith. Towards battery-free {HD} video streaming. In *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, pages 233–247, 2018.

[25] Openwrt. Openwrt project. https://openwrt.org/.

[26] Y. Peng, L. Shangguan, Y. Hu, Y. Qian, X. Lin, X. Chen, D. Fang, and K. Jamieson. Plora: A passive long-range data network from ambient lora transmissions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 147–160, 2018.

[27] A. P. Sample, D. J. Yeager, P. S. Powledge, and J. R. Smith. Design of a passively-powered, programmable sensing platform for uhf rfid systems. In *2007 IEEE international Conference on RFID*, pages 149–156. IEEE, 2007.

[28] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota. Lora backscatter: Enabling the vision of ubiquitous connectivity. *IMWUT*, 2017.

[29] S. J. Thomas and M. S. Reynolds. A 96 mbit/sec, 15.5 pj/bit 16-qam modulator for uhf backscatter communication. In *2012 IEEE International Conference on RFID (RFID)*, pages 185–190. IEEE, 2012.

[30] V. H. Tran, A. Misra, J. Xiong, and N. Hirunima. Can wifi beamforming support an energy-harvesting wearable? In *Proceedings of the Fifth ACM International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems*, ENSsys'17, page 14–20, New York, NY, USA, 2017. Association for Computing Machinery.

[31] D. Vasisht, G. Zhang, O. Abari, H.-M. Lu, J. Flanz, and D. Katabi. In-body backscatter communication and localization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 132–146, 2018.

[32] A. Wang, V. Iyer, V. Talla, J. R. Smith, and S. Gollakota. Fm backscatter: Enabling connected cities and smart fabrics. In *NSDI*, pages 243–258, 2017.

[33] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. *ACM SIGCOMM Computer Communication Review*, 42(4):61–72, 2012.

[34] P.-H. P. Wang, H. Jiang, L. Gao, P. Sen, Y.-H. Kim, G. M. Rebeiz, P. P. Mercier, and D. A. Hall. A 6.1-nw wake-up receiver achieving -80.5 dbm sensitivity via a passive pseudo-balun envelope detector. *IEEE Solid-State Circuits Letters*, 1(5):134–137, 2018.

[35] P.-H. P. Wang, C. Zhang, H. Yang, D. Bharadia, and P. P. Mercier. 20.1 a 28μw iot tag that can communicate with commodity wifi transceivers via a single-side-band qpsk backscatter communication technique. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 312–314. IEEE, 2020.

[36] P.-H. P. Wang, C. Zhang, H. Yang, M. Dunna, D. Bharadia, and P. P. Mercier. A low-power backscatter modulation system communicating across tens of meters with standards-compliant wi-fi transceivers. *IEEE Journal of Solid-State Circuits*, 55(11):2959–2969, 2020.

[37] X. Xu, Y. Shen, J. Yang, C. Xu, G. Shen, G. Chen, and Y. Ni. Passivevlc: Enabling practical visible light backscatter communication for battery-free iot applications. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 180–192, 2017.

[38] Z. Yang, Q. Huang, and Q. Zhang. Nicscatter: Backscatter as a covert channel in mobile devices. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, pages 356–367, 2017.

[39] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational rfid and sensing platform. *University of Massachusetts Computer Science Technical Report UM-CS-2011-020*, 2011.

[40] P. Zhang, D. Bharadia, K. Joshi, and S. Katti. Hitchhike: Practical backscatter using commodity wifi. In *SenSys*, 2016.

[41] 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, 2014.

[42] P. Zhang, C. Josephson, D. Bharadia, and S. Katti. Freerider: Backscatter communication using commodity radios. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 389–401, 2017.

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

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

[45] J. Zhao, W. Gong, and J. Liu. X-tandem: Towards multi-hop backscatter communication with commodity wifi. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 497–511, 2018.

[46] R. Zhao, F. Zhu, Y. Feng, S. Peng, X. Tian, H. Yu, and X. Wang. Ofdma-enabled wi-fi backscatter. In *The 25th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2019.

9 Appendix

9.1 BLE synchronization requirement

To find the synchronization requirement for BLE(Bluetooth low energy) backscatter communication, we perform Matlab simulations for different synchronization delays. BLE signals encode the data bits using two different sine tones located 250kHz on either side of the center frequency, where each symbol in the BLE packet occupies a duration of $1\mu s$. To encode tag data on a BLE packet, the frequency of the sine tone present in a BLE symbol is modified by the backscatter tag. In our simulation, we consider incident data packets that are modulated using FSK(frequency shift keying) technique. The tag's modulation signal is also generated from the tag bits using the FSK technique. To incorporate the synchronization delay while backscattering, we delay the tag modulation by a fixed duration (given by the synchronization delay) beginning from the packet start instant. The resulting backscattered signal is also an FSK signal and is decoded by an FSK receiver. Fig 14 plots the BER vs SNR curve for different values of synchronization delay. As can be seen, the BLE backscatter loses 4 dB of SNR at 10^{-3} BER due to a synchronization delay of 100 ns. For 150ns synchronization delay, it loses more than 8 dB SNR. To not degrade the SNR, BLE backscatter requires a synchronization accuracy better than 100 ns which is equal to $\frac{1}{10}$ th of the symbol duration.

9.2 OFDM Synchronization requirement

To simulate OFDM backscatter synchronization requirements, here we consider the standard 20MHz OFDM signals. A WiFi packet contains many OFDM symbols, with each symbol containing 64 sub-carriers. Each OFDM symbol occupies 3.2 us duration and is preceded by a cyclic prefix of 0.8us duration. To encode data onto a WiFi packet, the backscatter tag changes the phase of an OFDM symbol. For instance, to convey tag data 1, the tag induces an additional phase of π radians on all the sub-carriers. Similarly, to encode bit 0, the phase of

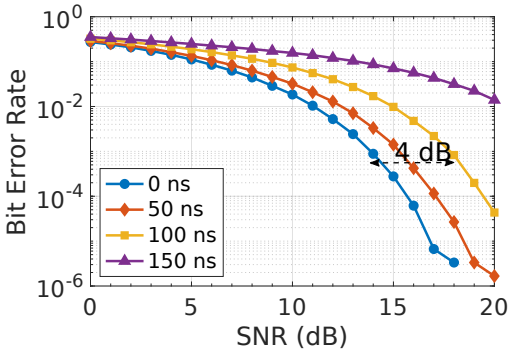


Figure 14: BER vs SNR curve for different synchronization delays in BLE backscatter

the sub-carriers is left unchanged. In this way, a sequence of tag bits is encoded on a WiFi packet by modifying the phase of each OFDM symbol in the packet. At the receiver, these phases are extracted to decode the tag data. If the backscatter tag is not aware of the OFDM symbol boundary, the tag bits will not be appropriately encoded on the WiFi packet, and the WiFi packet suffers from loss of SNR. The BER vs SNR curves for different synchronization delays are plotted in Fig 15. We note that the OFDM backscatter is able to tolerate a large synchronization delay of up to 1000ns. The primary reason the tolerance is very large is that every OFDM symbol has a cyclic prefix of length 800ns discarded while decoding the packet at the receiver. So, if the signal that is part of a cyclic prefix is corrupted, it will not impact the bit error rate. Another reason is that the same tag bit is encoded on every sub-carrier of the OFDM symbol, implying if the majority of the sub-carriers are decoded correctly, the BER would not be impacted much. From Fig 15, we notice that the BER floors, although the SNR increases and it loses more than 10 dB SNR to achieve 10^{-3} BER. This suggests synchronization delay plays a major role in reducing the bit error rate.

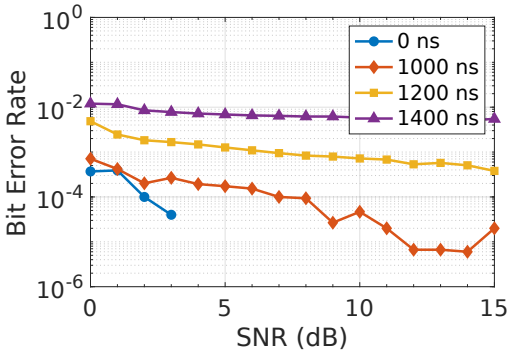


Figure 15: BER vs SNR curve for different synchronization delays in OFDM backscatter