SDR-based PHY Characterization of Zigbee Devices

Stefan Gvozdenovic
Electrical & Computer Engineering
Boston University
Boston, MA, USA
tesla@bu.edu

Johannes K Becker
Electrical & Computer Engineering
Boston University
Boston, MA, USA
jkbecker@bu.edu

David Starobinski
Electrical & Computer Engineering
Boston University
Boston, MA, USA
staro@bu.edu

***Abstract*—We demonstrate characterization of Zigbee devices at the physical layer using commodity software-defined radio (SDR) hardware and an open-source software testbed setup. Using this testbed, we run different types of experiments to characterize and distinguish between the performance of four Zigbee devices. Receiver sensitivity experiments show that the devices exhibit differences of up to 2-3 dB in terms of their reception capability. Interframe spacing and capture experiments demonstrate the ability to conduct fine-grained timing experiments, craft multi-packet transmissions, and derive device-specific reception curves. The results of this paper complement previous work on Wi-Fi and illustrate the flexibility of our SDR-based testbed to serve as as a cross-protocol evaluation platform.**

***Index Terms*—benchmarking, wireless, physical layer, capture effect, Internet of Things**

I. Introduction

Zigbee, based on the underlying IEEE 802.15.4 standard [1], has emerged as a popular protocol for the Internet of Things (IoT) [2]. The protocol has been widely adopted for smart home and building automation purposes as well as Industrial IoT (IIoT) use cases, such as sensor networks in construction and energy.

Typically, users rely on vendor-provided gateways to communicate with Zigbee devices. As a result, users have limited direct insight into communication happening between Zigbee devices. Direct monitoring of device communication is possible in specialized laboratories, but usually requires expensive testing equipment. Consequently, direct performance comparisons of chipsets from different vendors are hard to come by and not easy to generate.

We propose a commodity hardware-based, low-complexity testing setup that not only allows for direct communication with Zigbee devices, but also supports physical layer characterization of said devices. This work makes the following contributions:

- We present a desktop-sized experimental setup for measuring Zigbee's physical layer performance, which is low in complexity and cost.
- We demonstrate the measurement capability of our testbed by showing high-resolution differences among commercial Zigbee devices in terms of reception sensitivity.

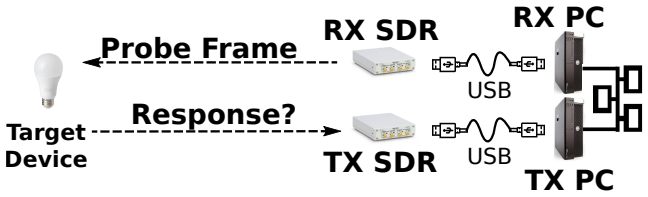


Fig. 1: The SDR instrument controlled by the Host computer communicates bidirectionally with a device under test (DUT), sending probe frames and collecting acknowledgements from the DUT.

- We conduct receiver capture and interframe spacing experiments that allow us to clearly distinguish between the behavior of the devices, as well as fingerprint the chipsets on which they are based.

The rest of this paper is organized as follows: In Section II we describe our experimental setup. In Sections III and IV we explain our methodology and define the experiments conducted in this work, followed by a result discussion in Section V. Section VI discusses related work, and finally Section VII concludes our paper and presents some avenues for further work.

II. Experimental Setup

The experimental testbed contains both a hardware setup, shown in Figure 1, and a software setup. The hardware setup is comprised of three main components:

- Two USRP B210 software-defined radios (SDRs): one to transmit to the device under test (DUT) and another to receive communication from the DUT.
- A host computer controlling the SDR instruments.
- A Zigbee device under test (DUT).

The software system is built on top of GNU Radio [3] and the gr-802154 library [4]. It is comprised of three main components:

- A transmission (TX) flowgraph that generates Zigbee packets which are transmitted to the DUT.

- A receive (RX) flowgraph that parses incoming communication from the DUT, and stores the recovered information into a packet capture (PCAP) file.
- A post-processing script that reads the PCAP file and calculates the necessary statistics from it.

All experiments are conducted inside a Ramsey STE3500 RF Test Shielded Enclosure [5] to prevent external signals from interfering with the experiments. To ensure reproducible measurements and portability of the experimental software setup across different host computers, we run all measurements inside a custom made Docker container. Finally, we use Scapy [6] to craft Zigbee frames.

III. Methodology

Measuring reception rates of smart home devices in a realistic setting without interfering with their normal behavior and without disassembling them to gain access to debugging ports such as the Cree lightbulb teardown [7] poses challenges to the experimental methodology. Typical consumer-grade IoT devices are tightly integrated and typically do not expose any ports for debugging or programmatic interface with the device outside of their intended application-level usage.

Without access to any invasive methods, our experimental results only rely on a device returning some frame type over the air (such as an acknowledgement or a beacon response). Thus, for each DUT, we first determine how a response can be triggered by subjecting the device to various frame types.

We found out that DUTs respond either with a Beacon-type message or an Acknowledgement (ACK) to a specific type of probe frames. The device-specific type of probe frame and typical response is listed in Table I in the columns *Probe Frame* and *Response*. While both of these response types indicate successful reception of the incoming request, only ACKs contain the sequence number of the corresponding Beacon (or Association) request. Thus, if the listening SDR records an ACK with a specific sequence number, that means that a beacon/association request with the same sequence number was successfully received by the DUT. In cases where a DUT responds with both an ACK and a beacon (such as the Ikea bulb), we base our statistics on the received ACK.

IV. Experiments

We run experiments against four Zigbee-based devices. Table I shows the list of devices that were tested. Apart from the EFR32 development board, the other tested devices are common smart home appliances.

A. Receiver Sensitivity

The first experiment characterizes the relative receiver sensitivity of different devices subjected to a range of probe frames with different signal strengths. The signal is varied over a range spanning four orders of magnitude to determine the threshold at which each device can receive packets successfully. At every power level, the same packet is sent 100 times, and responses are collected via the RX flowgraph. To ensure equal path loss, we placed the SDRs and each of the DUTs at an equal distance.

B. Shortest Supported Interframe Spacing

The goal of this experiment is to characterize how fast a device can return to the receiving state after processing an incoming packet (that is, receiving and acknowledging the packet). Toward this end, we send two packets of the same type/length/gain spaced by different offsets. The initial offset is zero, which means the packets are sent with no gap between them (back-to-back). Then, the offset is incremented in steps that are multiples of 250μs until we receive both packets reliably. To know which of the two packets are received, we capture the corresponding acknowledgements. We can always differentiate between the two packets using their sequence numbers. For every offset step of 250μs, we perform 100 test trials. Test trials are spaced 200ms apart and have their own sequence number.

We expect four types of results from these experiments:

1) The receiver cannot process either packet if the packets are sent too quickly one after the other;
2) The receiver processes the first packet, but is unable to receive the second packet;
3) The receiver only successfully processes the second packet;
4) The receiver successfully processes both packets.

C. Capture

A packet capture occurs when reception of a packet with weak power is interrupted in favor of a packet with stronger power. To measure the capture effect, we generate two competing packets and emulate their collision in the transmission flowgraph at various offsets of signal overlap (an approach similar to [8], [9]). The signal strengths of both signals and their offsets in time are tightly controlled relatively to each other, such that the first packet is weaker than the second packet by 20 dB. Then, the acknowledgements of both frames are counted to estimate the corresponding reception rates, as described in Section III. This experiment is similar to the interframe spacing experiment in that the two frames are sent with different sequence numbers. However, the key difference is that: (a) the gains of the two packets are not equal anymore, (b) the initial offset is when the two frames are fully overlapping; the final offset is when two frames are sent with no gap between them (i.e., back to back). The offset is incremented in steps that are multiples of 250μs. For every offset step of 250μs we perform 100 test trials. Test trials are spaced 200ms apart and have their own sequence number.

V. Results

A. Receiver Sensitivity

Results of the receiver sensitivity experiments are shown in Figure 2. We notice a difference of about 2.5dB between the most sensitive device and the least sensitive one. Devices that use the EFR32 chipset show sensitivity within 1dB, but they are still distinguishable by the experimental setup.

TABLE I: Tested IEEE 802.15.4 devices.

Make	FCC ID	System	Probe Frame	Response
Microchip ATSAMR21G18A	2ACQ6-A19	CREE Lightbulb A19	Association Request	Ack
SiliconLabs EFR32™	FHO-ICC-A-1	IKEA LED1732G11	Beacon Request	Beacon & Ack
SiliconLabs EFR32™	FHO-E1526	IKEA TRÅDFRI Gateway	Beacon Request	Beacon & Ack
SiliconLabs EFR32™	N/A	EFR32™ Mighty Gecko Wireless Starter Kit (WSTK)	Beacon Request	Ack

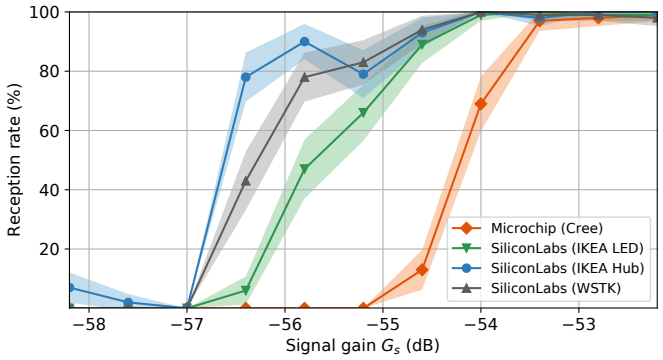


Fig. 2: Receiver sensitivity experiments. Devices based on the SiliconLabs EFR32-based chipset exhibit a more sensitive reception than the Cree device based on the Microchip-based chipset. Shaded areas around the curves represent 95% confidence intervals.

B. Shortest Supported Interframe Spacing

Results of the interframe spacing experiments are shown in Figure 2. The point at which the reception rate starts rising indicates the shortest interframe spacing between the probe frames of the corresponding devices. This interframe spacing includes the first packet's processing time as well as the time to send the corresponding ACK frame. If the gap between frames is shorter than the shortest supported interframe spacing, the transmission of the ACK of the first frame collides with the reception of the second frame, which corrupts them both. Hence, the offset at which the reception rate starts rising can be estimated as the following sum: (a) the length of the first frame (512μs for beacon request or 864μs for association request); (b) the processing time of the first frame (measured with the Universal Radio Hacker (URH) software [10]); and (c) the length of the acknowledgement frame (352μs). Hence, the the estimated rise offset is $864 + 190 + 352 = 1406\mu$s for the Cree, $512 + 250 + 352 = 1114\mu$s for the Ikea light bulb and hub, and $512 + 70 + 352 = 934\mu$s for the EFR32 development board. Our experimental results shown in Figure 2 are in line with these estimations.

C. Capture

Results of the capture experiments are shown in Figure 4. The reception rate of the stronger frame is near 100% for low offsets (close to zero) and high offsets (close to the frame length). At offsets ranging from 0 to 128μs, the frames' preambles collide. The weaker first frame, at a power level which would normally allow its reception, is not received. Instead, a preamble capture effect occurs whereby the radio locks on to the second, stronger frame.

Yet, once the radio locks on to the first frame's preamble and starts decoding, the second frame is not captured as shown by the 0% reception rate in the middle range of the graph. At the end of the tested offset range, however, the second frame is captured again. The length of the middle offset range presumably indicates the turnaround time of the radio state machine.

VI. Related Works

Speers et al. introduced a physical layer testbed called Isotope for IEEE 802.15.4 devices capable of crafting custom preambles and manipulating the header in non-standard ways [11]. In 2018, Knight and Speers presented an RF fuzzing tool based on Isotope which allows access to physical layer modifications using a USRP B210 software-defined radio setup [12]. They modify the existing gr-ieee802-15-4 library [4] to produce arbitrary preamble sequences to test their impact on devices under test. While we do not fuzz 802.15.4 preambles, our work synthesizes multiple packets into one precisely configured signal which can be sent out. This allows for physical layer testing not unlike fuzzing, but more focused on receiver behavior in the presence of multiple transmissions.

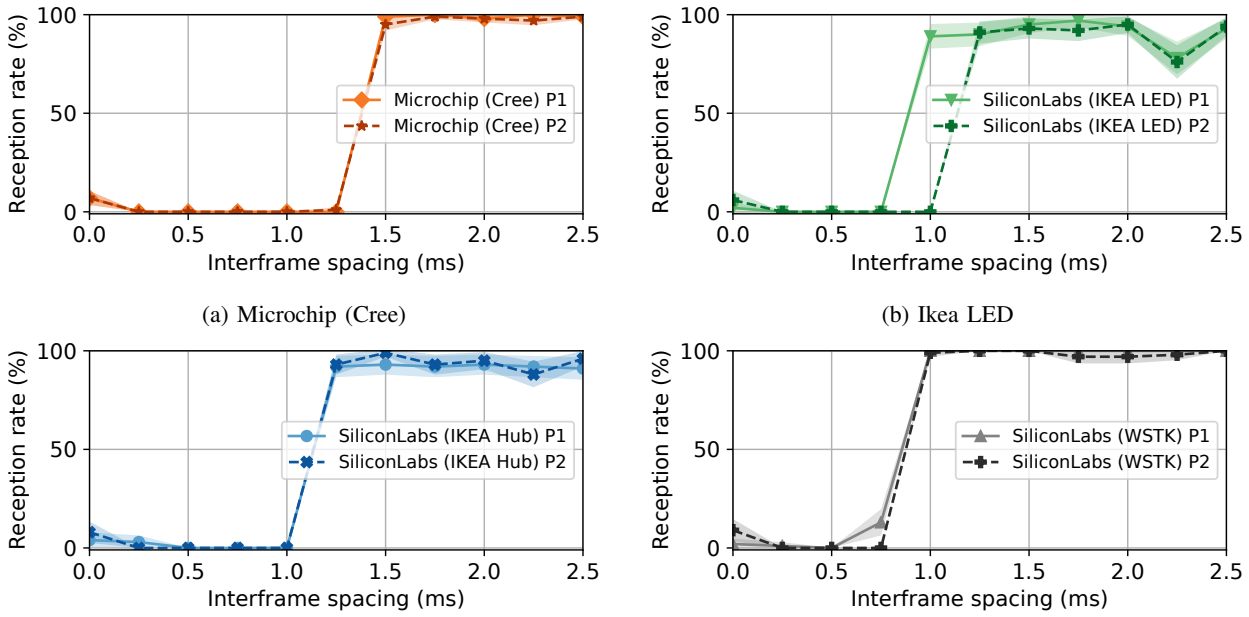
Xin et al. introduced a testbed based on commodity hardware which allows for physical layer wireless testing of Wi-Fi devices using SDR instruments [8], [9]. We generalize the methodology used in that work to the Zigbee protocol and perform both similar as well as entirely different tests on Zigbee devices. In contrast to this previous work, the Zigbee testing setup uses both TX and RX capabilities of the testbed to collect statistics, whereas Xin et al. controlled the DUT directly as a means to capture traffic and collect statistics.

VII. Conclusion

This work presented a testbed setup to measure PHY layer performance over-the-air of arbitrary commercial Zigbee devices. In contrast to prior work on Wi-Fi [8], [9], the testbed does not require any physical control over the tested devices. Therefore, this work could be further extended into more offensive testing scenarios such as denial of service or fuzzing-based approaches. The testbed could also be extended to IoT devices using other wireless protocols.

Acknowledgment

The authors would like to thank John Mikulskis for his contributions to Snout [13] which informed the experimental setup used in this work. This work was support in part by



(a) Microchip (Cree)

(b) Ikea LED

(c) Ikea Hub

(d) EFR32 WSTK

Fig. 3: The shortest supported interframe spacing experiments show how fast a device can return to a receiving state. Each device shows a clear minimum interframe spacing time at which it is capable of receiving a second packet. Furthermore, the reception rates were measured to stay at 100% until interframe spacing reaches 10ms. P1 and P2 indicate ACKs for the first and second packet, respectively.

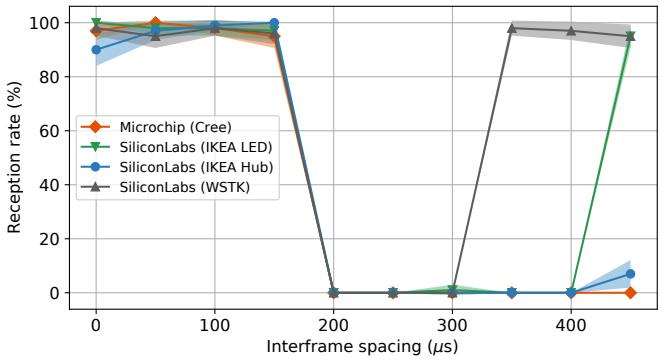


Fig. 4: Reception rates of the stronger (second) frame. The reception rate of the weaker (first) frame is zero throughout the measured range, hence is not shown.

NSF under grants CNS-1409053 and CNS-1908087, and by an Ignition Award from Boston University.

References

[1] IEEE Standards Association, *802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks*, IEEE, Ed., New York, New York, USA, 2015.

[2] Market Research Future, "ZigBee Market Research Report - Forecast to 2023," *Market Research Future*, no. May, 2020. [Online]. Available: https://www.marketresearchfuture.com/reports/zigbee-market-2617

[3] E. Blossom, "GNU radio: tools for exploring the radio frequency spectrum," *Linux journal*, vol. 2004, no. 122, p. 4, 2004.

[4] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer, "A GNU Radio-based IEEE 802.15.4 Testbed," in *12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013)*, Cottbus, Germany, sep 2013, pp. 37–40.

[5] Ramsey Electronics, "STE3500," 2019. [Online]. Available: http://www.ramseyelectronics.com/product.php?pid=14

[6] SecDev, "Scapy. packet crafting for python2 and python3." [Online]. Available: https://github.com/secdev/scapy

[7] J. Hobson, "Repurposing iot lightbulb chip for anything," 2015. [Online]. Available: https://hackaday.com/2015/02/15/repurposing-iot-lightbulb-chip-for-anything/

[8] L. Xin, J. K. Becker, S. Gvozdenovic, and D. Starobinski, "Benchmarking the physical layer of wireless cards using software-defined radios," in *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2019, pp. 271–278.

[9] J. K. Becker, S. Gvozdenovic, L. Xin, and D. Starobinski, "Testing and Fingerprinting the Physical Layer of Wireless Cards with Software-Defined Radios," *Computer Communications*, 2020, accepted.

[10] J. Pohl and A. Noack, "Universal Radio Hacker: A Suite for Analyzing and Attacking Stateful Wireless Protocols," in *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. Baltimore, MD: USENIX Association, 2018. [Online]. Available: https://www.usenix.org/conference/woot18/presentation/pohl

[11] R. Speers, T. Goodspeed, and I. R. Jenkins, "Fingerprinting IEEE 802.15.4 Devices with Commodity Radios," pp. 1–7, 2014.

[12] M. Knight, "Designing RF Fuzzing Tools to Expose PHY Layer Vulnerabilities," in *GNU Radio Conference 2018*, 2018.

[13] J. Mikulskis, J. K. Becker, S. Gvozdenovic, and D. Starobinski, "Snout: An Extensible IoT Pen-Testing Tool," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, nov 2019, pp. 2529–2531. [Online]. Available: https://dl.acm.org/doi/10.1145/3319535.3363248