WiTAG: Seamless WiFi Backscatter Communication

Ali Abedi
Cheriton School of Computer Science
University of Waterloo
ali.abedi@uwaterloo.ca

Farzan Dehbashi
Cheriton School of Computer Science
University of Waterloo
farzan.dehbashi@uwaterloo.ca

Mohammad Hossein Mazaheri
Cheriton School of Computer Science
University of Waterloo
mh2mazah@uwaterloo.ca

Omid Abari
Computer Science Department
UCLA
omid@cs.ucla.edu

Tim Brecht
Cheriton School of Computer Science
University of Waterloo
brecht@cs.uwaterloo.ca

ABSTRACT

WiFi backscatter communication has the potential to enable battery-free sensors which can transmit data using a WiFi network. In order for WiFi backscatter systems to be practical they should be compatible with existing WiFi networks without any hardware or software modifications. Moreover, they should work with networks that use encryption. In this paper, we present WiTAG which achieves these requirements, making the implementation and deployment of WiFi backscatter communication more practical. In contrast with existing systems which utilize the physical layer for backscatter communication, we take a different approach by leveraging features of the MAC layer to communicate. WiTAG is designed to send data by selectively interfering with subframes (MPDUs) in an aggregated frame (A-MPDU). This enables standard compliant communication using modern, open or encrypted 802.11n and 802.11ac networks without requiring hardware or software modifications to any devices. We implement WiTAG using off-the-shelf components and evaluate its performance in line-of-sight and non-line-of-sight scenarios. We show that WiTAG achieves a throughput of up to 4 Kbps without impacting other devices in the network.

CCS CONCEPTS

• **Networks** → **Network architectures**; *Wireless access points, base stations and infrastructure*; • **Hardware** → **Wireless devices**; **Wireless integrated network sensors**.

KEYWORDS

Battery-free communication; WiFi Backscatter; Internet of Things (IoT); 802.11 Networks; Sensors

ACM Reference Format:
Ali Abedi, Farzan Dehbashi, Mohammad Hossein Mazaheri, Omid Abari, and Tim Brecht. 2020. WiTAG: Seamless WiFi Backscatter Communication. In *Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20), August 10–14, 2020, Virtual Event, NY, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3387514.3405866

SIGCOMM '20, August 10–14, 2020, Virtual Event, NY, USA

ACM ISBN 978-1-4503-7955-7/20/08…$15.00
https://doi.org/10.1145/3387514.3405866

1 INTRODUCTION

Backscatter systems are very attractive as a means of communication for wireless sensors in applications ranging from implantable body sensors to farming [21, 31, 32, 38]. Because they do not require batteries they can have a lower cost, smaller form factor, and require less maintenance. Traditional backscatter systems (such as RFIDs) require a specialized reader to read the tag values. The high cost and large form factor of these readers have made them difficult to deploy and have limited the adoption of RFID tags in many applications. To overcome these challenges, researchers have designed WiFi backscatter systems. The vision is to design backscatter tags that can be read using WiFi devices, thus reducing the complexity and cost of deploying backscatter systems by using existing WiFi infrastructures instead of specialized readers.

Wi-Fi backscatter [14] is the first WiFi backscatter system that enables communication with commodity WiFi devices. Unfortunately, due to self-interference between WiFi transmission and backscatter signals the range of this system is very limited [38]. BackFi [4] and Passive WiFi [15] try to increase the range of communication, however they require specialized hardware which hinders the widespread deployment of these systems. Ideally WiFi backscatter systems need to satisfy the following key requirements:

- **Compatible with existing WiFi access points:** They should be compatible with already deployed commodity access points (APs), including 802.11n and 802.11ac standards, without requiring hardware or software changes.
- **Work with encrypted WiFi networks:** Most WiFi networks are secured using encryption. Therefore, WiFi backscatter systems should work with WiFi networks that use WPA or WEP encryption.
- **Battery-free:** Similar to traditional backscatter tags (RFIDs), WiFi backscatter tags need to be extremely low-power so that they can harvest their energy from the environment and operate without requiring a battery.

If the above requirements are satisfied, we can envision having battery-free wireless sensors which do not require specialized readers and can be deployed in environments with existing WiFi infrastructure. However, to the best of our knowledge, no current backscatter system satisfies all of these requirements. Recently, a group of systems namely, FS-Backscatter [38], HitchHike[36],

FreeRider [37], and MOXcatter [39], attempt to enable WiFi backscatter using only commodity WiFi devices. All of these systems reflect the signals onto an adjacent channel to avoid self interference. Therefore, they require two access points to work. Moreover, they require software modifications on the second access point to receive the backscattered packets. Most importantly, because HitchHike, FreeRider, and MOXcatter modify physical-layer symbols, they do not work if the network uses encryption.

In this paper, we present WiTAG, a novel WiFi backscatter system that communicates data by leveraging MAC-layer features. WiTAG works with encrypted WiFi networks and does not require any software or hardware modification to WiFi access points. Moreover, WiTAG does not require installing a second access point. WiTAG introduces two key innovations:

1) MAC-Layer Backscatter Communication: WiTAG introduces a new backscatter communication mechanism which utilizes MAC-layer "frame aggregation" available in 802.11n and 802.11ac standards. These standards place multiple MAC-layer data units (subframes) in a large PHY-layer packet (aggregated frame) to improve throughput, as shown in Figure 1. In WiTAG, a WiFi device sends an aggregated frame to an AP in the network using a physical rate that is likely to be successfully received. This frame acts as a query for the tag (e.g., IoT sensor). The tag embeds its data into the aggregated frame by selectively corrupting some subframes. Then, the AP sends a block ACK back to the transmitting WiFi device, indicating which subframes have or have not been successfully received. The WiFi device extracts the data being communicated by the tag from the bits in the block ACK. Subframes that are not corrupted by the tag are received successfully (represented by a 1 in the block ACK) and those that are corrupted are not received successfully (represented by a 0 in the block ACK). Note that the AP is completely oblivious to the existence of the tag and does not require any modification. Further, because the tag communicates its data by selectively corrupting subframes in query frames, it does not reflect signals onto a secondary channel and does not require a second access point. Most importantly, because tags communicate by corrupting encrypted or unencrypted MAC-layer subframes WiTAG works with networks that use encryption. Because the sole purpose of a query packet is to provide the tag with an opportunity to communicate, the actual contents of the query packet are irrelevant and are discarded by the system whether or not they are corrupted by the tag.

2) Passive Subframe Corrupting: WiTAG's second innovation is a technique to passively corrupt subframes. Note that an active radio which has a transmitter can easily corrupt a subframe by transmitting an interfering signal. However, a backscatter tag is a passive device and does not have a transmitter. To solve this problem, we design a backscatter tag which can modify the wireless channel during the transmission of a subframe, hence corrupting that subframe. Because frame aggregation performs channel estimation only once at the beginning of the aggregated frame, modifying the channel during the transmission of a subframe corrupts the subframe, therefore, it cannot be decoded.

We have built a prototype of a WiTAG tag using off-the-shelf components. We evaluate WiTAG using commodity WiFi devices without hardware or software modifications. We run experiments in both line-of-sight and non-line-of-sight scenarios. Our results show that WiTAG is able to send a tag's data to the querying device at 4 Kbps with high success rates, even when used in an office environment with other networks and devices operating in the same 2.4 GHz spectrum and while competing with other traffic on the same network.

Our contributions are:[1]

- The first battery-free backscatter communication system which is fully compatible with existing open or encrypted 802.11n and ac WiFi networks without requiring software or hardware modifications.
- We develop a technique that enables WiTAG tags to selectively corrupt subframes in an aggregated frame without requiring an active transmitter.
- To support multiple tags that communicate periodically, we design a new technique for synchronizing query packets with the time during which a tag is backscattering.
- We have implemented WiTAG using off-the-shelf components and commodity WiFi devices. Our empirical results show that WiTAG works in both line-of-sight and non-line-of-sight scenarios.

2 BACKGROUND

WiTAG takes advantage of MAC-layer frame aggregation (the combining of several subframes to create larger frames) to communicate data by altering the wireless channel. Therefore, we now provide background on frame aggregation and PHY layer channel estimation and correction.

2.1 802.11 Frame Aggregation

The IEEE 802.11n and ac standards provide a frame aggregation mechanism to improve the efficiency of the MAC layer [6, 7]. In order to avoid overheads such as performing channel sensing and transmitting an acknowledgment per frame, multiple *MAC Protocol DATA Units* (MPDUs) are combined into a larger aggregated frame (A-MPDU), as illustrated in Figure 1. By aggregating multiple subframes, the overhead is amortized over more useful bits and therefore the efficiency of the MAC layer improves significantly. The receiver of an A-MPDU transmits a *block ACK* back to the sender to report the fate of the individual subframes inside the A-MPDU. A block ACK is similar to the legacy 802.11 acknowledgment, however rather than acknowledging the successful reception of one frame, it reports the fate of each MPDU using a 64-bit bitmap. WiTAG leverages this frame aggregation scheme and block ACKs to allow tags to transmit data to a WiFi device.

Figure 1: 802.11n/ac A-MPDU structure

2.2 Channel Estimation and compensation

In all 802.11 standards including n and ac, the PHY header starts with multiple known training symbols. The receiver utilizes these

[1]This work does not raise any ethical issues.

symbols to perform channel estimation [6, 7]. The wireless channel determines how a signal changes as it propagates from transmitter to receiver. Since the PHY header includes known symbols for each subcarrier, the receiver can estimate the phase and amplitude per subcarrier. This is known as Channel State Information (CSI). The receiver uses the estimated CSI to remove the effect of the channel from the received signal. In Section 4, we explain how WiTAG alters the wireless channel for a short period of time during the transmission of a subframe, resulting in a subframe that can not be decoded.

3 DESIGN OVERVIEW

WiTAG is a WiFi backscatter system designed for use with existing WiFi networks. It enables a battery-free tag to communicate data to existing WiFi devices without requiring any modifications. WiTAG operates in two steps, as shown in Figure 2.

In the first step (labelled 1), the client device, transmits an A-MPDU (consisting of n subframes) to an AP. During the transmission of each subframe, the tag either does nothing, or it corrupts the subframe. If the tag does nothing, the subframe will be decodable at the AP. If the tag corrupts the subframe, it will not be decodable. Therefore, the tag can encode its data by selectively corrupting some subframes and not others. In particular, to transmit 1, the tag does nothing during the transmission of a subframe so it can be decoded successfully. To transmit 0, the tag corrupts the subframe so the AP cannot decode that subframe. The result will be a sequence of failed or successfully decoded subframes that represent the bits for the tag's data. Note that since the sole purpose of query packets is to get the tag's data, the client does not use their content for communication. In Section 4, we describe how a passive tag can create interference for WiFi subframes in order to corrupt the frame.

In the second step (labelled 2 in Figure 2), the access point transmits a block ACK to the client device to notify it about the fate of the subframes in the last A-MPDU. The client device obtains the tag's data directly from the block ACK. Note that although we use the example of a client device transmitting a query packet, the AP could also initiate this process. More importantly, both the client and AP obtain the tag's data irrespective of which device initiates the query.

In WiTAG communication, there is a chance that some subframes are corrupted due to other reasons such as path loss and interference from other sources. Although, WiTAG tries to avoid this by using a robust PHY-layer transmission rate for the query packet, there is still a small chance that unintentional subframe corruption happens. Therefore, similar to any wireless communication system, WiTAG requires an error detection and/or correction mechanism. Note that bit errors caused by unintentional subframe corruption are considered in our bit error rate measurements in Section 8.

WiTAG does not require any modifications to existing WiFi networks because the access point receives normal A-MPDUs and transmits normal block ACKs. Therefore it is not even aware of backscatter communications. Similarly, the client device transmits and receives normal 802.11 frames and hence requires no modifications to the MAC or PHY layer. It only requires an application that reads the tag's data from block ACKs. Since WiTAG utilizes MAC-layer A-MPDU aggregation, it is oblivious to the complexities of the PHY layer. As a result, it works with any modulation scheme, coding rate, MIMO configuration, guard interval, and channel width. Most importantly, this feature makes WiTAG compatible with new standards and works with WEP and WPA encrypted packets. In addition to working with 802.11n and ac networks, WiTAG will be compatible with the 802.11ax standard because it also supports A-MPDU aggregation [1].

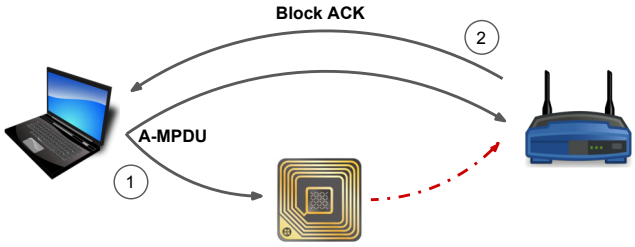


Figure 2: WiTAG overview. WiTAG's tag selectively interferes with subframes in a query packet transmitted by a client to an access point. Then, the client device obtains the tag's data from the block ACK.

The next few sections describe the components that contribute to the design of WiTAG. We start by explaining how WiTAG can passively corrupt a subframe. We then describe how WiTAG synchronizes a tag and the WiFi devices that transmit query packets. Finally we explain how these components work together to enable seamless WiFi backscatter communication.

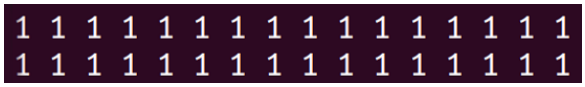
4 CORRUPTING A SUBFRAME

In the previous section, we explained how WiTAG sends its data to a WiFi device by corrupting selected subframes. In this section, we describe how WiTAG corrupts a subframe.

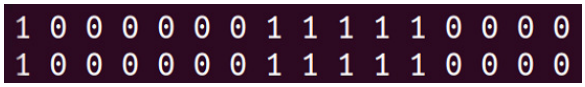
As described in Section 2, an A-MPDU includes a single PHY header followed by n subframes. The header is used to estimate the channel. This estimation is then used to correct the channel for subsequent subframes. Note, in a typical WiFi network, an A-MPDU transmission lasts a few milliseconds and wireless channels do not change during this short time [26, 27]. Therefore, a single channel estimation at the beginning of the A-MPDU is sufficient to successfully correct all sub frames. However, if we modify the wireless channel during the transmission of a subframe, then the channel estimation done at the beginning of the A-MPDU will no longer be valid for that subframe and as a result the subframe will not be received successfully. Therefore, WiTAG selectively corrupts desired subframes by changing the wireless channel during their transmission.

4.1 Changing the Wireless Channel

A wireless channel consists of a direct and multiple indirect paths created by reflectors in an environment. Therefore, if the phase or amplitude of a signal reflected from one of these reflectors changes, the wireless channel will change. A tag in WiTAG uses an antenna which can be switched between two modes: reflective and non-reflective. The antenna is reflective when it is short circuited and



(a) Tag is in sleep mode



(b) Tag changes the channel

Figure 3: Corrupting subframes. Block ACKs of four A-MPDUs (with 16 subframes) when a) the tag is doing nothing, and b) the tag periodically changes the channel between two states.

non-reflective when it is open circuited [32]. Therefore, the tag can quickly change the wireless channel by switching its antenna between these two modes. Because this switching process can be done very quickly, the tag can be non-reflective during channel estimation (i.e., the beginning of an A-MPDU), and then become reflective during the transmission of a subframe. This will corrupt the subframe since the channel estimation is no longer valid for that subframe.

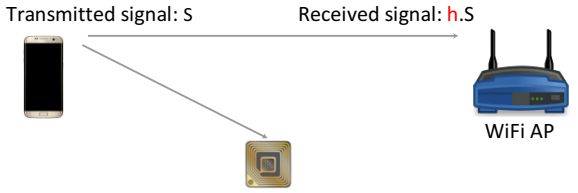
Figure 3 shows the block ACKs of four A-MPDUs (with 16 subframes) when a) the tag is not active, and b) the tag changes the channel state. As depicted in Figure 3a when the tag is in sleep mode, it has no impact on the channel and all subframes can be received successfully. In contrast, Figure 3b shows the case when WiTAG's tag periodically changes the channel. As can be seen, the ACKs are 0 during the time that the tag changes the channel. This implies that WiTAG can successfully corrupt subframes by changing the channel during their transmission.

4.2 Maximizing the Channel Change

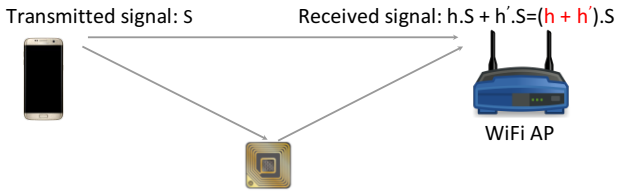
In order to increase the likelihood that a tag corrupts a subframe, we want to maximize the channel change caused by the tag. Wireless channels can be modelled using complex numbers. Figure 4 shows the impact that WiTAG has on the wireless channel, where S is the transmitted signal. In (a), the tag is not reflecting. In this case the received signal is simply $h.S$, where h is the channel coefficient. In (b), the tag is reflecting. In this case the received signal is $(h + h^{'}).S$, where $h^{'}$ is the amount of channel change created by the tag. We want to maximize the channel change because a larger change increases WiTAG's ability to successfully corrupt subframes.

Previous studies [18, 33] propose a technique to change the phase of a reflected signal to improve the performance of WiFi networks. We build on that technique to maximize the amount of channel change for backscatter communication. Instead of switching the tag's antenna between reflecting and non-reflecting modes, we design and implement a tag which is always reflecting, but which can switch the phase of the reflected signal between 0 and 180 degrees.[2] Figure 4 (b) and (c) show the change in the wireless channel when the tag is reflecting with 0 and 180 degrees, respectively. When the tag is reflecting with 0 degrees, the channel changes by

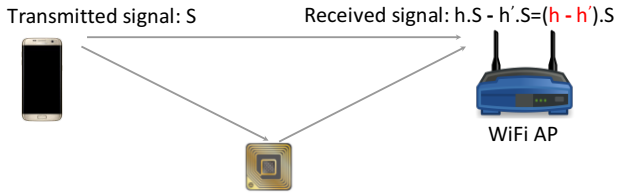
[2]Note, this can be implemented simply by connecting two short circuited cables with different lengths to the output of the switch. The difference between the length of these cables is one quarter of the wavelength and therefore they create a 180 degree phase difference.



(a) Tag is not reflecting



(b) Tag is reflecting with 0 degree phase change



(c) Tag is reflecting with 180 degree phase change

Figure 4: Maximize the change in wireless channel. WiTAG maximizes the change in wireless channel by always reflecting, and switching the phase of the reflected signal between 0° and 180°.

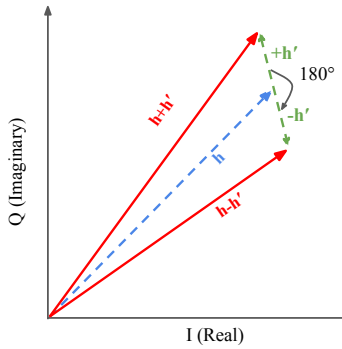
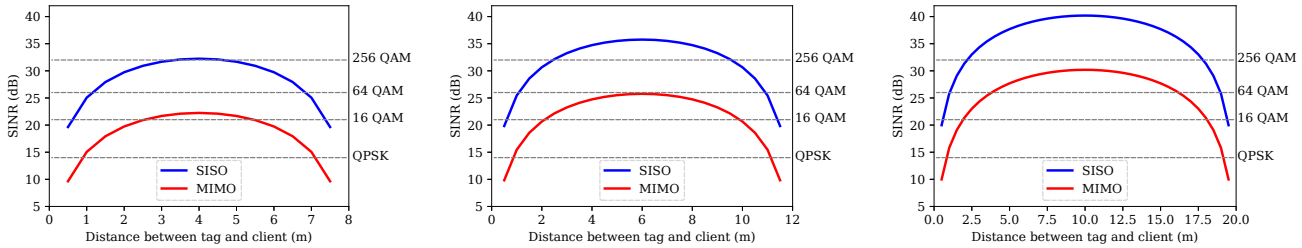


Figure 5: Passive packet corruption in WiTAG

$+h^{'}$ and when the tag is reflecting with 180 degrees, the channel changes by $-h^{'}$. Therefore, the channel difference between these two stages will be always $2h^{'}$, regardless of the value of h (as shown in Figure 5). In particular, the tag reflects with 0 degrees during the channel estimation (done at the beginning of an A-MPDU), and then reflects with 180 degrees during the transmission of a subframe. This increases the impact of WiTAG's tag on the channel by a factor of two, increasing WiTAG's ability to successfully corrupt a subframe since the channel estimation is no longer valid for that subframe. We utilize this technique in our WiTAG prototype, which we empirically evaluate in Section 8.



(a) Client to AP distance: 8 m (b) Client to AP distance: 12 m (c) Client to AP distance: 20 m

Figure 6: Impact of WiTAG's tag on the SINR at the AP side.

4.3 Effectiveness in Corrupting a Subframe

WiTAG selectively corrupts WiFi subframes by changing the wireless channel. Therefore, it is important to understand how effective WiTAG is at changing the channel and if that change is sufficient to corrupt a WiFi subframe. In order to answer this question, we present an analytical characterization of a tag's impact on the SINR (Signal to Interference and Noise Ratio) of WiFi signals at the AP. The SINR is defined as

$$SINR = \frac{P_{Client}}{P_{Tag} + P_{Noise}} \tag{1}$$

where P_{Client} is the power of WiFi signal from the client device, P_{Tag} is the power of backscattered signal, and P_{Noise} is the power of the noise. P_{Client} can be obtained from the free-space propagation model [5], while P_{Tag} can be obtained from the radar reflection model [29]:

$$P_{Client} = \frac{P_C G_C G_A \lambda^2}{(4\pi)^2 D_{CA}}$$
$$P_{TAG} = \frac{P_C G_C G_A \sigma \lambda^2}{(4\pi)^3 D_{TC}^2 D_{TA}^2} \tag{2}$$

where G_C and G_A are the antenna gains of the WiFi client and the AP, respectively. P_C is the transmission power of the WiFi client. D_{TC} and D_{TA} are the distances between the tag and the WiFi client and AP, respectively. D_{CA} is the distance between the WiFi client and the AP and σ is the Radar Cross-Section (RCS) of the tag's antenna. This indicates how much of the incident signal is re-radiated. The RCS of a typical 2.4 GHz dipole antenna with an open circuit load is around 0.015 dBm^2 [10]. Finally, λ is the wavelength of the WiFi 2.4 GHz signal.

Next, we use Equation 1 to plot the SINR for different distances between a tag, the WiFi client and the AP. We assume the antenna gains G_C and G_A are 3 dBi and the transmission power (P_C) is 30 dBm. We also assume the power of the noise (P_{Noise}) for a 20 MHz channel is -97 dBm which is a typical noise floor for WiFi devices. Figure 6 shows the SINR versus the distance between the tag and the client, for different client to AP distances. The figure also shows the minimum SINR required to successfully decode packets transmitted using different modulation schemes [34, 35]. For example, WiFi requires an SINR of at least 21 dB and 26 dB to be able to decode 16 QAM and 64 QAM modulations, respectively. The graphs in Figure 6 show that when a single stream is used (SISO), the SINR of the WiFi signal (i.e., the blue curve) is mostly above the minimum required SINRs for most modulations (i.e., the gray lines). This means that in a single stream system, the only scenarios where a tag is able to degrade the SINR to a point below that required for successful decoding is when it is very close to the AP or the client, or when dense modulation schemes (such as 256 QAM) are used. Note, that in WiFi networks, the 256 QAM modulation scheme works only when the AP is relatively close to the client (i.e., less than a few meters). Hence, these results show that if the client and AP are using only a single stream to communicate, a backscatter tag is not capable of creating enough interference to corrupt a WiFi packet unless the tag is very close to one of the two WiFI devices.

To solve this problem and improve the range of our system, we exploit the *noise amplification* characteristics of MIMO systems. It is known that MIMO systems suffer from the noise amplification problem where they amplify the noise and interference, and hence degrade the effective SINR at the receiver [23]. We now briefly describe how noise is amplified in MIMO systems and how that amplification benefits WiTAG.

A MIMO system can be described using the formula $y = Hx + \Delta y$, where x and y are the transmitted and received signals, respectively. The matrix H represents the channels between pairs of transmitting and receiving antennas, and Δy is the noise and interference. In a typical MIMO system, the receiver receives y and estimates x using the equation $x = H^{-1}(y - \Delta y)$. The effect of Δy on x is amplified by the inverse of the channel matrix. In a MIMO system that uses the zero-forcing[3] MIMO decoding algorithm, prior work [23] has shown that this noise amplification can be formulated as:

$$\frac{||\Delta x||}{||x + \Delta x||} \leq \kappa(H) \frac{||\Delta y||}{||y||} \tag{3}$$

where $||.||$ represents the 2-norm. Δx and Δy are the change in the output and input of the MIMO decoder, respectively, and $\kappa(H)$ is the *condition number* of the matrix H, defined as $\kappa(H) = ||H||\,||H^{-1}||$. Based on Equation 3, prior work has also shown that MIMO systems reduce the upper-bound on the SINR by κ^2. This reductions is typically between 10 and 30 dB, depending on the channel condition and the number of streams (the reduction increases with the number of streams) [2, 23].

WiTAG takes advantage of this characteristic of MIMO systems, making it possible to corrupt subframes at larger distances than in SISO scenarios. Figure 6 plots the effective SINR when MIMO is used (shown using the red curves). To compute the SINR, we use

[3] Zero-forcing is a standard MIMO decoding algorithm that solves the MIMO system equation efficiently [19].

Equation 1 while we consider the impact of MIMO noise amplification. Note that in our calculations, we assume $\kappa^2 = 10\ dB$ which is very conservative.

Figure 6 shows that MIMO systems help the tag to reduce the SINR, and hence it makes it easier to corrupt a subframe. For example, when the client and the AP are 8 metres apart (shown in Figure 6(a)), the tag can always reduce the SINR to a level below that required for 64 QAM decoding. This means, that for this example, a tag can successfully corrupt subframes and hence embed its data in the query packet for any location between the tag and the AP. Figures 6(b) and 6(c) also show results for when the client and the AP are 12 m and 20 m apart, respectively. These graphs show that as the distance between the AP and the client increases, it becomes harder for the tag to corrupt packets. In particular, when the AP and the client are 20 m apart, the distance between the tag and the AP or the tag and the client must be less than 4 meters to be able to corrupt 64 QAM packets. However, as long as the AP and the client are less than 12 m apart (as shown in Figure 6(b)), the tag will be able to corrupt 64 QAM packets at any line-of-sight location between the client and AP. Note that MIMO systems with larger numbers of streams will further increase the tags ability to corrupt subframes and extend the range at which WiTAG can communicate.

In summary, if the client and AP are capable of MIMO communication, WiTAG utilizes MIMO to increase its range. However, if MIMO communication is not available, WiTAG leverages dense modulation schemes to enable WiFi backscatter communication for SISO systems.

5 SYNCHRONIZATION

So far we have explained how WiTAG corrupts subframes to communicate its data to a WiFi device. IoT devices typically measure physical variables (such as temperature and occupancy) periodically, and need to transmit it to a nearby WiFi device. In this paper, we refer to the data that a tag wants to transmit as *tag's message*. In this section, we explain how WiTAG synchronizes a query packet with a tag's message.

One solution to synchronizing a query packet with a tag's message is to have the tag detect and distinguish query packets from other WiFi packets. This can be done by transmitting a specific, known bit pattern in the payload of the first few subframes (trigger subframes) of a WiFi packet to indicate that the packet is a query packet. This distinguishes query packets from packets being transmitted by other devices and allows the tag to measure the subframe lengths. The tag then uses a power detector and a comparator to detect the trigger subframes (i.e., the beginning of the query packet) and to determine the timing between two consecutive subframes. This is how previous backscatter systems detect the beginning of a WiFi packet.

This approach significantly limits the practicality of backscatter communication. The problem is that low-power power detectors have low sensitivity and therefore require the tag to be very close to the transmitter to work (i.e., tens of centimeters).[4] Although, there exist power detectors with higher sensitivity, they have high power consumption (i.e., several miliwatts) which is much higher than the power budget of a battery-free device. Therefore, using a power detector is not feasible for WiFi backscatter tags. Although previous WiFi backscatter systems utilize low-power power detectors in their designs, their implementations use power detectors with high sensitivity which have power consumption as high as 21 milliwatts [20, 22, 37, 39].

In the rest of this section, we describe how WiTAG shifts the responsibility of synchronization entirely to the transmitter of the query packet. This eliminates the need for power hungry power detectors and ensures a simple design for the tag. Specifically, instead of having the tag detect the beginning of a query packet, WiTAG synchronizes query packets with the tag's messages which are programmed to be periodic.

5.1 Synchronizing a Query with a Tag's Message

To synchronize a query packet with a tag's message, WiTAG works as follows. The tag wakes up periodically and tries to embed its data. The querying device estimates the wake-up time of the tag and tries to overlap its query packet with tag's wake-up time. To estimate the wake-up times of the tag, the querying device needs to know the tag's first wake-up time and its duty cycle. As we will explain in Section 6.1, the querying device (i.e. AP) discovers these two parameters in the initialization stage. However, there are still two main challenges which need to be addressed to synchronize a query packet with a tag's message. First, the tag's clock drifts. The drift makes it difficult for the WiFi device to estimate the exact tag's wake-up times. Second, sending a WiFi packet at a given time is not guaranteed due to the random nature of channel access. For example, if the querying device estimates that the tag will send its next message at time t_0, it is impossible to know when the query packet must be generated to ensure that it is being transmitted at t_0. This is because the query packet experiences random delays due to channel access protocols. As a result, there is a chance that the tag starts corrupting during the preamble of the query packet, which would be problematic.

To address this problem, instead of transmitting one query packet, the WiFi device transmits multiple query packets during a windows of time that includes t_0, as shown in Figure 7. Further, the tag repeats its message multiple times during its wake-up time. This increases the probability of overlapping a tag's message with at least one of the query packets. Further, the length of each query packet is chosen to be double the tag's message length. In this case, if the tag's clock slightly drifts or if the query packets experience some delays, there will be at least one query packet which overlaps with a tag's message. In Section 8, we show measured distributions of the drift and channel access delay. These distributions are used to determine the number of times a tag's message and query packets need to be repeated.

5.2 Synchronizing a Subframe with a Tag's Bit

So far, we have explained how WiTAG synchronizes a query packet with a tag's message. However, for WiTAG to work, we also need to make sure that each bit of tag's message is synchronized with

[4] Unlike WiFi backscatter, traditional backscatter tags (such as RFID) use low-power detectors and still can achieve a reasonable range (a few meters). This is because RFID operates at much lower frequency than WiFi and hence their signals experience less attenuation, and RFID readers use directional antennas while WiFi devices uses omni-directional antennas.

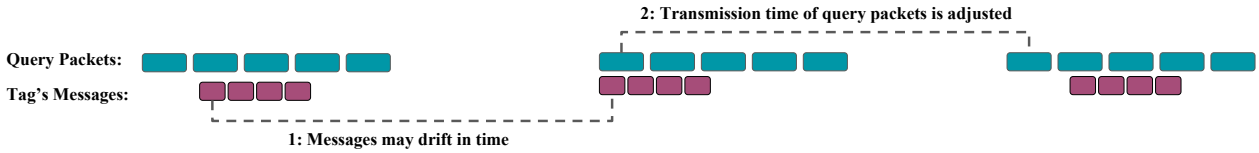


Figure 7: WiTAG synchronization mechanism. The tag wakes up periodically and repeats its message multiple times (4 times in the figure) and the WiFi device also repeats query packets multiple times (5 times in the figure). The WiFi devices adjust the time of query packets based on the tag's clock drift.

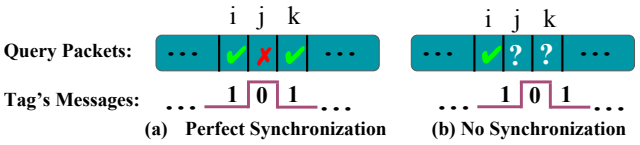


Figure 8: Synchronization of query subframes and tag's bits

a query's subframe. As shown in Figure 8(a), if the tag's bits are synchronized with the query's subframes, it can interfere with a particular subframe accurately. Hence, the WiFi device can decode the tag's message from the block ACK. On the other hand, when a tag's bit is not aligned with a query's subframe, the tag's bit will overlap with two subframes (i.e., j and k in Figure 8(b)). In this case, there is a chance that both or none of the subframes get corrupted, and hence the block ACK will not present the tag's data correctly.

To overcome this problem, we add redundancy bits to the tag's messages. Specifically, instead of assigning a single subframe to each data bit, we assign k consecutive subframes to each data bit. In particular, if the tag wants to send the bit 0, it corrupts k consequent subframes, and if it wants to send bit 1, it does nothing for k consecutive subframes. In this case, even when the bits are not aligned with the query subframes, the WiFi device still can decode the message because there is always at least one subframe that is completely corrupted. Finally, it is worth mentioning that k needs to be at least 2 for the system to work. Larger values of k improve the robustness of WiTAG at the expense of lower effective data rates.

6 OTHER SYSTEM DETAILS

The previous sections presented the two main components of WiTAG, namely, corrupting a subframe, and synchronization. However, a number of system details must be addressed in order to combine these components. In particular: How do we discover the existence of a tag and its duty cycle? How can WiTAG support multiple tags? What is the impact of WiTAG on other WiFi traffic? This section explains how we address these issues.

6.1 Initialization

In the initialization phase, the WiFi device discovers active tags and their duty cycles. Specifically, the WiFi device transmits query packets back to back to receive the message from all tags within range. When the WiFi device discovers all tags, WiTAG switches to the normal operation mode in which the WiFi device transmits query packets only when it expects a tag backscattering a message. The question that remains is how WiTAG discovers the duty cycle (i.e., interval between consecutive wake-ups) of each tag. One approach is to have the WiFi device discover the duty cycle of the tags. In this approach, the WiFi device sends back to back query packets until it receives two different messages from the same tag. The delay between these two messages represents the tag's duty cycle. Although this approach requires sending query packets back to back during initialization, it does not significantly impact other clients on the network since initialization happens only once. A second approach is to use a predefined duty cycle for all tags which is known to the WiFi device. This approach is preferred when the duty cycle is long to minimize the impact on other WiFi devices during initialization. Note that when a new tag is added, the system needs to perform an initialization sequence for that tag to detect the time at which it transmits and its duty cycle.

6.2 Support for Multiple Tags

WiTAG supports multiple active tags to enable applications such as smart home monitoring that require multiple sensors. To distinguish tags from each other, all tags add their ID to the messages they backscatter so that the WiFi device knows which tag generated a received message. This ID can be hardcoded into tags during manufacturing or can be programmed possibly using USB during the initialization phase. Each sensor starts at a random time, therefore it is unlikely that two sensors choose to backscatter at exactly the same time. If this happens, the drift of clocks on the colliding sensors will eventually separate their backscattering times. In Section 8, we evaluate the efficacy of WiTAG in supporting multiple sensors using experimental measurements and simulation.

7 IMPLEMENTATION

7.1 Hardware

We implemented a WiTAG tag on a printed circuit board (PCB) using off-the-shelf components, as shown in Figure 9. The design includes a micro-controller and two RF switches. For the micro-controller, we used ATtiny44 which does not require any external oscillators [3], and it controls the state of the RF switches. For the RF switches, we used SKY 13314-374LF [30]. One switch is used for the 2.4 GHz WiFi band and the other is used for 5 GHz band. Each RF switch is connected to an antenna and two lanes with the length of zero and a quarter of the wavelength. The switch connects the antenna to one of these lanes depending on the control signal to create changes of 0 and 180 degrees in the phase of the backscattered signal (as described in Section 4.2). In all experiments, we use a

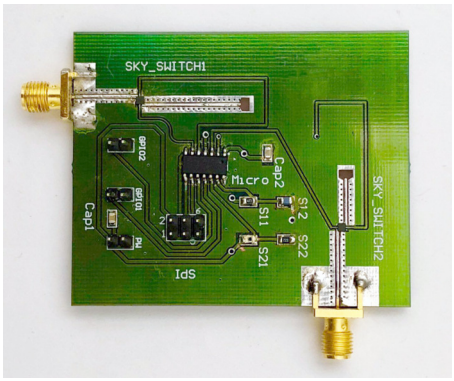


Figure 9: Our custom designed board for WiTAG's tag

typical omni-directional WiFi antenna for the tag. Our tag costs $5.6 USD, which can be dramatically reduced with mass production.

In our evaluations, the WiFi access point is a Google Wifi access point [8] that utilizes the Qualcomm IPQ 4019 [25] chipset (Wave 2 802.11ac). The WiFi client device is a desktop with a TP-Link TL-WDN4800 wireless N adapter. It supports up to three streams (i.e., a 3x3:3 MIMO configuration), and is equipped with standard antennas. All devices used in our evaluations are unmodified commodity WiFi adapters.

7.2 Software

To implement WiTAG, we have developed a C program that runs on a WiFi device that generates query packets. We have made no changes to the hardware or driver of the WiFi device. The WiTAG software has two main components: 1) a query packet generator and 2) the tag's message tracker and decoder. In our implementation, these two components are executed in two separate threads.

The WiFi traffic generator has been implemented using the *pktgen* [24] kernel module available in Linux. Note that other approaches such as iperf [16] and custom traffic generators are also conceivable for generating query packets. We have chosen *pktgen* because it can generate WiFi traffic with low overhead. Since our query packets carry no useful payload, we set the payload size to zero bytes in our implementation to increase the number of subframes we can transmit. This improves the throughput of our system and minimizes the impact on other WiFi devices.

To implement the message decoder component, we utilize TCPDUMP [17] to receive the block ACKs (of query packets). Similar to the traffic generator component, there are other options for reading the block ACKs but we have chosen TCPDUMP because it is available on many platforms and it does not cause any noticeable overhead. The only drawback of using TCPDUMP is the delay it incurs for delivering the packets to our software. In Section 8.1, we investigate this delay and show that it does not cause problems in our prototype.

The TCPDUMP data is fed into the message decoder component of our program that handles two main tasks. The first task is to keep track of the timing of tags' messages including any possible drift in their time and to schedule the next round of query packet generation accordingly. The second task is to detect and extract tags' messages from block ACKs. To support multiple tags, we include a tag ID in the tag's messages to distinguish messages coming from different tags.

8 EVALUATION

8.1 Synchronization Timing Accuracy

Tag timing accuracy: Instead of using a power detector to detect the beginning of query packets, a tag periodically wakes up, and the WiFi device synchronizes its query packets with the tag's message. Therefore, it is important to make sure that the tag's clock is accurate and hence the tag's messages do not experience large drifts. We measure the accuracy of our micro-controller's built-in clock. Specifically, the tag wakes up every second for 20 ms and goes back to sleep. Figure 10 shows the distribution of the error in timing of tag. The figure shows that the maximum error is 200 microseconds. This is much shorter than the duration of a query sent by the WiFi device. Therefore, the tag's message will always collide with the query.

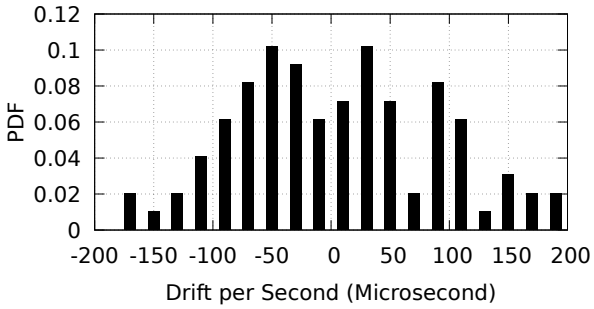


Figure 10: Timing accuracy of tag's clock

Query timing accuracy: Next, we evaluate the accuracy of the WiFi device in scheduling its queries. Specifically, we empirically measure the delay between generating a WiFi packet and when the packet is actually transmitted. We run 1000 measurements and plot the CDF of variations in the delay. Figure 11 shows the results of this experiment. As can be seen, the delay varies from 50 ms to 200 ms. Therefore, in order to make sure that query packets always collide with a tag's trigger, we must request a 150 ms query, 50 ms prior to the expected trigger time. Since the query timing accuracy mostly depends on the WiFi NIC and the operating system, measuring the accuracy once during the initialization of the WiTAG system is enough for calibration.

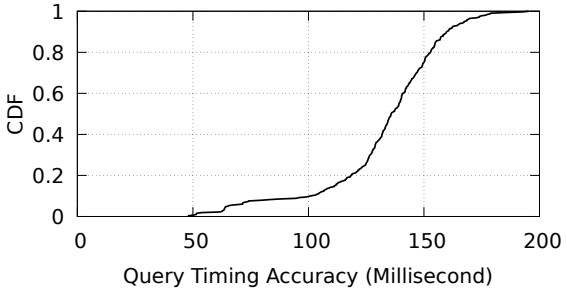
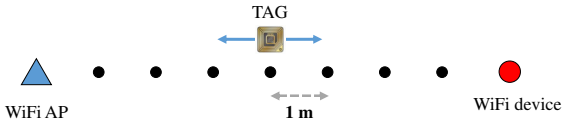
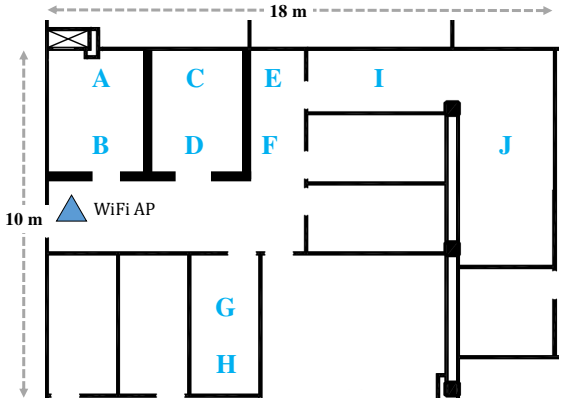


Figure 11: Timing Accuracy of Query Packets



(a) Line-of-sight experiments

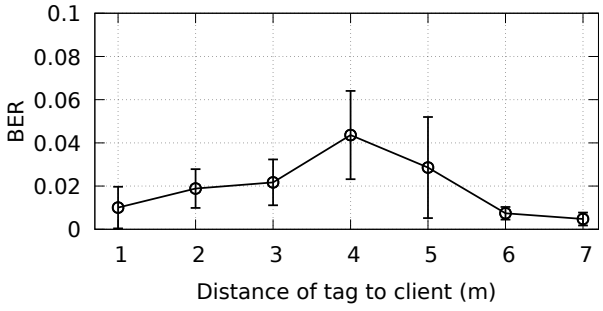


(b) Non-line-of-sight experiments

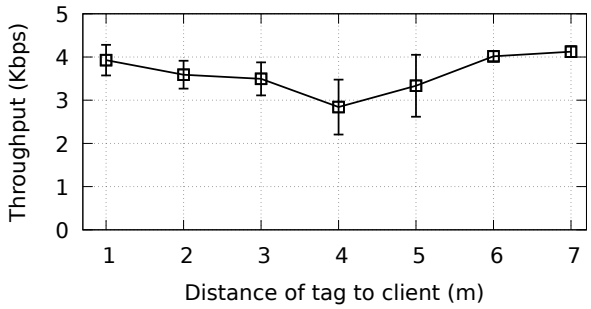
A	B	C	D	E	F	G	H	I	J
38	50	37	41	30	36	35	34	31	19

(c) SNR (dB) for non-line-of-sight locations

Figure 12: Our test bed. (a) line-of-sight experiment, where the AP and client are 8 meters away and the tag is placed between them. (b) non-line-of-sight experiments, where we tested the tag in 10 different locations (A through J). (c) SNR (dB) in each non-line-of-sight location.



(a) Bit error rate



(b) Throughput

Figure 13: BER and throughput of WiTAG in the line-of-sight scenario. The client and AP are 8 meters apart.

8.2 WiTAG's BER and Throughput

We conduct our experiments in lab and office spaces in a building on a university campus as illustrated in Figure 12. We conduct experiments in both line-of-sight and non-line-of-sight scenarios while students move in the vicinity of AP and client devices.

(a) Line-of-sight Scenario: First, we evaluate the performance of WiTAG in terms of bit error rate (BER) and throughput. We place the AP on one side of the room and the WiFi client 8 meters away on the other side, as shown in Figure 12(a). We place the tag between the AP and the client. The WiFi client device continuously transmits A-MPDUs packets, and WiTAG embeds its data into these packets. Query packets are transmitted at the physical layer bitrate of 130 Mbps (i.e., three stream MIMO, 20 MHz channel, and index 5). The AP receives the packets and transmits the block ACKs for the packets to the client. The client extracts the tag's data by examining the block ACK bitmap. We compare the decoded bits with the expected bits to measure the BER. In each measurement, the tag sends data for one minute. We run each experiment 8 times for each of the 7 different locations to compute the BER.

Figure 13(a) shows the BER of WiTAG along with the 95% confidence interval when the tag is placed at different distances from the client. The results show that WiTAG achieves a low BER at all locations between the AP and WiFi client. Specifically, the BER is as low as 0.005 when the tag is close to the AP or the WiFi client. However, it slightly increases when the tag is half way between the AP and client. This is because the strength of a reflected signal (received at the receiver) is proportional to $\frac{1}{D_s^2 \times D_r^2}$, where D_s and D_r are the distance between the reflector object and the sender and receiver, respectively [28]. If the reflector is between the sender and receiver then $D_s + D_r$ is constant and is equal to the distance between the sender and receiver. In this case, $\frac{1}{D_s^2 \times D_r^2}$ is minimized when the reflector is exactly in the middle of the sender and receiver (i.e., $D_s = D_r$). Therefore, because the strength of the reflected signal (received at the receiver) is minimized, the BER is slightly increased. On the other hand, the strength of the reflected signal increases as the reflector is moved closer to the AP or the client, and hence the BER is decreased.

Figure 13(b) plots the average throughput at the seven locations along with the 95% confidence interval. The figure shows that WiTAG achieves a throughput of about 4 Kbps when the tag is closer to the AP or client. The throughput decreases to about 3 Kbps when the tag is in between the AP and client due to higher bit error rates. However, this throughput is still sufficient for enabling battery-free communication for many IoT devices such as temperature, occupancy, and light sensors. Finally, it is worth mentioning that WiTAG achieves much longer range than WiFi Backscatter [14]

which achieves a very limited range.[5] Moreover, although some existing WiFi-based backscatter systems, such as HitchHike, achieve a range similar to WiTAG, and provide throughput of up to 300 Kbps, they require modification to the existing WiFi networks and are not compatible with networks that use security protocols such as WEP and WPA.

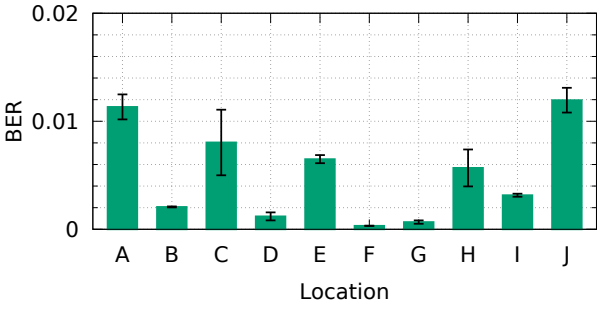
Note, that although WiTAG operates with very low spectrum efficiency compared with WiFi, it has only a negligible impact on the performance of other devices that use the same band. This is due to the fact that WiTAG has been designed primarily for devices (such as temperature or moisture sensors) that only require intermittent communication. In Section 8.3, we show that when two tags transmit as frequently as every 10 seconds, the performance of nearby devices that are streaming video is not impacted.

(b) Non-line-of-Sight Scenarios: Next, we evaluate the robustness and performance of WiTAG in non-line-of-sight scenarios. We place the tag one meter from the client and the AP in another room. We run 15 measurements, each lasting for one minute. We repeat our experiment for ten different tag locations (A through J) as shown in Figure 12. The table in this figure shows the SNR of the WiFi signal (when WiTAG is not operating) at each of these locations. As seen from the values in this table, this experiment covers a wide range of signal qualities from 19 to 50 dB. During these experiments, students work in the lab and move around in the vicinity of the AP and the client. The line-of-sight path between the AP and the client is obstructed by metal cabinets, concrete and wooden walls, and doors.

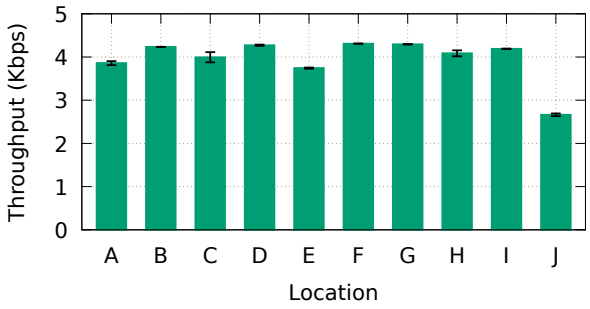
Figure 14(a) shows the average BER and 95% confidence intervals for all ten locations. This figure shows that WiTAG achieves low BERs at all times, even in the presence of obstacles and people that are moving. The BER at location J is slightly higher because there are more obstacles blocking the line of sight and therefore the signal is significantly attenuated. However, WiTAG's performance is very stable over an extended period of time even when the AP and client device are 17 meters apart and the line of sight is completely blocked. Note that the BERs measured in the non-line-of-sight experiments are lower than those measured in the line-of-sight experiments (shown in Figure 13(a)). This is because in non-line-of-sight experiments, the tag is always 1 meter from the WiFi client while in the line-of-sight experiments the distance between the tag and client varies from 1 to 7 meters (as shown in Figure 12).

Figure 14(b) presents the throughput of WiTAG in the non-line-of-sight experiments. The achieved throughput is about 4 Kbps in all locations except location J. At this location, a lower transmission rate had to be used when sending query packets because of the obstacles between the AP and WiFi device. Specifically, the query packets are transmitted at 144.4 Mbps (MCS 15) in all locations, except at locations E and J where the transmission rate is 130 (MCS 20) and 86.7 Mbps (MCS 12), respectively. Since lower transmission rates are used at location J, we can transmit fewer query packets

[5]The range of WiFi Backscatter [14] is very limited because it tries to detect minor changes caused by the reflected signal in the amplitude of the original WiFi signal. Therefore, when the distance between the tag and the WiFi device is large, this change is very small, and hence, not detectable by a WiFi receiver. In contrast, WiTAG uses backscatter signals to change the channel. Although this change can be very small when the tag is far from the WiFi device, it is still large enough to make the subframe undecodable since WiTAG automatically picks a transmission rate that breaks with small changes in the channel.



(a) Bit error rate



(b) Throughput

Figure 14: BER and Throughput of WiTAG in the non-line-of-sight scenario. The tag and WiFi device are placed in different locations of the testbed (shown in Figure 12).

(i.e., subframes) per second. Therefore, the throughput is lower in this location (given that the BER is similar to other locations). The stable throughput in all locations shows that WiTAG enables robust backscatter communication even in non-line-of-sight scenarios.

8.3 WiTAG System Performance

We now evaluate the performance of the WiTAG system as a whole. We place the AP and WiFi device about 2.5 m apart, where there is no line of sight between them. Both AP and WiFi device utilize unmodified 802.11ac cards with 2x2 and 3x3 MIMO capabilities, respectively. Moreover, to evaluate the efficacy and robustness of WiTAG, we have chosen a channel which is heavily utilized by our campus WiFi network. Specifically, we observed 4 other active APs. We also observed a few microwave ovens, one Xbox controller, one Bluetooth device and 4 other non-WiFi devices. We place two tags about 50 cm from the WiFi device. Each tag transmits 10-bit messages every 10 seconds. The WiFi device synchronizes the query packets with the tags' messages as described in Section 5.1. We run this experiment for 17 minutes.

We perform this evaluation for four different scenarios (S1 to S4). In S1 and S3, one and two tags are active, respectively. In S2 and S4, we preform the same evaluations as S1 and S3, but in addition to the existing background WiFi traffic, we inject traffic to further increase channel utilization. Specifically, we add two WiFi devices that stream videos from YouTube for the duration of the experiment.

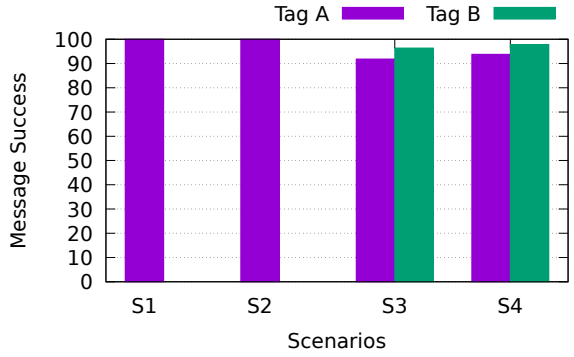


Figure 15: Performance of WiTAG with one and two tags.

Figure 15 shows the percentage of the tag's message that are successfully received by the AP. Despite all the background and injected traffic which increases the delay in channel access, WiTAG delivers all messages successfully when there is one tag. The message success rate decreases slightly when there are two tags. However, the message success rate is still over 92%.

Impact on other clients: Although WiTAG transmits data at 4 Kbps, its impact on the available bandwidth for other users is negligible. This is because WiTAG targets applications such as sensors that read their values infrequently (i.e., every few seconds or minutes). Therefore, occupying the channel for a few millisecond every few seconds has almost no impact on the available bandwidth for other users. Finally, although there is a chance that the tag embeds its messages in packets of other WiFi devices which are close to the tag, its impact on their data rate is negligible since tag's messages are very infrequent. To evaluate the impact of WiTAG on other clients in the network, we monitor the QoS for the clients that stream video on the same channel. We noticed no buffering or degradation of video quality during any of the experiments when a message is backscattered every 10 seconds in these experiments.

8.4 Studying WiTAG Tag Density

As the density of WiTAG tags increases, the probability of two (or more) tags waking up and attempting to backscatter communication at the same time increases. If such a collision occurs the device transmitting the query packet will not be able to correctly decode either message. In this section, we examine a range of deployment scenarios including increasing the density of tags and increasing the frequency of tag communication while studying the probability of such collisions. If tags were capable of communicating at precisely the correct interval in time each and every time, as long as they were initialized to start at different times there would be no collisions. Unfortunately, due to clock drift, even tags that start with different initial times may eventually overlap and cause collisions.

We now describe a simulation study where we examine the probability of collisions with dense deployments of tags (e.g., up to 20 tags), different time intervals at which they wish to communicate (e.g., as short as every 10 seconds), and over varying periods of time (e.g., up to 24 hours). We use simulations for this study because we can more easily study wider varieties and combinations of these parameters. In our simulation, each tag repeatedly tries to embed its message for 20 millisecond. Simulations are conducted using three different time intervals for the tag's sleep/wake cycle. Every 10 minutes, every minute and every 10 seconds. The start time for each tag is determined using a uniform distribution in the time interval being used.

We model the per second clock drift of the tag's microprocessor using a normal distribution with a mean of 0 and standard deviation of 83.3 microseconds. These values were obtained from the experiments discussed in Section 8.1. If the interval over which a tag wakes and backscatters (20 ms) overlaps with one or more other tags the simulator naively assumes that those messages are corrupted and cannot be decoded. In Figure 16, our experiments report the percentage of times that one or more tags overlap during the 20 ms window in which they are backscattering. The results shown are the averages of 10 runs with each run simulating a 24 hour period (except for the 10 second interval case which simulates 3 hours).

The results in Figure 16 show that even with a dense deployment of 20 tags and extremely frequent communication of every 10 seconds, the tags' message failure rate is less than 8%. As expected, the percentage of overlaps decreases as the interval between tag messages increases. For many applications we expect the communication intervals to be significantly higher (e.g., every 1 minute or 10 minutes). In these case there are very few overlaps even with as many as 20 tags in the system.

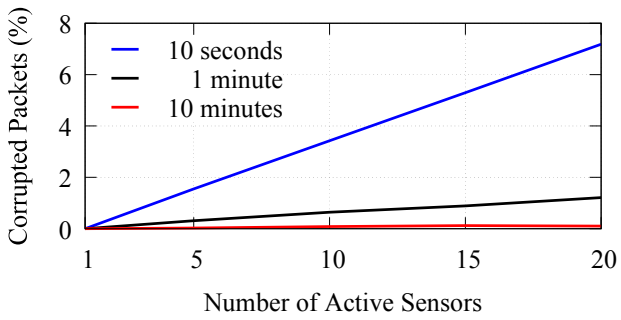


Figure 16: Packet drop rate in percent proportion to the number of active WiTAG tags.

8.5 WiTAG Power Consumption

A WiTAG's tag consists of two components: RF switch and data modulator. The RF switch (SKY 13314-374LF [30]) consumes 9 μW. The data modulator component generates data and the control signal for the RF switch. Previous studies report that the power consumption of such a data modulator is 1 μW when implemented in 45nm ASIC technology [36]. As a result, the power consumption of an ASIC implementation of WiTAG will be 10 μW which is much lower than the ASIC implementation of HitchHick [36] and PassiveWiFi [15] which consume 33 μW and 59.2 μW, respectively.

The major source of power consumption in past WiFi backscatter tags is their clock generation block, which is typically an oscillator [38]. An oscillator's power consumption is proportional to the square of the clock frequency (i.e., the higher the clock frequency, the higher the power consumption). Prior work such as HitchHike [36], FreeRider [37], and MOXcattter [39] need to shift the backscatter signal to another channel. As a result, they require an

oscillator to operate at a minimum of 20 MHz. The power consumption of high-precision oscillators in the MHz range is higher than 1 mW, rendering battery-free implementation impractical [38]. Therefore, instead of using high-precision oscillators, these studies use ring oscillators which consume only tens of micro watts. However, ring oscillators suffer from low accuracy and their frequency is significantly impacted by temperature.[6] Therefore, these systems work only in environments where the temperature is very stable. In contrast, because WiTAG does not require shifting the signal to another channel, it does not require a high-frequency oscillator.

Finally, it worth mentioning that WiTAG can be coupled with existing technologies for harvesting RF energy from WiFi signals. Gudan et al. [9] and Abd et al. [13] design WiFi RF harvesting systems that provides 36.6 μW and 43.8 μW of power, respectively, which are more that the energy requirement of WiTAG. Depending on the use case, WiTAG can also utilize alternative energy harvesting technologies such as indoor or outdoor solar harvesting. A small solar panel can provide 26 mW/cm^2 and 15 μW/cm^2 in outdoor and indoor environment, respectively, which are more than the energy requirements of WiTAG.

9 RELATED WORK

Backscatter communication systems have gained significant attention in the recent years [11, 12]. These systems typically require a specialized reader to generate the trigger signal and to receive the backscattered data. The high cost and large form factor of these readers have made them difficult to deploy and have limited the adoption of RFID tags in many IoT applications. Recent work such as WiFi Backscatter [14], BackFi [4] and Passive WiFi [15] have proposed WiFi backscatter technology which eliminates the need for specialized readers by utilizing WiFi devices instead. However, these systems either have a very limited range [14], require specialized hardware, or require modifications to WiFi devices [36, 37]. Therefore, deploying these systems is costly and impractical for many IoT applications.

More recently, a new category of backscattered systems has been explored that utilizes only commodity WiFi devices. In HitchHike [36], a WiFi device transmits an 802.11b packet that is received by an access point (AP 1) and a tag. The tag embeds its data in the packet by changing the transmitted 802.11b symbols to other valid symbols. The tag also shifts the signal to a non-overlapping channel where another access point (AP 2) receives the backscattered signal. Finally, AP 1 and 2 transfer the received packets to a host where the original and backscattered packets are compared in order to extract the data embedded by the tag.

Although HitchHike utilizes commodity WiFi devices, it is not compatible with existing WiFi networks for a number of reasons: 1) HitchHike can not be used with networks that use security protocols such as WPA and WEP because after HitchHike modifies existing symbols in the encrypted packet, it can no longer be decrypted. 2) The CRC of backscattered packets fail due to modifying the packet payload. Access points would normally drop such packets assuming that these packets are corrupted. Therefore, HitchHike requires modifications to access points to make sure they do not drop these packets. 3) HitchHike only works with 802.11b networks, while most of today's WiFi networks are 802.11n and ac. 802.11b devices use the direct-sequence spread spectrum (DSSS) communication scheme which is fundamentally different from the frequency-division multiplexing (OFDM) scheme used in 802.11n and ac networks. 4) In order to decode backscatter packets, HitchHike requires receiving both the original and backscatter signal. Therefore, it requires an additional access point.

More recent work like FreeRider [37] and MOXcatter [39], propose using similar backscatter communication systems for 802.11g and 802.11n standards. In order for the tag to work with 802.11g networks, FreeRider changes an OFDM symbol to another valid OFDM symbol by changing the phase of the signal. For example, no phase offset represents zero and a 180 degree phase offset represents one. MOXcatter proposes a backscatter system that works with WiFi networks that utilize MIMO communication such as 802.11n and ac. Due to the complexity of MIMO signals MOXcatter cannot perform the phase offset on individual OFDM symbols. Therefore, to transmit 0s and 1s, it instead changes the phase of the signal for each packet. Since FreeRider and MOXcatter reflect the original signal to a secondary channel, they have the same compatibility limitations and shortcomings as HitchHike. FS-Backscatter [38] also shifts the backscatter signal to an adjacent channel to avoid self interference. In FS-Backscatter, a tag transmits its data by reflecting or not reflecting WiFi packets to another channel. As a result, this system also requires a second access point (with software modifications) to receive the backscattered packets.

In contrast with previous systems, WiTAG alters the wireless channel to communicate data by leveraging MAC-layer features. As a result, WiTAG is compatible with existing WiFi networks, requires no modifications to access points, and works with both open and encrypted networks. In addition, because WiTAG does not backscatter to a secondary channel, it does not require high-frequency components, resulting in significantly lower power consumption than previous systems. Finally, in contrast to past studies which use power hungry power detector to detect the beginning of a WiFi packet, WiTAG introduces a synchronization method, eliminating the need for power detectors.

10 CONCLUSION

This paper presents WiTAG, a system that enables backscatter WiFi communication using existing WiFi networks without any software or hardware modification to WiFi access points. WiTAG leverages 802.11 frame aggregation to enable WiFi backscatter which works with the latest WiFi standards (802.11ac and ax), and does not require a second access point. Most importantly, it is compatible with both open and encrypted WiFi networks. Our results show that WiTAG tags are able to communicate their data to a WiFi device, even when used in an office environment with other networks and devices operating in the same channel.

ACKNOWLEDGMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Foundation for Innovation (CFI), the Ontario Research Fund (ORF), and Google. We also thank the anonymous reviewers for their helpful feedback.

[6] For example, a 5°C change in the temperature can shift the frequency by 600 KHz, which significantly increases the error rate of backscatter systems [38].

REFERENCES

[1] M. S. Afaqui, E. Garcia-Villegas, and E. Lopez-Aguilera. 2017. IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi. *IEEE Wireless Communications* 24 (2017).

[2] Narendra Anand, Ryan E. Guerra, and Edward W. Knightly. 2014. The Case for UHF-Band MU-MIMO. In *MobiCom*. 29–40.

[3] Atmel 2015. *8-bit AVR Microcontroller*. Atmel.

[4] Dinesh Bharadia, Kiran Raj Joshi, Manikanta Kotaru, and Sachin Katti. 2015. BackFi: High Throughput WiFi Backscatter. In *SIGCOMM*.

[5] H. T. Friis. 1946. A Note on a Simple Transmission Formula. *Proceedings of the IRE* 34, 5 (1946), 254–256.

[6] Mattew S. Gast. 2012. *802.11n: A Survival Guide*. O'Reilly.

[7] Mattew S. Gast. 2013. *802.11ac: A Survival Guide*. O'Reilly.

[8] Google 2019. *Google Wifi*. Google. https://store.google.com/product/google_wifi.

[9] K. Gudan, S. Chemishkian, J. J. Hull, M. S. Reynolds, and S. Thomas. 2012. Feasibility of wireless sensors using ambient 2.4GHz RF energy. In *SENSORS*.

[10] R. Harrington and J. Mautz. 1967. Straight wires with arbitrary excitation and loading. *IEEE Transactions on Antennas and Propagation* 15, 4 (1967), 502–515.

[11] Haitham Hassanieh, Jue Wang, Dina Katabi, and Tadayoshi Kohno. 2015. Securing RFIDs by Randomizing the Modulation and Channel. In *NSDI*.

[12] Vikram Iyer, Vamsi Talla, Bryce Kellogg, Shyamnath Gollakota, and Joshua Smith. 2016. Inter-Technology Backscatter: Towards Internet Connectivity for Implanted Devices. In *SIGCOMM*.

[13] E. A. Kadir, A. P. Hu, M. Biglari-Abhari, and K. C. Aw. 2014. Indoor WiFi energy harvester with multiple antenna for low-power wireless applications. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*.

[14] Bryce Kellogg, Aaron Parks, Shyamnath Gollakota, Joshua R. Smith, and David Wetherall. 2014. Wi-fi Backscatter: Internet Connectivity for RF-powered Devices. In *SIGCOMM*.

[15] Bryce Kellogg, Vamsi Talla, Shyamnath Gollakota, and Joshua R. Smith. 2016. Passive Wi-Fi: Bringing Low Power to Wi-Fi Transmissions. In *NSDI*.

[16] ESnet / Lawrence Berkeley National Laboratory. 2019. *iPerf - The ultimate speed test tool for TCP, UDP and SCTP*. http://sourceforge.net/projects/iperf/.

[17] Lawrence Berkeley National Laboratory. 2019. *TCPDUMP*. http://www.tcpdump.org/.

[18] Zhuqi Li, Yaxiong Xie, Longfei Shangguan, Rotman Ivan Zelaya, Jeremy Gummeson, Wenjun Hu, and Kyle Jamieson. 2019. Towards Programming the Radio Environment with Large Arrays of Inexpensive Antennas. In *NSDI*.

[19] Kate Ching-Ju Lin, Shyamnath Gollakota, and Dina Katabi. 2011. Random Access Heterogeneous MIMO Networks. In *SIGCOMM*. 146–157.

[20] Linear Technology 2010. *50MHz to 3GHz RF Power Detector with 60dB Dynamic Range*. Linear Technology. Rev. C.

[21] Yunfei Ma, Nicholas Selby, and Fadel Adib. 2017. Minding the Billions: Ultra-wideband Localization for Deployed RFID Tags. In *MobiCom*.

[22] Maxim Integrated 2010. *LF-to-2.5GHz Dual Logarithmic Detector/Controller for Power, Gain, and VSWR Measurements*. Maxim Integrated. Rev. 1.

[23] Konstantinos Nikitopoulos, Juan Zhou, Ben Congdon, and Kyle Jamieson. 2014. Geosphere: Consistently Turning MIMO Capacity into Throughput. In *SIGCOMM*. 631–642.

[24] Pktgen. 2019. https://wiki.linuxfoundation.org/networking/pktgen.

[25] Qualcomm Technologies, Inc 2019. *IPQ4019*. Qualcomm Technologies, Inc.

[26] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. 2002. Opportunistic Media Access for Multirate Ad Hoc Networks. In *MobiCom*.

[27] Wei-Liang Shen, Yu-Chih Tung, Kuang-Che Lee, Kate Ching-Ju Lin, Shyamnath Gollakota, Dina Katabi, and Ming-Syan Chen. 2012. Rate adaptation for 802.11 multiuser MIMO networks. In *Mobicom*.

[28] M.I. Skolnik. 2008. *Radar Handbook, Third Edition*. McGraw-Hill Education.

[29] M. I. Skolnik. 1980. *Introduction to Radar Systems /2nd Edition/* (2 ed.). McGraw Hill Book Co., New York.

[30] Skyworks 2013. *SKY13314-374LF: 0.1 to 6.0 GHz GaAs SPDT Switch*. Skyworks.

[31] Deepak Vasisht, Guo Zhang, Omid Abari, Hsiao-Ming Lu, Jacob Flanz, and Dina Katabi. 2018. In-body backscatter communication and localization. In *SIGCOMM*. 132–146.

[32] Ju Wang, Liqiong Chang, Shourya Aggarwal, Omid Abari, and Srinivasan Keshav. 2020. Soil moisture sensing with commodity RFID systems. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*. 273–285.

[33] Allen Welkie, Longfei Shangguan, Jeremy Gummeson, Wenjun Hu, and Kyle Jamieson. 2017. Programmable Radio Environments for Smart Spaces. In *HotNets*.

[34] Yong Xi, Qingyan Huang, Jibo Wei, and Haitao Zhao. 2007. Rate adaptive protocol for multirate IEEE 802.11 networks. *Journal of Electronics (China)* 24 (05 2007), 289–295.

[35] Y. Xi, B. Kim, J. Wei, and Q. Huang. 2006. Adaptive Multirate Auto Rate Fallback Protocol for IEEE 802.11 WLANS. In *MILCOM 2006 - 2006 IEEE Military Communications conference*. 1–7.

[36] Pengyu Zhang, Dinesh Bharadia, Kiran Joshi, and Sachin Katti. 2016. HitchHike: Practical Backscatter Using Commodity WiFi. In *SenSys*.

[37] Pengyu Zhang, Colleen Josephson, Dinesh Bharadia, and Sachin Katti. 2017. FreeRider: Backscatter Communication Using Commodity Radios. In *CoNEXT*.

[38] Pengyu Zhang, Mohammad Rostami, Pan Hu, and Deepak Ganesan. 2016. Enabling Practical Backscatter Communication for On-body Sensors. In *SIGCOMM*.

[39] Jia Zhao, Wei Gong, and Jiangchuan Liu. 2018. Spatial Stream Backscater Using Commodity WiFi. In *MobiSys*.