

Gateway over the Air: Towards Pervasive Internet Connectivity for Commodity IoT

Jinhwan Jung
KAIST
jhhung@lanada.kaist.ac.kr

Jihoon Ryoo
SUNY Korea
jihoon.ryoo@sunykorea.ac.kr

Yung Yi
KAIST
yiyung@kaist.edu

Song Min Kim*
KAIST
songmin@kaist.ac.kr

ABSTRACT

This paper presents *GateScatter*, the first backscatter-based gateway connecting commodity IoT to WiFi. The backscatter design of GateScatter is an economic option towards pervasive Internet connectivity for ever-growing IoT. The carefully designed tag optimally reshapes ZigBee IoT packets with an arbitrary payload into an 802.11b WiFi packet over the air, such that the payload can be reliably retrieved at the WiFi receiver (hence a gateway). GateScatter is highly compatible – it works with a wide range of IEEE 802.15.4-compliant systems, is agnostic to upper layer proprietary protocols, and does not require any modification to the commodity IoT platforms. GateScatter is extended to BLE IoT for generality. We prototype GateScatter hardware on FPGA where the wide applicability is demonstrated through evaluations on five popular IoT devices including Samsung SmartThings sensor, Philips smart bulb, and Amazon Echo Plus. Further extensive evaluations show that GateScatter consistently achieves throughput above 200 kbps and range of over 27 m under diverse practical scenarios including a corridor, dormitory room, and under user mobility.

CCS CONCEPTS

- Networks → Network management; Cyber-physical networks.

KEYWORDS

Internet-of-Things (IoT); Backscatter; WiFi; ZigBee; BLE

ACM Reference Format:

Jinhwan Jung, Jihoon Ryoo, Yung Yi, and Song Min Kim. 2020. Gateway over the Air: Towards Pervasive Internet Connectivity for Commodity IoT. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20)*, June 15–19, 2020, Toronto, ON, Canada. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3386901.3388949>

1 INTRODUCTION

The Internet of Things (IoT) era is rapidly emerging with the explosive growth of wireless devices covering every corner of our living spaces, expected to reach 20 billion by 2020 [17]. IoT envisions anywhere and anytime service, where pervasive Internet connectivity

*Song Min Kim is the corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MobiSys '20, June 15–19, 2020, Toronto, ON, Canada

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-7954-0/20/06...\$15.00

<https://doi.org/10.1145/3386901.3388949>

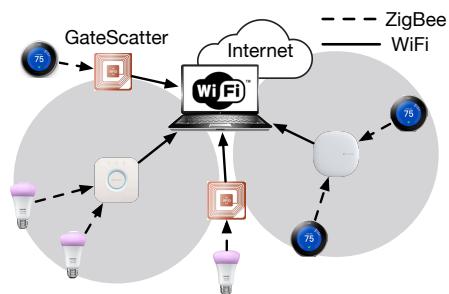


Figure 1: GateScatter covers blind spots of traditional multi-radio gateways, inevitable with an ever-growing body of IoT.

is the key. IoT devices are often equipped with low-power radio (e.g., ZigBee), where they rely on multi-radio gateways for connection to WiFi networks (and to the Internet). Therefore, IoT Internet connectivity is fundamentally throttled by the gateway deployment. In other words, it is crucial to have prevalent deployment of the gateways in order to provide seamless Internet connectivity as IoT continuously grows to extreme scale, becomes increasingly mobile, and produces a greater volume of data.

Despite the criticality of gateway prevalence (i.e., pervasive Internet connectivity) on the performance of IoT services, the current gateways commonly face a few challenges against massive deployment that enforce limited installation in practice: (i) Gateways are equipped with multi-radio interfaces, making them power hungry, and thus typically wall-plugged [45, 48]. This significantly limits outdoor deployment as well as mobility. (ii) Gateways are high-priced, ranging up to 600 USD [12, 21, 45]. (iii) They are vendor-specific – i.e., Different manufacturers have their own gateway. Such incompatibility is another limiting factor against prevalent gateway support. The aforementioned constraints call for a scalable, low-cost, low-energy, and universal (i.e., across vendors) solution which brings IoT Internet connectivity where the current gateway systems fall short.

This paper presents GateScatter, a unique backscatter-based gateway with the aim of offering pervasive IoT Internet connectivity. As depicted in Figure 1, we envision that GateScatter can be a prevalent technology that fills in the inevitable blind spots of the current commercial gateway systems (due to limited deployments). Specifically, backscatter-based design essentially enables prevalent installation of GateScatter at a very low deployment cost (typically < 1 USD per tag) and near-zero maintenance effort via batteryless operation with energy harvesting. Furthermore, GateScatter is compatible with various commodity, readily deployed IoT devices running ZigBee. This is achieved by carefully leveraging common physical layer properties in the wireless signal.

GateScatter seamlessly converts arbitrary ZigBee packets with proprietary upper layer protocols (e.g., [15, 19]) emitted from commodity IoT devices into WiFi signal over the air, while preserving the original bits (hence a gateway). The key technique of GateScatter is *signal reshaping*, which converts ZigBee OQPSK chips into 802.11b WiFi Barker code in an extremely energy-efficient manner. Thus, it can be performed on batteryless tags by energy harvesting. This becomes possible by carefully-designed impedance pattern that selectively and optimally reshapes (i.e., adjusts frequency and phase) only the quadrature of the ZigBee's OQPSK (complex signal) to approximate the WiFi's DBPSK (real signal). In the meantime, the in-phase component is effectively rejected by leveraging orthogonality to suppress noise in WiFi signal. Most importantly, this technique applies to *any* ZigBee packet with an arbitrary payload which need not be known a priori regardless of various proprietary upper layer protocols – an essential property to operate as a gateway. The general applicability of GateScatter as an IoT gateway was demonstrated on five popular commodity IoT systems including Philips Hue, Samsung SmartThings, and Amazon Echo Plus. We also extend GateScatter to support BLE IoT devices and evaluate the design on physical testbeds.

The large body of backscatter work [5, 25, 31, 36, 62, 65] commonly aims at conveying tag data (e.g., readings from sensors attached to the tag). As an example of Interscatter [25], the tag manipulates a dedicated Bluetooth signal (i.e., a single-tone sine wave) into a WiFi frame that carries the tag's sensory readings. On the contrary, tag data is not of interest in GateScatter. Instead, GateScatter focuses on gateway operation of translating and transparently delivering data between two incompatible commodity networks – e.g., ZigBee and WiFi. The key technical difference lies in how to maintain the original content sent by IoT devices and how to recover this content at the WiFi receiver. That is, our design suggests a clean slate approach where backscatter is leveraged to seamlessly reshape the signal over the air to bridge heterogeneous networks. To the best of our knowledge, GateScatter is the first of its kind, where the idea can be generally applied to various networks. Unique technical challenges in the design of GateScatter include: (i) the tag that optimally reshapes an IoT packet into a WiFi packet in the presence of significant bandwidth gaps and modulation differences among the signals, e.g., ZigBee with 2 MHz OQPSK → WiFi with 22 MHz DBPSK (Section 3.3), (ii) selectively capturing quadrature components of the ZigBee signal for signal synchronization with a WiFi receiver (Section 3.4), and (iii) assembling a standard-compliant WiFi packet (Section 3.5). The contributions of our work are three-fold:

- We design GateScatter, a novel backscatter-based gateway that connects IoT devices to the Internet. This is achieved by reshaping ZigBee packets into WiFi while keeping the original contents intact. GateScatter does not require modifications on the commodity IoT devices, indicating high practicality. We further extend GateScatter for BLE to show the generality of GateScatter.
- GateScatter incorporates a technique of signal selection where it selectively backscatters a signal portion of interest (i.e., quadrature). This is a fundamental technique for conversion between complex signals (e.g., ZigBee) and real signals (e.g., WiFi 802.11b).

- We implement GateScatter on FPGA platform for extensive evaluation. We demonstrated GateScatter working as the IoT gateway with commodity IoT devices from various vendors such as Samsung, Amazon, and Philips. Throughput and communication ranges under various environments were also tested, where GateScatter achieved a high throughput of 222 kbps and 662 kbps for ZigBee and BLE IoT devices, respectively, reaching near the maximum throughput limited by the standards. Communication ranges were 27 m and 23 m for ZigBee and BLE, respectively.

Table 1: Vendor-specific gateways

Vendor	Samsung	Philips	Hive	Xiaomi	IKEA
Proprietary	Smart Things	Hue	Hive	Mijia	Tradfri
PHY	802.15.4	802.15.4	802.15.4	802.15.4	802.15.4

2 MOTIVATION

While traditional multi-radio gateways are highly effective, prevalent deployment of such gateways is limited in practice, due to their high power consumption¹ (thus wall-plugged), high cost, and incompatible proprietary protocols. Here, we discuss the possibility of backscatter-based gateway that can be pervasively deployed to augment the current IoT systems.

2.1 Opportunity of Backscatter-based Gateway

Various commodity IoT systems running vendor-specific (=proprietary) protocols (Table 1) share the same physical (i.e., PHY) layer of IEEE 802.15.4 [34]. This applies to diverse standard-compliant IoT systems (e.g., ZigBee [67], WirelessHART [50], ISA100.11a [23], TSCH [1]). Such consistency offers backscatter with a unique opportunity to operate as a unified gateway, by essentially leveraging the known waveform following the same modulation. This is because, applying the fixed and simple signal manipulation at the passband – which is achievable with a backscatter – transforms arbitrary IoT packet into a target signal (WiFi in GateScatter). This indicates a potential of backscatter-based unified gateway compatible with a wide range of IEEE 802.15.4-compliant IoT systems.

2.2 Potential for Pervasive Deployment

Backscatter-based design has significant benefits in power and cost efficiencies. Specifically, as signal is processed in the passband, it consumes only tens of microwatts. This is a three-fold power efficiency compared to traditional multi-radio gateways (with baseband signal processing) that typically consume more than tens of milliwatts [14, 54]. This enables GateScatter to be powered solely by energy harvesting, while power-hungry multi-radio gateways need to be wall-plugged. Furthermore, the cost of backscatters can be projected from commodity RFID tags, ranging at 1-3 USD [2, 51]. Such low power and cost enable GateScatter to be pervasively deployed with a minimum maintenance and deployment cost, to effectively support mobile scenarios and cover blind spots left out by the traditional gateways.

¹SmartThings [48] and Hue [45] gateways consume 3 W and 1.5 W, respectively.

Mobility support. The simple backscatter-based design minimizes the formfactor which can be attached to mobile devices and other personal items in the form of stickers. For instance, attaching GateScatter to a mobile WiFi device (e.g., smart phone or tablet) enables direct interaction with the IoT without the help of traditional gateways. This is especially effective for outdoor scenarios and management of large-scale IoT deployed across a wide region, such as smart farm and smart factory. This is demonstrated in Section 7.7.

Blind spot coverage. Explosive growth of IoT devices deployed in every corner of our living space leads to inevitable blind spots from the coverage of the traditional gateways – thus disconnection from the Internet. Deploying GateScatter on those blind spots restores the Internet access of the IoT devices. In other words, GateScatter is an economic solution for blind spot coverage to push towards pervasive Internet connectivity for the ever-growing IoT. This is demonstrated in Section 7.8.

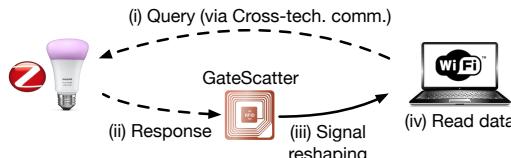


Figure 2: GateScatter operation scenario. No changes are made to the commodity IoT. The WiFi device receives the reshaped IoT packet as the legitimate WiFi packet.

3 GATESCATTER DESIGN

This section discusses the overview of the GateScatter design, followed by the technical background and the detailed design.

3.1 Overview

This section discusses the operation overview of our design via a walk-through example in Figure 2. In this typical scenario, IoT devices upload sensor data in response to a query over the WiFi (or the Internet). (i) The WiFi device (e.g., laptop, smartphone) transmits a query directly to the IoT device. This leverages the recently-announced cross-technology communication [37] which enables a commodity WiFi to transmit ZigBee or BLE packets encapsulated in a WiFi frame payload. Briefly, the technique exploits the high degree of freedom in signal manipulation of WiFi QAM, to approximate the target signals (i.e., ZigBee or BLE). (ii) Upon receiving the query, the IoT device responds with a ZigBee packet carrying sensor data, which is (iii) then reshaped into a WiFi signal in the air via GateScatter. (iv) The WiFi device receives the reshaped signal as the legitimate WiFi packet, and then it retrieves the original IoT bits from the received packet according to the mapping from WiFi bits to IoT bits (see Section 3.6 for details).

GateScatter aims at 802.11b because of two reasons: (i) The data rate of ZigBee and BLE is 250 kbps and 1 Mbps, respectively, and thus the minimum data rate – 1 Mbps of 802.11b is enough to support both ZigBee and BLE. (ii) In order to provide wide compatibility for WiFi devices, almost every WiFi chip is backward compatible. Hence, choosing 802.11b as the target signal makes any WiFi device receive 802.11b packets. It is important to note that IoT side remains as is without any reprogramming or additional operation throughout the entire scenario. This is often the case under practice since most IoT devices are difficult to reprogram due to lack of interface,

if not impossible. Lastly, GateScatter blindly passes IoT data bit by bit to WiFi without data processing or interpretation. Our backscatter approach does not require disclosing packet structures and therefore bypasses security concerns related to protocol exposure.

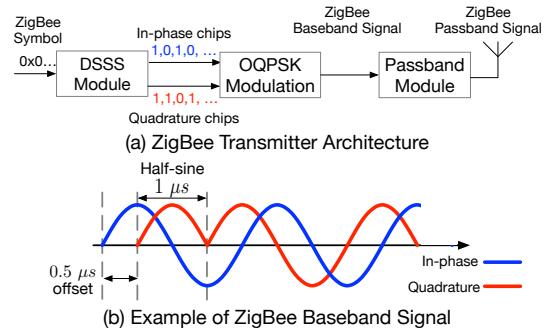


Figure 3: (a) ZigBee transmitter architecture and (b) example ZigBee baseband signal.

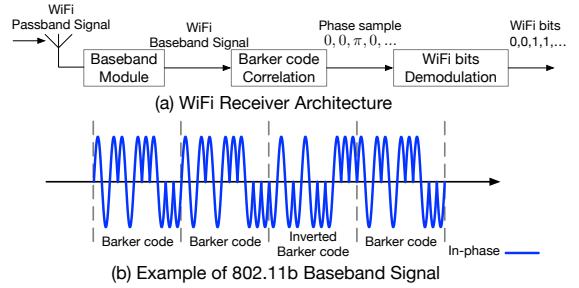


Figure 4: (a) WiFi (802.11b) receiver architecture and (b) example 802.11b baseband signal (Barker code).

3.2 Background

Here, we briefly describe ZigBee and WiFi signals, as a background to understand the GateScatter design.

ZigBee signal. Figure 3(a) depicts the ZigBee transmitter architecture. A ZigBee symbol (0-F) is first spread into 32 chips (either 0 or 1) via the mapping table in the Direct Sequence Spread Spectrum (DSSS) module. Then, the series of chips take turns to be assigned to in-phase and quadrature, where chip 1 and 0 are modulated into 1 μs-long positive and negative half-sine signals, respectively. Figure 3(b) shows an example baseband signal demonstrating ± half-sine signals. Between in-phase and quadrature is an offset of 0.5 μs (thus Offset Quadrature Phase Shift Keying).

WiFi signal. Figure 4(a) illustrates the WiFi (802.11b) receiver architecture. Upon reception, the signal is down converted to baseband. Figure 4(b) is an example of the base WiFi signal made up of 1 μs-long (non-)inverted Barker codes with the phase difference of π . We also note that WiFi 802.11b signal with 1 Mbps only has in-phase (i.e., real signal), as opposed to in-phase and quadrature in ZigBee (i.e., complex signal). The Barker code in the received signal is detected via correlation with the ideal Barker code, where the correlation greater than a certain threshold indicates detection. Finally, the transition between consecutive Barker codes (i.e., non-inverted \leftrightarrow inverted) indicates bit 1, and 0 otherwise (thus Differential Binary Phase Shift Keying).

Backscatter. A backscatter tag is a low-power device which communicates by reflecting the wireless signal in the air, powered by harvesting only or batteries for several years. Recent literature demonstrates that backscatter is capable of modifying amplitude, frequency, and phase of the wireless signal [25, 62], where the reflected signal can be represented as the time-domain multiplication between the passband RF signal (i.e., excitation signal) and the signal generated by the tag (i.e., tag signal). Thus, by manipulating the tag signal, the passband RF signal can be reshaped over the air. The GateScatter is built on top of this foundational technique.

3.3 Signal Reshaping

In GateScatter, the ZigBee signals are first transformed to approximate a WiFi signal; we refer to this process as *signal reshaping*. This is applied bit by bit, without any assumption on the data contained or the packet structure (except for IEEE standard compliance). Signal reshaping operates at the passband of 2.4 GHz, without the need for the power-hungry GHz oscillator (required to bring down the signal to baseband). To that end, GateScatter leverages the property where the frequency and phase transition performed at the passband propagates to the baseband operation at the receiver (i.e., WiFi) and directly affects the decoding. This can be easily derived as follows: let $S(t)$ and f_c indicate the baseband signal and the carrier frequency, respectively. By denoting the GateScatter frequency and phase as f_T and θ_T , the reshaped signal becomes:

$$S(t)e^{j2\pi f_c t} \cdot e^{j(2\pi f_T t + \theta_T)}$$

passband signal tag signal

which is fed into the a mixer and a low-pass filter at the receiver. This simply yields $S(t) \cdot e^{j(2\pi f_T t + \theta_T)}$, indicating that the received WiFi signal directly reflects the frequency and phase of the tag signal. This validates that the signal reshaping can be performed at the passband, enabling backscatter operation.

Signal reshaping holds even between the signals with significant bandwidth gap and disparate modulations. Figure 5 depicts a walk-through example of signal reshaping for the case of ZigBee (2 MHz OQPSK) to WiFi (22 MHz DBPSK). ZigBee's quadrature half-sine signal with 2 MHz bandwidth (Figure 5(a)) is multiplied to the tag signal in Figure 5(b) to correlate with the WiFi's 22 MHz Barker code in Figure 5(c). Quadrature portion of the ZigBee signal (instead of in-phase) is leveraged to exploit the unique quadrature bit sequence for lossless decoding at the WiFi receiver, which we discuss in detail in Section 3.6. Signal reshaping effectively expands the bandwidth of the 2 MHz ZigBee signal by multiplying higher frequency tag signal to follow the 22 MHz WiFi.² In other words, signal reshaping manipulates the ZigBee signal to approximate the WiFi signal, overcoming the significant bandwidth gap.

We note that, when ZigBee signal is negative half-sine signal, the reshaped signal correlates to *inverted* Barker code, and non-inverted Barker code for positive half-sine. This demonstrates the key idea of signal reshaping – the fixed tag signal in Figure 5(b) outputs (non-)inverted Barker codes on the backscattered signal, directly reflecting the original ZigBee data (i.e., \pm half-sine). In other words, interpreting the decoded bits at the WiFi receiver enables

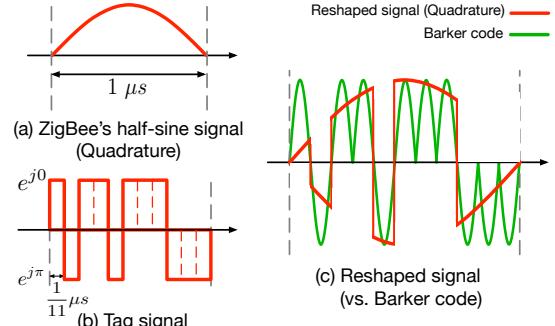


Figure 5: Reshaping the ZigBee signal (i.e., half-sine signal) with phases π or 0 .

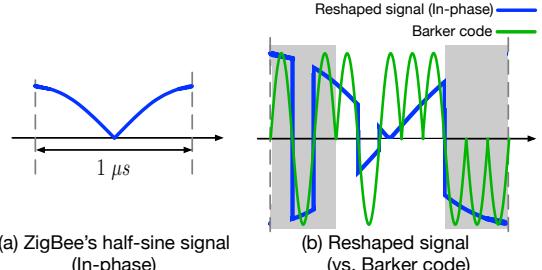


Figure 6: (a) example of ZigBee in-phase signal, and (b) result of signal reshaping.

recovering the original IoT bits (details in Section 3.6). From the implementation feasibility point of view, the tag signal in Figure 5(b) has the dominant frequency component of 22 MHz, with the phase changing between 0 and π at most every $\frac{1}{11} \mu s$ following the Barker code. This can easily be generated on backscatter devices with a low-power ring oscillator, which is known to operate reliably more than 36 MHz with only $9.7 \mu W$ power consumption [25].

WiFi decodes with the correlation threshold of $\frac{2}{11}$ and $-\frac{2}{11}$ [53] to detect non-inverted and inverted Barker codes, where the correlation is computed as the cross-correlation between the received signal and the non-inverted Barker code. The non-inverted \leftrightarrow inverted transition between consecutive Barker codes are interpreted as bit 0 (no transition) and 1 (transition). It is worth noting that the threshold is set to be very low for robustness of WiFi. Due to this low threshold, there are many feasible tag signals that can be used for signal reshaping (i.e., has higher correlation than the threshold). Figure 5(c) depicts an example tag signal, T , that reaches the absolute maximum correlation of 0.69; $T = [1 -1 1 1 -1 1 1 1 -1 -1 -1]$. We let \mathbb{S} be a set of tag signals with the correlation exceeding the threshold. Among them, signal selection optimally picks the tag signal to reshape only the target signal (i.e., the quadrature component). This is discussed in the following section.

3.4 Signal Selection

In addition to achieving high correlation, GateScatter needs to precisely select the target signal to backscatter. This is essentially because the ZigBee signal (OQPSK) contains both in-phase and quadrature components (i.e., complex signal), while WiFi (DBPSK) expects only one of the two (i.e., real signal). To address this, GateScatter selects the quadrature portion of ZigBee (detailed reason in Section 3.6) when it generates WiFi packet containing IoT data. The

²Tag and ZigBee signals are multiplied in the time domain, or equivalently, convoluted in the frequency domain, indicating frequency expansion.

selection should be done in passband, as backscatters are unable to perform baseband processing. We note that, due to passband processing, reshaping simultaneously affects in-phase and quadrature. From the system point of view, selecting quadrature is equivalent to *synchronizing* the WiFi receiver to the reshaped quadrature signal.

Figure 6(a) shows the in-phase component of the ZigBee signal, where Figure 6(b) is the corresponding backscattered signal using the example tag signal T in Figure 5(b). The correlation between the Barker code and the reshaped in-phase is also beyond WiFi's correlation threshold of $\frac{2}{11}$ (i.e., 0.69), which may induce false synchronization to in-phase. To avoid this issue, we carefully reshape (i.e., select tag signal) such that only the quadrature exceeds the threshold. If reshaping is correctly performed, in-phase (below threshold) is naturally rejected at the WiFi receiver, only leaving quadrature in effect. Since \mathbb{S} contains all the tag signals with quadrature above the threshold, the optimal tag signal for signal selection must be in this set. Finding such a tag signal from the set \mathbb{S} can be formulated as an optimization problem defined for the ZigBee half-sine signal duration of $1\ \mu s$. Let $I(t)$ and $Q(t)$ be the in-phase and the quadrature of the ZigBee signal at time t , respectively. Then, for some time t_0 at which the target signal (i.e., $Q(t_0)$) needs to be synchronized, the problem is given by:

$$\underset{T \in \mathbb{S}}{\operatorname{argmax}} \operatorname{Cor}(Q(t_0) \cdot T), \quad (1)$$

$$\text{s.t. } \operatorname{Cor}(Q(t_0 + \Delta t) \cdot T) < C_{th}, \forall \Delta t \in (0, 1\ \mu s], \quad (2)$$

$$\operatorname{Cor}(I(t_0 + \Delta t) \cdot T) < C_{th}, \forall \Delta t \in [0, 1\ \mu s], \quad (3)$$

where for $X(t) \in \{Q(t), I(t)\}$, $X(t) \cdot T$ is a multiplication between $X(t)$ and T for $1\ \mu s$ in the time domain, or equivalently, $X(t) \cdot T$ reshaped with T . $\operatorname{Cor}(X(t) \cdot T)$ is the correlation value between $X(t) \cdot T$ and the Barker code, and C_{th} is the correlation threshold of the WiFi receiver. The intuitive explanation of the optimization problem is to select the optimal tag signal by setting the tag signal to 0 (i.e., let the tag absorb the energy without reflecting) for the parts at which in-phase has the high correlation with the Barker code, where such segments are denoted as gray boxes in Figure 6(b). We denote the optimal tag signal as $T_{sel} = [0\ 0\ 0\ 1\ -1\ 0\ 1\ 1\ 0\ 0\ 0]$. By doing so, the optimal tag signal achieves: (i) maximizing the correlation of the target quadrature signal $Q(t_0)$ at time t_0 so as to be synchronized correctly (Equation 1), (ii) limiting the correlation under the threshold at time $t_0 + \Delta t$ for any $\Delta t \neq 0$ (Equation 2), and (iii) making the in-phase signal (i.e., $I(t_0)$) always have the correlation below the threshold so that the WiFi receiver avoids being synchronized with the in-phase signal (Equation 3). This tag signal is validated to satisfy the optimal conditions, and is used for GateScatter design.

Figure 7 demonstrates empirical measurement of the time and frequency domain representations of the backscattered signal, under the optimal tag signal T_{sel} . (a) clearly shows that in-phase signal is suppressed for successful selection of quadrature, where (c)-(d) demonstrate the similarity between the backscattered signal and the actual Barker code. In summary, by implementing the tag signal of T_{sel} , GateScatter reshapes ZigBee's positive and negative half-sine quadrature signals such that it can be reliably decoded as non-inverted and inverted Barker codes at the WiFi receiver.

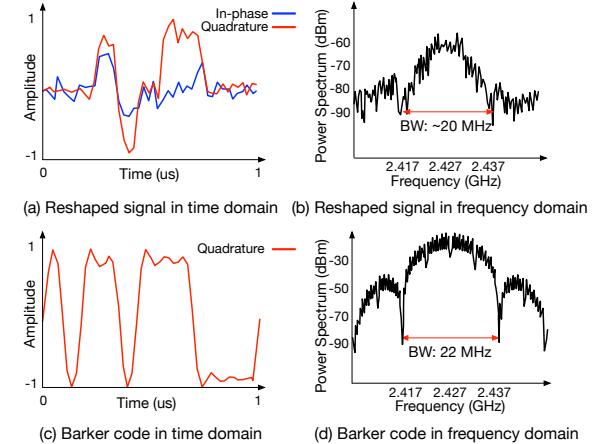


Figure 7: Empirical measurement on backscattered signal vs. Barker code in the time and frequency domains.

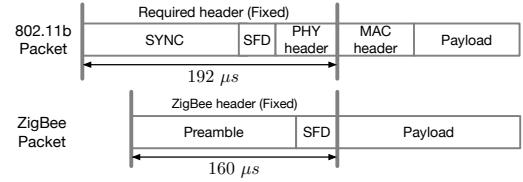


Figure 8: WiFi (802.11b), ZigBee (802.15.4) packet structures.

3.5 WiFi Packet Assembly

To assemble a WiFi (802.11b) packet, a valid WiFi header should be constructed, where GateScatter exploits the predefined nature of both ZigBee and WiFi headers. Since both headers are known a priori, assembly only involves a fixed sequence of signal reshaping.

Figure 8 presents ZigBee and WiFi packet structures and how the ZigBee header is reconstructed to various fields of the WiFi packet. Among various fields, SYNC, SFD, and PHY header are strictly required in order for a packet to be received by the WiFi (i.e., decoded and passed to the application), where they are predetermined³. GateScatter assembles the fields from the ZigBee header (preamble and SFD), which is also predefined. Hence, the packet assembly turns out to be a process of reshaping between fixed ZigBee and WiFi headers. Note that the ZigBee header is shorter ($160\ \mu s$) than that of the WiFi ($192\ \mu s$). This is addressed simply by constructing a slightly shorter SYNC field of $96\ \mu s$ (out of $128\ \mu s$) – our experiment showed negligible impact as SYNC length is defined conservatively.

Recall that signal reshaping uses the optimal tag signal T_{sel} to convert positive and negative half-sine ZigBee signals to be decoded as non-inverted and inverted Barker codes, respectively. Conversely, by adopting T_{sel} and $-T_{sel}$ and given the predetermined sequence of positive and negative half-sine ZigBee signals, GateScatter is able to reconstruct the non-inverted and inverted Barker codes pattern (i.e., WiFi bit pattern) that corresponds to the target WiFi header. In other words, WiFi header reconstruction from the ZigBee header is merely presetting a sequence of T_{sel} and $-T_{sel}$. Finally, converting the ZigBee payload to WiFi is achieved by simply keeping the tag signal as T_{sel} , which yields inverted and non-inverted Barker codes depending on the variable payload data.

³In PHY header, modulation and packet length are always set to DBPSK and the maximum packet size, respectively, to ensure reception of the entire ZigBee bits.

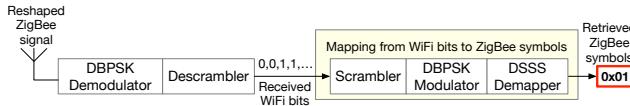


Figure 9: Procedure to retrieve the original ZigBee symbol at WiFi. The example ZigBee symbol 0x01 is successfully retrieved at the WiFi device.

3.6 WiFi Receiver

The reconstructed WiFi packet is fully compatible with any commodity WiFi device and therefore, the packet is decoded as is. The last step needed to operate as a gateway is retrieving the original ZigBee symbols from the received WiFi packet. Figure 9 illustrates the procedure to retrieve the ZigBee symbols at the WiFi receiver from the reshaped ZigBee signal. As noted earlier in Section 3.3, the signal is reshaped such that the WiFi DBPSK demodulator synchronizes to the quadrature component of the signal from which the received WiFi bits are extracted. At the receiver, as defined in the 802.11b standard [22], the reshaped signal passes through the DBPSK demodulator and descrambler with the fixed scrambling seed⁴ where the result from the descrambler is received WiFi bits.

In order to retrieve the original ZigBee symbols, we find a mapping from the WiFi bits to the ZigBee symbols considering different protocol stacks (i.e., ZigBee and WiFi), since the original packet follows the ZigBee stack, but it is decoded by the WiFi stack. First, the WiFi bits should be processed by the scrambler and BPSK modulator so as to recover the original quadrature chips. Since the scrambler and descrambler have the fixed seed defined by the standard, the WiFi device can easily recover the quadrature chips sent by the ZigBee device. For the last part, the DSSS demapper retrieves the ZigBee symbols from those quadrature chips. ZigBee communicates using 16 different DSSS symbols, where each symbol consists of 16 quadrature and in-phase half-sine signals. While DSSS symbols contain both quadrature and in-phase components, we figure out that the symbols can be uniquely identified with only the quadrature portion (this does not hold for in-phase). Thus, GateScatter enables the entire information in the ZigBee packet to be recovered from the WiFi bit sequence naturally decoded on commodity WiFi devices.

More specifically, for every 16 chips from the recovered quadrature chips, the WiFi device computes the hamming distance between the 16 chips and the quadrature components of DSSS symbols, and outputs a symbol that has the minimum hamming distance. Then, the output symbol is the same as the symbol sent by the ZigBee device (e.g., 0x01 in Figure 9), unless link loss occurs. Thus, any ZigBee symbol reshaped by GateScatter can be recovered at the WiFi receiver. Lastly, due to the different packet structures of ZigBee and WiFi, the reshaped ZigBee signal inevitably has a checksum (CRC) error at the WiFi receiver. This issue can be handled simply by configuring the WiFi receiver into monitor mode [4, 8, 46] which allows those packets to be received with the checksum errors. Even though the requirement for disabling CRC may limit GateScatter's applicability, many commodity WiFi devices allow disabling CRC by modifying software without any firmware or hardware modification.

⁴The scrambling seed is determined by [1101100] in the standard.

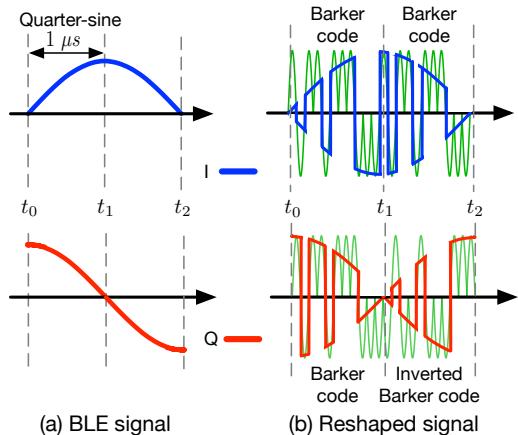


Figure 10: (a) example of the BLE GFSK signal in time domain and (b) reshaped signal by GateScatter vs. Barker code.

4 BLE GATESCATTER

To show the generality of our design, we demonstrate GateScatter for Bluetooth Low Energy (BLE). While the general idea of signal reshaping (Section 3.3) applies to various IoT communication technologies, the signal reshaping needs to be fine-tuned so as to achieve our goal as a IoT gateway for the BLE's Gaussian Frequency Shift Keying (GFSK) [7] signal (vs. OQPSK in ZigBee). BLE's GFSK signal modulates bits by switching between two frequencies, where the phase is changed accordingly⁵. Figure 10(a) illustrates the time-domain representation of BLE's GFSK signal – with 1 MHz bandwidth, bits 1 and 0 are represented as phase offsets of $\pm \frac{\pi}{2}$ for every 1 μ s (e.g., $t_i \rightarrow t_{i+1}$ in Figure 10(a)) corresponding to quarter-sine signals. BLE signals are reshaped into WiFi (802.11b) DQPSK signal by GateScatter.

BLE GateScatter utilizes the same tag signal introduced earlier in Figure 5(b), where the difference from the ZigBee version is that in-phase and quadrature are both utilized to construct WiFi DQPSK. This is because, unlike ZigBee, BLE does not use spreading code (and hence has no duplication) and therefore both in-phase and quadrature components are required to recover the BLE data. Figure 10(b) demonstrates the reshaped signal for in-phase (upper) and quadrature (lower) components. The reshaped signals are compared to the correlated (non-)inverted Barker code as interpreted by the WiFi receiver, from which the BLE signal is recovered.

The example in Figure 10(a) shows a phase shift of $-\frac{\pi}{2}$ in $t_1 \rightarrow t_2$, indicating BLE bit 0. This can be inferred from the received (non-)inverted Barker codes pattern at the WiFi; i.e., two consecutive non-inverted Barker codes for in-phase between t_0 and t_2 , where it is non-inverted followed by inverted for the quadrature. There exist 16 combinations of such (non-)inverted Barker codes, each of which either indicates BLE bit 1 or 0. This is applied to the entire packet to recover the BLE data at the WiFi receiver. Lastly, the header in BLE packets is too short (56 μ s) to fully reconstruct the WiFi header (192 μ s). To address this issue, BLE simply starts the payload with the predetermined bit sequence, which is collectively (with the BLE header) reshaped to a legitimate WiFi header.

⁵Frequency and phase shift keying are interchangeable, i.e., $\cos(2\pi(f \pm f')t) = \cos(2\pi ft \pm \Phi(t))$, where $\Phi(t) = 2\pi f't$.

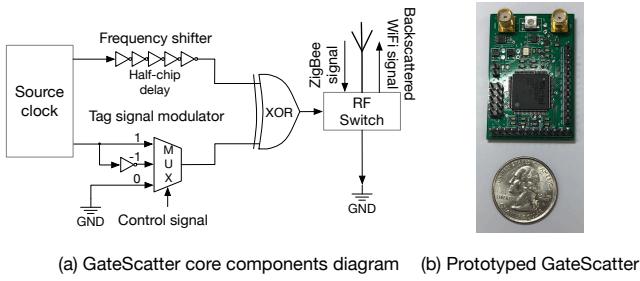


Figure 11: GateScatter hardware design.

5 IMPLEMENTATION

In this section, we present the implementation details and entire process of GateScatter.

5.1 Hardware Design

We implement GateScatter by redesigning the open source design presented in HitchHike [62]. Figure 11 illustrates the core component, where we focus the discussion on ZigBee backscattering. We use a ring oscillator based PLL module [25] to provide clocks for the entire system including the frequency shifter and tag signal modulator. As the frequency shifter, the clock is delayed to reflect the $0.5 \mu\text{s}$ offset of the quadrature signal and connected to the XOR gate. To implement our tag signal, we note that the tag signal modulator requires three operations: 1 (e^{j0}), -1 ($e^{j\pi}$), and 0. Delay at the '-1' path implements the phase offset π . The absorption of the RF signal, denoted by 0, is implemented by matching the impedance of an antenna, such that the RF signal is absorbed into the ground. In our prototype, we use Igloo Nano AGLN250 FPGA [42] to control MUX so that the output of the tag signal modulator becomes the optimal tag signal T_{sel} . The delayed clock for shifting frequency and the optimal tag signal are multiplied through the XOR gate, and then injected to the RF switch, where we choose ADG902 switch [3] to backscatter or absorb the RF signal. As a result, a ZigBee signal is backscattered into a WiFi signal. BLE GateScatter also uses the same system with a slightly adjusted FPGA control signal.

5.2 Frequency Shifter

GateScatter shifts the frequency of backscattered signals to WiFi channels 4 or 8 so as to (i) avoid the self-interference by IoT signals, and (ii) avoid interference from and to WiFi as much as possible [31, 65], since WiFi normally uses the channel 1, 6, and 11 to avoid overlaps. By doing so, we make GateScatter and WiFi networks co-exist (see the evaluation results in Section 7.5). Moreover, when GateScatter backscatters the signal with shifting the frequency, it also generates mirror copies of the backscattered signals as addressed in prior works [25, 62], incurring frequency inefficiency. Thus, we adopt the idea from those works [25, 62] to generate single sideband backscattering. Thus, GateScatter does not disturb communication on unintended frequencies.

5.3 Putting Everything Together

Figure 12 shows how GateScatter works. To collect data from an IoT device, (i) a WiFi device directly queries the IoT device via CTC (Cross-technology Communication), i.e., WEBee [37]. This is done

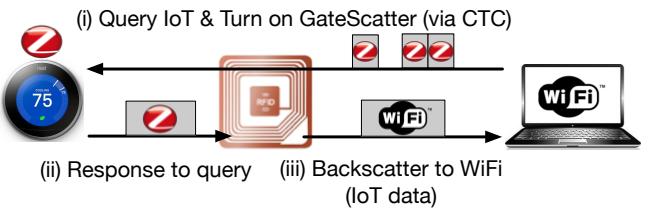


Figure 12: (i) a WiFi device sends query packets to an IoT device (e.g., ZigBee) via CTC, where those packets are to turn on GateScatter using OOK. (ii) the IoT device responses to the query. (iii) GateScatter backscatters the ZigBee signal to a WiFi signal.

in a series of short packets modulating OOK (On-Off Keying) to turn on GateScatter. To decode OOK (represented by the presence and the absence of packets) at the tag, we choose an ultra-low-power envelope detector design similar to the state-of-the-arts [25, 30, 39]. After it is turned on, GateScatter waits to begin backscattering. (ii) Upon receiving the query, the IoT device replies with the requested IoT data. The beginning of this packet is captured by the GateScatter using the envelope detector, where (iii) it then starts backscattering the IoT packet into a WiFi packet of IoT data.

We note that GateScatter is a general solution applicable to a wide-range of commodity IoT devices, where a WiFi-side software (installed on the WiFi device) should be provided by the vendor. This is because both (i) and (iii) require knowledge on the upper-layer protocols, including encryption and error detection algorithms, which are often proprietary and closed.

6 DISCUSSION

In this section, we discuss GateScatter compatibility to encryption and error detection code adopted in commodity IoT packets and operating GateScatter in a non-disruptive fashion.

6.1 Compatibility to Encryption/Error Code

Commercial IoT vendors adopt proprietary encryption and error detection codes (e.g., CRC) for privacy and reliability. GateScatter works transparently to such algorithms, as IoT packets are converted to WiFi in a bit-by-bit fashion. This uniquely enables GateScatter to operate as a unified gateway across IoT vendors with proprietary algorithms. This is because GateScatter keeps the original encryption and CRC intact, regardless of the applied algorithms. In fact, GateScatter avoids security concerns for vendors as it does not require the algorithms to be revealed at all.

6.2 Non-disruptive Operation

GateScatter requires a WiFi receiver to be configured in monitor mode so as to receive the backscattered IoT packet with a checksum error. This implies disconnection of the WiFi device from the AP. This impact can be minimized by operating GateScatter on demand. This is a feasible approach since: (i) GateScatter operates in a query-response fashion, thus IoT data can be collected on demand only when a short WiFi disconnection is affordable (e.g., when WiFi connection is idle). (ii) IoT data is sparsely generated at a level of seconds-minutes [13, 41], and therefore only occasional switch to monitor mode suffices.

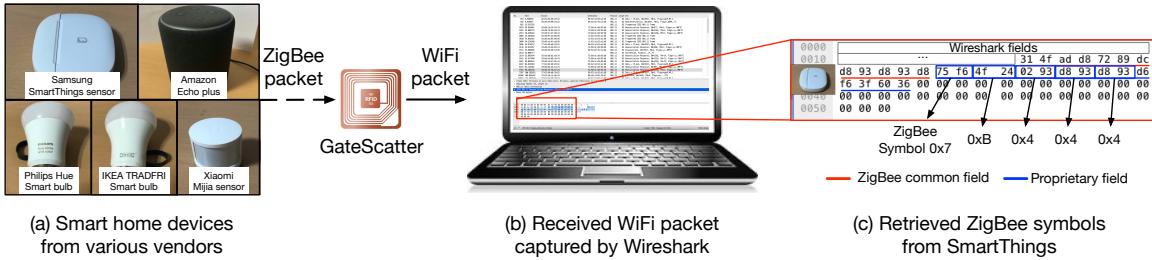


Figure 13: Experiment on commodity IoT devices. ZigBee packets sent by those commodity devices are reshaped by GateScatter to WiFi packets, the laptop receives those WiFi packets, and ZigBee symbols are retrieved from WiFi bits.



Figure 14: Experimental map for line-of-sight (LoS) and non-line-of-sight (NLoS) scenarios.

7 EVALUATION

To validate the performances of GateScatter, we perform extensive experiments under various environments.

7.1 Experimental Environment

We evaluate GateScatter performance using commodity IoT devices (i.e., ZigBee and BLE), a WiFi receiver, and the prototyped tag. As the IoT devices, we choose two types: (i) IoT smart home devices from various vendors to show the possibility to operate as a unified IoT gateway, and (ii) an RF chip CC2650 [55], which is the multi-standard (i.e., ZigBee and BLE) low-power device. The transmission (tx) power of CC2650 can be configured up to 5 dBm, where we choose 5 dBm and 0 dBm in our evaluations which are usually acceptable for low-power IoT devices. To receive 802.11b packets, we use a MacBook Pro laptop which is equipped with a Broadcom BCM4360 WiFi chipset [8]. That commercial WiFi chipset supports 802.11 a/b/g/n/ac, so that it can receive the reshaped 802.11b signals by GateScatter. In addition, we configure the laptop as monitor mode to receive packets with checksum errors. Figure 14 shows the map at which we perform our evaluation with positions of the transmitter (TX), the tag, and the receiver (RX) in both i) the Line-of-Sight (LoS), and ii) the Non-Line-of-Sight (NLoS) scenarios. We measure three metrics: throughput, BER (Bit Error Rate), and RSSI (Received Signal Strength Indicator). The throughput is measured by the number of correctly received bits divided by the number of transmitted bits per second and the BER is calculated by the correctly received bits over the transmitted bits for the successfully detected packets at the WiFi receiver.

7.2 GateScatter Performance as Gateway

Evaluation on IoT Smart Home. To verify that GateScatter works as a unified IoT gateway, we evaluate GateScatter on IoT smart

Table 2: GateScatter performance on commodity IoT

	Smart Things Sensor	Echo Plus	Hue Smart bulb	IKEA Smart bulb	Mijia Sensor
BER (10^{-3})	1.59	1.49	2.58	4.89	7.56
RSSI (dBm)	-74	-68	-80	-78	-80

home devices from various vendors such as Samsung, Amazon, Philips, IKEA, and Xiaomi. In this evaluation, we choose 5 smart home devices which are ZigBee compliant with proprietary upper layer protocols from the vendors. Figure 13(a) shows pictures of the devices. To demonstrate that GateScatter can convert ZigBee packets sent by the smart home devices into WiFi packets, we place the laptop to capture the reshaped WiFi packets and a USRP device with 802.15.4 PHY [49] to sniff the ZigBee packets.

Table 2 shows the evaluation result on the smart home devices with BER and RSSI. We measure the BER performance by comparing the reshaped WiFi packets received by the WiFi laptop and the original ZigBee packets sniffed by the USRP device. RSSI is captured on the WiFi laptop. Figure 13(b) shows the received WiFi packets (enclosing the ZigBee packets from those vendors) at the WiFi laptop (Wireshark screen capture), and the ZigBee symbols are retrieved from the received WiFi bits (Figure 13(c)). These results demonstrate that even though those devices may have different upper layer protocols, GateScatter is compatible with those IoT devices and is able to retrieve the original ZigBee payload (thus operating as a gateway). More precisely, when the IoT devices transmit ZigBee packets which include ZigBee data such as temperature, motion detection, or status reports, GateScatter converts those packets into WiFi packets; thus, the WiFi laptop receives the WiFi packets and reads the original ZigBee data from the packets. In addition, due to the robustness of 802.11b, the achieved BER and RSSI results are enough to support many IoT applications.

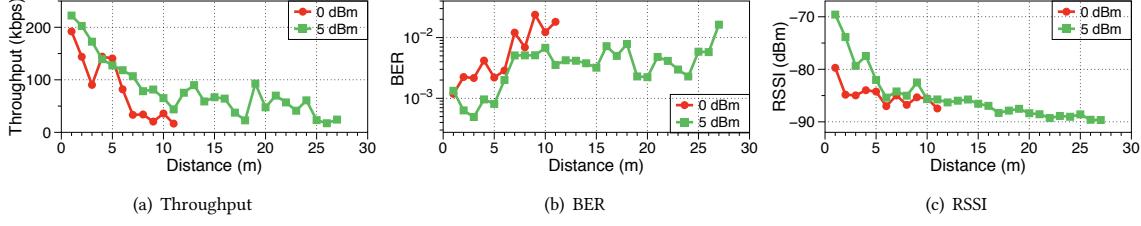


Figure 15: Evaluation of ZigBee to WiFi in the line-of-sight scenario.

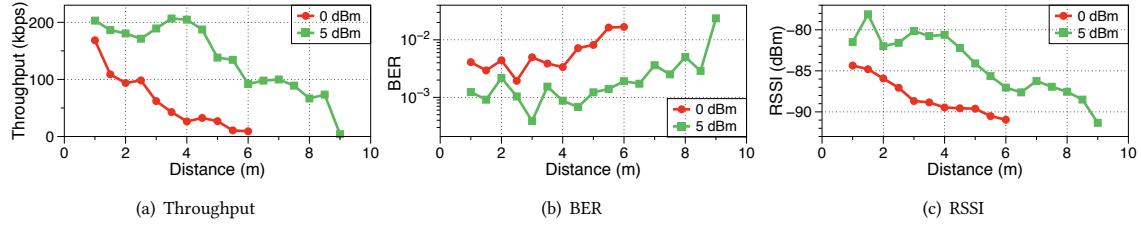


Figure 16: Evaluation of ZigBee to WiFi in the non-line-of-sight scenario.

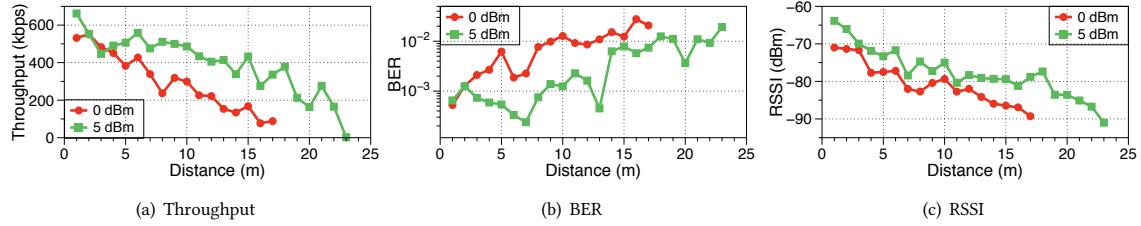


Figure 17: Evaluation of BLE to WiFi in the line-of-sight scenario.

ZigBee to WiFi: Line-of-Sight. Figure 15 shows the GateScatter performance of ZigBee to WiFi communication in the LoS scenario. We fix the distance between TX and the tag as 50 cm, and measure the throughput, BER, and RSSI by increasing the distance between the tag and RX. As shown in Figure 15(a), at the closest distance (1 m), the achieved throughput results are 222 kbps and 192 kbps at the tx powers 5 dBm and 0 dBm, respectively, which are very close to the maximum throughput of ZigBee (250 kbps). As the distance increases, the throughput performance also decreases due to degraded RSSI (see Figure 15(c)), while we achieve the communication distance at most 27 m with the tx power 5 dBm and 11 m with 0 dBm. At the same time, the result of BER is also degraded at longer distances, from 10^{-4} to 10^{-1} as depicted in Figure 15(b).

ZigBee to WiFi: Non-Line-of-Sight. As shown in Figure 14, we also perform the NLoS evaluation of ZigBee to WiFi, where TX and the tag are located in the room, while RX moves along the aisle. We observe that when RX is close to the tag, the achieved throughput performances (203 kbps and 163 kbps at 5 dBm and 0 dBm, respectively) are similar to the result of the LoS scenario, even though the received signal is much weaker than the signal of the LoS scenario (see Figure 16(c)). However, as the distance increases, the throughput and BER performances drop rapidly, so that the communication distance is only achieved up to 9 m and 6 m at tx powers 5 dBm and 0 dBm, respectively, as shown in Figures 16(a) and 16(b). In the NLoS scenario, GateScatter suffers from high BER which requires multiple retransmissions, resulting

in additional energy consumption. Under this environment, we let GateScatter have dwell time after finishing backscattering to deal with the retransmission without starting from the beginning. Due to an obstacle between the tag and RX, although the communication distance becomes shorter, GateScatter can still achieve reasonable communication ranges in various applications (e.g., home IoT, Near-Field Communication (NFC)).

BLE to WiFi: Line-of-Sight. As explained in Section 4, GateScatter can also convert BLE packets into WiFi packets. Under the same experiment environment as the ZigBee to WiFi evaluation for the LoS, we present the result of the BLE to WiFi evaluation in Figure 17. In terms of throughput, GateScatter achieves 662 kbps and 531 kbps at 5 dBm and 0 dBm powers, respectively, as shown in Figure 17(a). Since the maximum throughput of BLE is 1 Mbps, the achieved throughput is much higher than that of the ZigBee to WiFi communication. On the other hand, because there is no coding in the BLE signal, it suffers high BER as depicted in Figure 17(b) while a large portion of bit errors in the ZigBee to WiFi communication can be recovered due to the DSSS coding of the ZigBee signal. It is worth noting that the RSSI of the BLE to WiFi communication is higher than that of the ZigBee to WiFi communication. This is because the GateScatter uses the whole BLE signal to generate the WiFi signal while only half of the ZigBee signal is backscattered to generate the WiFi signal. Interestingly, due to the characteristics of the WiFi receiver which uses the correlation to decode the signal, the available communication ranges are quite similar in both cases.

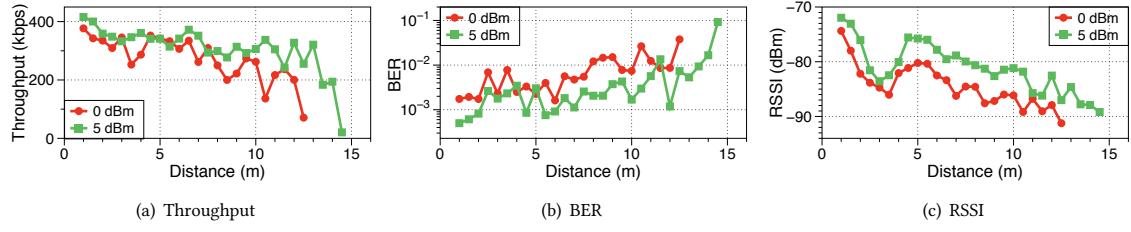


Figure 18: Evaluation of BLE to WiFi in the non-line-of-sight scenario.

BLE to WiFi: Non-Line-of-Sight. Figure 18 shows the evaluation result of BLE to WiFi under the NLoS scenario. As expected, in terms of throughput, BER, and RSSI, all performances are degraded compared to that of the LoS scenario. However, with the existence of the obstacle between the tag and RX, GateScatter achieves the throughput up to 415 kbps and 377 kbps, when TX sends 5 dBm and 0 dBm signals, respectively (see Figure 18(a)). The communication ranges that can be achieved by GateScatter are at most 14.5 m (at 5 dBm) and 12.5 m (0 dBm). The performances of BER and RSSI are depicted in Figures 18(b) and 18(c), which show the same tendency with the previous results. The dwell time can be applied to BLE to WiFi for minimizing energy cost under unreliable environments similar to ZigBee to WiFi. We claim that for low-power IoT devices, the achieved communication ranges which are longer than 10 m are suitable for many applications that require short-range communication.

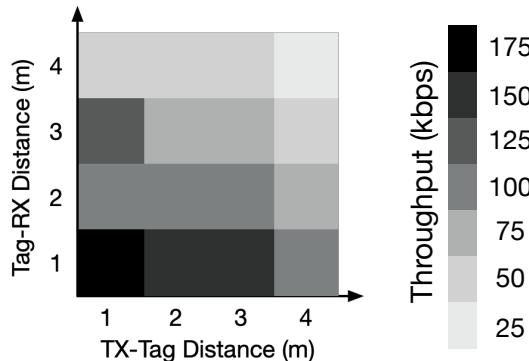


Figure 19: Impact of varying the TX-Tag distance in the ZigBee to WiFi communication.

7.3 Impact of TX-Tag Distance

In the previous evaluations, we fix the distance between TX and the tag (TX-Tag distance). Since the performance of backscatter communication also depends on the TX-Tag distance, we evaluate the impact of varying the TX-Tag distance from 1 m to 4 m with different Tag-RX distances for the ZigBee to WiFi communication. As shown in Figure 19, the throughput performance highly depends not only on the Tag-RX distance, but also the TX-Tag distance. At the closest distances (i.e., both are 1 m), GateScatter achieves about 175 kbps throughput while it degrades as either of the distances increases. When the TX-Tag distance is 4 m, it can achieve up to 100 kbps. Thus, the TX-Tag distance does not severely limit the deployment of GateScatter.

Table 3: GateScatter power consumption

	Oscillator	Modulator	Net power
Power consumption	9.7 μ W	29 μ W	38.7 μW

7.4 Power Analysis

In order to show the feasibility of the low-power operation of GateScatter, we analyze the power consumption of GateScatter, as shown in Table 3. The ring oscillator based clock source (denoted by Oscillator in Table 3), which is used for the frequency shifting, and supplies the clock for the rest of GateScatter consumes 9.7 μ W. The main modulation module of GateScatter including the RF switch (denoted by Modulator in Table 3) consumes 29 μ W power, which is obtained using a power analysis tool provided by the FPGA vendor. In practice, it is possible to fully implement GateScatter by converting the FPGA design to ASIC (Application-Specific Integrated Circuit). The power analysis tool estimates how much power this FPGA consumes when it is implemented as an integrated circuit; thus, we estimate the net power consumption of GateScatter (as implemented in ASIC) at 38.7 μ W. Since energy harvesting devices can produce more than 100 μ W (e.g., from indoor lights [6]), GateScatter can operate without battery as the backscatter gateway or be powered by a coin cell (1000 mAh) for 9 years. When implemented in ASIC, the main difference from our prototype is requiring a customized antenna (e.g., patch antenna), where we believe well-customized patch antennas can provide similar performances.

7.5 Impact on WiFi and GateScatter

In this section, we verify the impact on existing WiFi networks by backscattering of GateScatter and vice versa. One laptop repeatedly transmits 802.11b (1 Mbps) or 802.11n (65 Mbps) packets on the WiFi channel 6 (2.437 GHz). Another laptop which is 1 m away from the transmitter receives the WiFi packets and measures throughput. When a CC2650 transmitter sends ZigBee signals, GateScatter backscatters the ZigBee signal, where the tag is located 1 m away from the receiver laptop.

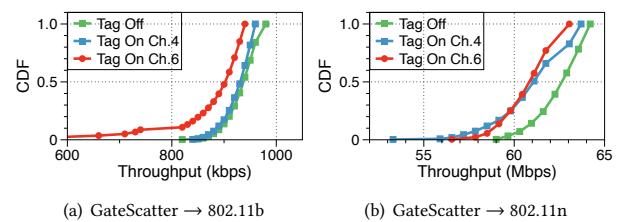


Figure 20: Impact on WiFi networks by GateScatter.

Impact on WiFi networks. As explained in Section 5.2, we shift the frequency of the ZigBee signals into WiFi channel 4 (2.427 GHz) or channel 8 (2.447 GHz) to minimize the interference on WiFi networks, since WiFi uses channels 1, 6, or 11 to avoid overlapping. To validate that, we measure the throughput of 802.11b and 802.11n, when GateScatter does not backscatter (Tag Off), backscatters on the WiFi channel 4 (Tag On Ch.4), or on the WiFi channel 6 (Tag On Ch.6), as shown in Figures 20(a) and 20(b), respectively. In both cases, it is easy to see that the highest throughput is achieved when the tag is absent. When there exists interference from GateScatter (i.e., the backscattered signal is overlapped with WiFi), WiFi throughput performances are degraded, especially for Tag On Ch.6, but it shows still high throughput, since the power of the backscattered signal is relatively small. In other words, because of the capture effect⁶ [33, 35] the stronger WiFi signal can be successfully decoded while the weak backscattered signal by GateScatter is naturally suppressed.

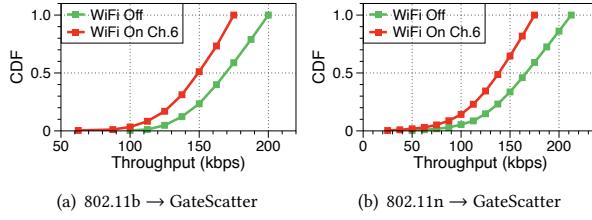


Figure 21: Impact on GateScatter by WiFi networks.

Impact on GateScatter. We also investigate the impact of WiFi interference on the backscattered signal. Figures 21(a) and 21(b) show the throughput performances of GateScatter when the WiFi signal is absent or the signal (802.11b or 802.11n, respectively) is sent on the channel 6. Since the backscattered signal is weaker than the WiFi signal, it is inevitable to have performance loss when there exist ongoing WiFi signals in adjacent channels. However, the robustness of 802.11b makes the WiFi receiver capable of receiving the backscattered signal (converted into 802.11b by GateScatter) on the WiFi channel 4, despite of WiFi interference on the WiFi channel 6 (denoted by WiFi On Ch.6). Moreover, Figure 21(b) shows better throughput performance, since 802.11n occupies smaller bandwidth than that of 802.11b due to OFDM modulation (thus, smaller interference). This is why we choose WiFi channels 4 and 8 to backscatter the ZigBee signal to avoid bi-directional impact on the existing WiFi network and the backscattered signals.

GateScatter with multiple IoT devices. GateScatter is naturally applied under multiple devices by virtue of CSMA for collision avoidance. That is, assume that multiple IoT devices and GateScatter tags are deployed in a certain area. Because of carrier sensing, the IoT devices transmit signals one at a time, so that collisions among the IoT devices are naturally minimized. Hence, GateScatter operates as is without collision, unless CSMA fails.

7.6 GateScatter with Multiple Tags

In this section, we show that deploying multiple tags improves the GateScatter performance of the throughput and communication

⁶When there exist more than two wireless signals, the stronger signal of them will be demodulated.

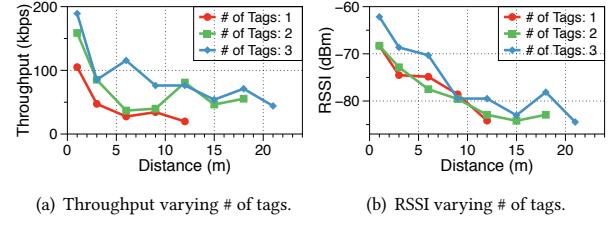


Figure 22: Evaluation on multiple tags.

distance. Under our operation scenario, it is easily assumed to deploy multiple tags on site for pervasive IoT gateways, since the tag is very cheap and easy to deploy. In order to verify the performance improvement by multiple tags, we configure the same environment in the ZigBee to WiFi: Line-of-Sight evaluation with varying the number of tags from 1 to 3, where the tx power is 0 dBm.

Figure 22 shows GateScatter performances in terms of the throughput and RSSI. As the number of tags increases, we observe that the throughput performances as well as the communication distances are improved up to 189 kbps and 21 m, respectively. Obviously, the more tags are deployed, the better RSSI can be observed due to aggregated backscattered power from multiple tags. This result indicates that whenever an IoT device transmits a packet, some of deployed tags nearby the IoT device can start to backscatter so as to convert the IoT packet into a WiFi packet. Since GateScatter targets 802.11b which is resilient to multipath fading, the aggregated power of the backscattered signals can help GateScatter to improve the performance even under multipath fading.

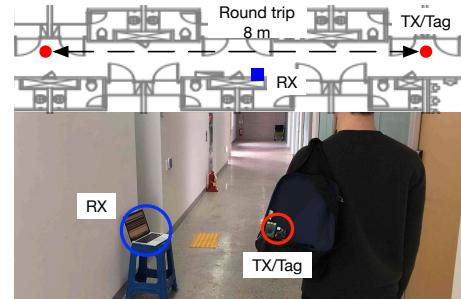


Figure 23: Mobility map of fixed RX scenario. For mobile RX scenario, TX/Tag and RX change their positions.

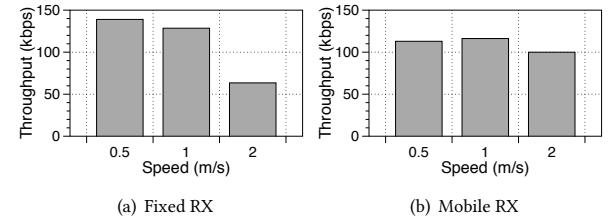


Figure 24: Evaluation of mobile scenarios. (a) RX is fixed, while TX and the tag are mobile. (b) RX is mobile against fixed TX and the tag.

7.7 In-situ Deployment #1: User Mobility

We evaluate GateScatter under mobility. Some IoT applications require data collection from moving objects (e.g., animals and humans) under fixed receivers. In other cases, static IoT sensors remain

static, where receivers are mobile (e.g., smartphone). GateScatter is applicable to both scenarios. We measure the mobility performance as in Figure 23. The first scenario is fixed RX (in blue rectangle), where we move back and forth 8 m between red circles while holding TX and the tag. The second scenario is for mobile RX; we fix TX and the tag, and let RX follow the same mobility pattern in the first scenario. Figure 24 shows the throughput performances in both scenarios with different movement speeds from 0.5 m/s to 2 m/s, where GateScatter consistently achieves up to 140 kbps. Mobile tag and TX decrease the performance by a larger margin due to the bigger impact on the backscattered signal. Under such scenario, the throughput was kept above 63 kbps.

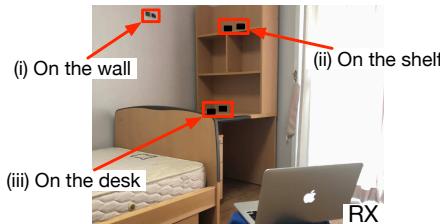


Figure 25: Experimental map: deployed in home. RX is located at the center of the room (at the bottom right). The tag and TX change their positions for each scenario (red boxes).

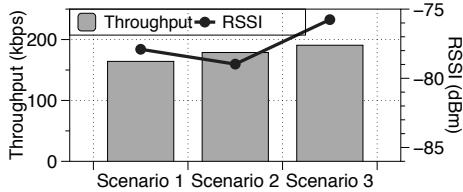


Figure 26: Smart home evaluation: (i) on the wall, (ii) on the shelf, and (iii) on the desk.

7.8 In-situ Deployment #2: Smart Home

As the second application, we consider real deployment scenarios of IoT with GateScatter. When IoT sensors are deployed in our environment (e.g., home, office), by deploying GateScatter near the sensors, we show that GateScatter can operate as a gateway from IoT to WiFi. To validate this application, we place RX at the center of our room and deploy TX and the tag in different positions, where RX and the others are at most 3 m apart, as shown in Figure 25. We measure throughput and RSSI performances for each scenario (see Figure 26). Wherever TX is located, all scenarios with GateScatter show high throughput and reasonable RSSI performances, which are at most 190 kbps and -76 dBm, respectively. Thus, it indicates that by attaching GateScatter around IoT sensors, GateScatter can lead the pervasive IoT gateways to our real life.

8 RELATED WORK

Our work is inspired by the state-of-the-art backscatter works and is assisted by the latest cross-technology communication technique. In the following, we discuss the two streams of works in relation to GateScatter.

Backscatter: In recent years, a large number of backscatter works on various RF sources were proposed, where most of them are between homogeneous wireless systems or require specialized (i.e.,

non-commodity) hardware for signal sources or readers [5, 16, 18, 20, 24, 29, 31, 36, 39, 40, 43, 44, 47, 52, 56–60, 62–65]. For example, Passive-ZigBee [36] requires specialized readers for a tag to generate a ZigBee packet. HitchHike [62] and FreeRider [64] leverage homogeneous devices (e.g., WiFi) as signal sources and readers. Our design is fundamentally different as it considers backscattering among heterogeneous (WiFi, ZigBee, and BLE) commodity devices.

GateScatter is most closely related and inspired by the pioneering work of Interscatter [25], where a single tone signal, generated via a commodity BLE device, is converted to a WiFi signal to read data of the tag. Nevertheless, the technical contributions of GateScatter are clearly distinctive, mainly in two aspects: First, unlike Interscatter, GateScatter converts an *uncontrolled* IoT packet emitted from commodity IoT devices into a WiFi packet, such that the original IoT data is recoverable at a WiFi receiver. Also, GateScatter focuses on unique challenges related to ZigBee OQPSK (complex signal) to WiFi DBPSK (real signal) backscattering, such as reshaping quadrature while suppressing in-phase component.

Cross-Technology Communication (CTC): In recent years CTC was introduced, where it refers to a set of techniques that enable direct communication between different wireless technologies [9–11, 26–28, 32, 37, 38, 61, 66]. Among them, the latest WiFi to ZigBee CTC [37] (implemented on WiFi side) was leveraged in combination with GateScatter for a complete gateway operation. GateScatter is fundamentally different from CTC as it does not require reprogramming and operates silently over the air. This is the key feature that enables GateScatter to be adopted on unprogrammable commodity IoT platforms running proprietary protocols.

9 CONCLUSION

We proposed a novel backscatter-based gateway that connects commodity IoT to the Internet, called *GateScatter*. Our tag is able to convert IoT (ZigBee and BLE) packets to WiFi packets by reshaping IoT signals over the air. The reshaped IoT packets with an arbitrary payload are received on the commodity WiFi receiver, from which the original IoT payload can be recovered at the WiFi side; thus GateScatter operates as an IoT gateway connecting IoT to WiFi. GateScatter is compatible with IEEE 802.15.4-compliant systems and is agnostic to vendor-specific upper layer protocols. We implemented GateScatter prototype based on FPGA and orchestrated with commodity IoT devices and WiFi receivers. We demonstrated the compatibility and wide applicability through the evaluation on smart home IoT devices. We envision that GateScatter will be an economic option towards pervasive Internet connectivity for IoT ecosystems.

ACKNOWLEDGMENTS

We would like to thank our shepherd Dr. Eric Rozner and the anonymous reviewers for their constructive comments and suggestions. This work was supported in part by the National Research Foundation of Korea (NRF) under grant (NRF-2019R1F1A1058947), Institute of Information & Communications Technology Planning & Evaluation (IITP) under grants (No.2016-0-00160, Versatile Network System Architecture for Multi-dimensional Diversity) and (IITP-2020-2011-1-00783, ICT Consilience Creative Program), all funded by the Korea government (MSIT).

REFERENCES

- [1] 802.15.4e Task Group. IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANS) Amendment 1: MAC sublayer. *IEEE Std 802.15.4e*, 2012.
- [2] AEROSPACE INNOTECH. Card Type Active RFID Tag. <http://www.htrfid.com/En/Products/info/id/164.html>.
- [3] Analog Devices. SPST Switch ADG902. <https://www.analog.com/en/products/adg902.html>.
- [4] Atheros. AR9271. <https://wikidevi.com/files/Atheros/specsheets/AR9271.pdf>.
- [5] 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.
- [6] N. A. Bhatti, M. H. Alizai, A. A. Syed, and L. Mottola. Energy harvesting and wireless transfer in sensor network applications: Concepts and experiences. *ACM Transactions on Sensor Networks (TOSN)*, 12(3):24, 2016.
- [7] Bluetooth SIG. Bluetooth Core Specification v 5.0. <https://www.bluetooth.com/specifications/adopted-specifications>, 2017.
- [8] Broadcom. BCM4360. <https://www.broadcom.com/products/wireless/wireless-lan-infrastructure/bcm4360>.
- [9] K. Chebrolu and A. Dhekne. Esense: communication through energy sensing. In *Proc. of ACM MobiCom*, 2009.
- [10] Y. Chen, Z. Li, and T. He. Twinbee: Reliable physical-layer cross-technology communication with symbol-level coding. In *Proc. of IEEE INFOCOM*, 2018.
- [11] Z. Chi, Y. Li, H. Sun, Y. Yao, Z. Lu, and T. Zhu. B2w2: N-way concurrent communication for iot devices. In *Proc. of ACM SenSys*, 2016.
- [12] Dell. Edge Gateway 3000 Series. <https://www.mouser.com/new/dell/dell-edge-gateway-3000>.
- [13] M. Erol-Kantarci and H. T. Mouftah. Wireless sensor networks for cost-efficient residential energy management in the smart grid. *IEEE Transactions on Smart Grid*, 2(2):314–325, 2011.
- [14] ESPRESSIF. ESP32-WROOM-32D. <https://www.espressif.com/en/products/hardware/modules>.
- [15] D. Flowers and Y. Yang. Microchip MiWi Wireless Networking Protocol Stack. *Microchip Technology Inc*, 2010.
- [16] C. Gao, Y. Li, and X. Zhang. LiveTag: Sensing human-object interaction through passive chipless WiFi tags. In *Proc. of USENIX NSDI*, 2018.
- [17] Gartner Inc. Gartner Report. <https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016>.
- [18] W. Gong, H. Liu, K. Liu, Q. Ma, and Y. Liu. Exploiting channel diversity for rate adaptation in backscatter communication networks. In *Proc. of IEEE INFOCOM*, 2016.
- [19] T. group. Thread. <https://www.threadgroup.org>.
- [20] M. Hessar, A. Najafi, and S. Gollakota. Netscatter: Enabling large-scale backscatter networks. In *Proc. of USENIX NSDI*, 2019.
- [21] Hive. Active Heating. <https://www.hivehome.com/products/hive-active-heating>.
- [22] IEEE 802.11 Working Group and others. Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications: higher-speed physical layer extension in the 2.4 GHz band. ANSI/IEEE Std 802.11, 1999.
- [23] ISA. Wireless system for industrial automation: process control and related applications. ANSI/ISA-100.11a-2011.
- [24] V. Iyer, R. Nandakumar, A. Wang, S. Fuller, and S. Gollakota. Living IoT: A Flying Wireless Platform on Live Insects. In *Proc. of ACM MobiCom*, 2019.
- [25] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith. Inter-technology backscatter: Towards internet connectivity for implanted devices. In *Proc. of ACM SIGCOMM*, 2016.
- [26] W. Jiang, S. M. Kim, Z. Li, and T. He. Achieving receiver-side cross-technology communication with cross-decoding. In *Proc. of ACM MobiCom*, 2018.
- [27] W. Jiang, Z. Yin, S. M. Kim, and T. He. Transparent cross-technology communication over data traffic. In *Proc. of IEEE INFOCOM*, 2017.
- [28] W. Jiang, Z. Yin, R. Liu, Z. Li, S. M. Kim, and T. He. Bluebee: a 10,000 x faster cross-technology communication via phy emulation. In *Proc. of ACM SenSys*, 2017.
- [29] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: Internet connectivity for rf-powered devices. *ACM SIGCOMM Computer Communication Review*, 44(4):607–618, 2014.
- [30] B. Kellogg, V. Talla, and S. Gollakota. Bringing gesture recognition to all devices. In *Proc. of USENIX NSDI*, 2014.
- [31] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith. Passive wi-fi: Bringing low power to wi-fi transmissions. In *Proc. of USENIX NSDI*, 2016.
- [32] S. M. Kim and T. He. Freebee: Cross-technology communication via free side-channel. In *Proc. of ACM MobiCom*, 2015.
- [33] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala. Sniffing out the correct physical layer capture model in 802.11 b. In *Proc. of IEEE ICNP*, 2004.
- [34] LAN/MAN Standards Committee and others. IEEE Standard for Local and metropolitan area networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANS). IEEE Computer Society, 2011.
- [35] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proc. of ACM International Workshop on Wireless network testbeds, experimental evaluation and characterization*, 2007.
- [36] Y. Li, Z. Chi, X. Liu, and T. Zhu. Passive-ZigBee: Enabling ZigBee Communication in IoT Networks with 1000X+ Less Power Consumption. In *Proc. of ACM SenSys*, 2018.
- [37] Z. Li and T. He. Webee: Physical-layer cross-technology communication via emulation. In *Proc. of ACM MobiCom*, 2017.
- [38] Z. Li and T. He. Longbee: Enabling long-range cross-technology communication. In *Proc. of IEEE INFOCOM*, 2018.
- [39] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith. Ambient backscatter: wireless communication out of thin air. *ACM SIGCOMM Computer Communication Review*, 43(4):39–50, 2013.
- [40] V. Liu, V. Talla, and S. Gollakota. Enabling instantaneous feedback with full-duplex backscatter. In *Proc. of ACM MobiCom*, 2014.
- [41] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proc. of ACM International Workshop on Wireless Sensor Networks and App*, 2002.
- [42] Microsemi. IGLOO nano low-power FPGA. <https://www.microsemi.com/product-directory/fpgas/1689-igloo-igloo-nano>.
- [43] J. Ou, M. Li, and Y. Zheng. Come and be served: Parallel decoding for cots rfid tags. In *Proc. of ACM MobiCom*, 2015.
- [44] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. *ACM SIGCOMM Computer Communication Review*, 44(4):619–630, 2015.
- [45] Philips. Hue. <https://www2.meethue.com/en-us>.
- [46] Ralink. RT3070. <https://wikidevi.com/files/Ralink/RT307x%20product%20brief.pdf>.
- [47] J. Ryoo, Y. Karimi, A. Athalye, M. Stanaćević, S. R. Das, and P. Djurić. Barnet: Towards activity recognition using passive backscattering tag-to-tag network. In *Proc. of ACM MobiSys*, 2018.
- [48] Samsung. SmartThings. <https://www.smarthings.com/products>.
- [49] T. Schmid. Gnu radio 802.15. 4 en-and decoding. Technical report, UCLA NESL, 2006.
- [50] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt. WirelessHART: Applying wireless technology in real-time industrial process control. In *Proc. of IEEE RTAS*, 2008.
- [51] SYRIS. Active RFID compact asset tag. <http://www.syris.com/product.php?id=259>.
- [52] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota. Lora backscatter: Enabling the vision of ubiquitous connectivity. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):105, 2017.
- [53] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker. Sora: high-performance software radio using general-purpose multi-core processors. *Communications of the ACM*, 54(1):99–107, 2011.
- [54] Texas Instruments. CC2538 A Powerful System-On-Chip for 2.4-GHz IEEE 802.15.4-2006 and ZigBee Applications. <http://www.ti.com/product/CC2538>.
- [55] Texas Instruments. SimpleLink multi-standard CC2650 SensorTag. <http://www.ti.com/tool/TIDC-CC2650STK-SENSORTAG>.
- [56] D. Vasith, G. Zhang, O. Abari, H.-M. Lu, J. Flanz, and D. Katabi. In-body backscatter communication and localization. In *Proc. of ACM SIGCOMM*, 2018.
- [57] A. Wang, V. Iyer, V. Talla, J. R. Smith, and S. Gollakota. FM Backscatter: Enabling Connected Cities and Smart Fabrics. In *Proc. of USENIX NSDI*, 2017.
- [58] J. Wang, J. Zhang, R. Saha, H. Jin, and S. Kumar. Pushing the Range Limits of Commercial Passive RFIDs. In *USENIX NSDI*, 2019.
- [59] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu. Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proc. of ACM MobiCom*, 2014.
- [60] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu. See Through Walls with COTS RFID System! In *Proc. of ACM MobiCom*, 2015.
- [61] Z. Yin, W. Jiang, S. M. Kim, and T. He. C-morse: Cross-technology communication with transparent morse coding. In *Proc. of IEEE INFOCOM*, 2017.
- [62] P. Zhang, D. Bharadia, K. Joshi, and S. Katti. Hitchhike: Practical backscatter using commodity wifi. In *Proc. ACM SenSys*, 2016.
- [63] P. Zhang, P. Hu, V. Pasikanti, and D. Ganesan. Elkhonet: High speed ultra low-power backscatter for next generation sensors. In *Proc. of ACM MobiCom*, 2014.
- [64] P. Zhang, C. Josephson, D. Bharadia, and S. Katti. Freerider: Backscatter communication using commodity radios. In *Proc. of ACM CoNEXT*, 2017.
- [65] P. Zhang, M. Rostami, P. Hu, and D. Ganesan. Enabling practical backscatter communication for on-body sensors. In *Proc. of ACM SIGCOMM*, 2016.
- [66] X. Zhang and K. G. Shin. Gap sense: Lightweight coordination of heterogeneous wireless devices. In *Proc. of IEEE INFOCOM*, 2013.
- [67] ZigBee Alliance. ZigBee Specification. <https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0-csg-zigbee-specification.pdf>, 2015.