# IoT-Scan: Network Reconnaissance for the Internet of Things

Stefan Gvozdenovic*, Johannes K Becker†, John Mikulskis‡ and David Starobinski§

***Abstract*—The rapid growth of the IoT has resulted in an array of competing, largely incompatible wireless communication technologies. This plethora of technologies has resulted in a complex landscape, notably a lack of visibility, making it difficult for organizations to come up with appropriate policies and tools to secure their operational environments. In this paper, we present `IoT-Scan`, a holistic approach for IoT network reconnaissance to enable enumeration of IoT devices in one's organization. `IoT-Scan` is based on software defined radio (SDR) technology, which allows for a flexible software-based implementation of radio protocols. We present a series of passive, active, multi-channel, and multi-protocol scanning algorithms to speed up the discovery of devices with `IoT-Scan`. We benchmark the passive scanning algorithms against a theoretical traffic model based on the non-uniform coupon collector problem. We implement the scanning algorithms for four popular IoT protocols: Zigbee, Bluetooth LE, Z-Wave, LoRa. Through extensive experiments with dozens of IoT devices, we evaluate and compare the performance of the various algorithms in terms of their discovery time, packet loss, and energy consumption. Notably, using multi-protocol scanning, we demonstrate a reduction of 70% in the discovery times of Bluetooth and Zigbee devices in the 2.4GHz band and of LoRa and Z-Wave devices in the 900MHz band, compared to sequential passive scanning.**

***Index Terms*—Device management, naming and addressing, service middleware and platform.**

## I. Introduction

The Internet of Things (IoT) device market is currently exhibiting exponential growth [1]. These devices run a variety of low-power wireless communication protocols, such as Bluetooth Low Energy (BLE) [2], Zigbee [3], Z-Wave [4], and LoRa [5]. which support applications in smart homes, assisted living, smart grid, health care, and environmental monitoring. These various IoT protocols grew organically, with little coordination between communication standard bodies. Thus, each protocol has its own network interface card (NIC) implementations with associated software stack tools.

The heterogeneity of the IoT ecosystem represents a major challenge from a network security monitoring perspective [6], [7]. A recent report by Palo Alto Networks concluded "the first thing businesses need to do is get visibility into the exact number and types of devices on their networks, keeping a detailed, up-to-date inventory of all connected IoT assets, their risk profiles, and their trusted behaviors" [8, p. 5].

*Reconnaissance* is a critical process toward improving the security of IoT networks [9], [10]. It entails the ability to enumerate all the IoT devices in one's organization, verify their authenticity, and assess their potential vulnerabilities to known attacks. In the context of the Internet of Things, network reconnaissance is challenging for several reasons:

- **IoT fragmentation.** The IoT ecosystem is fragmented into a multitude of competing, non-interoperable standards and platforms. Even devices operating on the same protocol may be incompatible if they run different versions of the protocol (e.g., normal versus long-range Z-Wave [11]). Using dozens of different USB dongles or network cards for each protocol is prohibitive for practical network security auditing.
- **Lack of IP addressing.** Many popular IoT protocols, including BLE, Zigbee, Z-Wave, and LoRa do not support IP addressing. As such, traditional network reconnaissance tools that operate on top of the IP protocol stack, such as Nmap [12], are fundamentally limited when it comes to IoT devices. Proposed solutions to standardize communications, such as Thread [13] and Matter [14], are promising, but touch only a fraction of technologies in use today and partly support backward-compatibility with legacy devices.
- **Physical-layer constraints.** In many IoT devices, PHY-layer parameters are hard coded on the network card. For instance, in Z-Wave and LoRa, networks identifiers are hard coded; devices that wish to communicate must use the same network identifier. These constraints significantly complicate the detection of IoT devices and analysis of their traffic.

To address this current challenge, we propose `IoT-Scan`, an extensible, multi-protocol IoT network reconnaissance tool for enumerating IoT devices. `IoT-Scan` runs both on the 900 MHz and 2.4 GHz bands and currently supports four popular IoT protocols: Zigbee, BLE, LoRa, and Z-Wave. Remarkably, `IoT-Scan` runs on a single piece of hardware, namely a software-defined radio (SDR) [15]. The small form factor of the SDR simplifies portability.

`IoT-Scan` leverages software-defined implementation of IoT communication protocol stacks, mostly under the GNU Radio ecosystem [16], [17]. This approach reduces the amount of hardware needed to address the growing number of IoT protocols. This further allows for future expansion into new protocol versions, thus eliminating the need for purchasing or upgrading protocol-specific hardware [18].

A key challenge faced in the design of `IoT-Scan` lies in minimizing the discovery time of devices. A simple approach (which we refer to as a *sequential* algorithm) is to scan devices

*†‡§Department of Electrical and Computer Engineering, Boston University Boston, US
*tesla@bu.edu, †jkbecker@bu.edu, ‡jkulskis@bu.edu, §staro@bu.edu

in a round robin fashion across each individual protocol, and in turn across each individual channel within each protocol. However, this approach does not scale. Consider, for instance, that Zigbee devices can communicate over 16 different channels.

To address the above, we propose, implement, and benchmark several scanning algorithms to speed up the discovery of IoT devices. These algorithms, of increasing sophistication, can listen in parallel *across different channels and different protocols*. To achieve this, our work takes on the challenge of integrating single-protocol software radio receiver implementations into a hybrid receiver which can switch between protocols (scanning one at a time) or may receive several protocols in parallel. The parallel scanning across channels and protocols depends mainly on the constraints of limited instantaneous bandwidth (i.e., the range of frequencies to which the SDR is tuned at a given point in time). Indeed, the channel spread defined by most protocols operating in the 2.4 GHz band is wider than the typical instantaneous bandwidth of an SDR, i.e., it is typically not possible to monitor the entire spectrum of a protocol simultaneously with one monitoring device.

Another challenge is that some IoT devices transmit sparingly, and enumerating devices passively on the wireless channel can result in long discovery times. To speed up discovery of such devices, we propose *active scanning algorithms* that send probe messages to discover which channels are actively used by devices of a given protocol, and skip channels on which no communication is taking place. We implement and evaluate the performance of these algorithms, both in terms of discovery time and energy consumption, for Zigbee devices.

Another important consideration is evaluating the efficiency of the scanning algorithm implementation on the SDR, namely whether devices are indeed discovered as fast as possible and no packet loss is incurred due to imperfect SDR implementation. We achieve this by establishing a connection between our network scanning problem and the non-uniform coupon collector problem [19], [20], whereby each transmission by a specific device corresponds to a coupon of a certain type and the objective is to collect a coupon of each type as fast as possible. The non-uniformity of the problem stems from the different rates at which different devices transmit packets. Under appropriate statistical assumptions, we can analyze this problem and numerically compute the expectation of the order statistics of the discovery times (i.e., the average time to discover $n$ out of $N$ devices, for any $n = 1, 2, \ldots, N$). For the cases of Zigbee and BLE, we show that the discovery times, as measured through several repeated experiments, closely align with these theoretical benchmarks.

Our main contributions can thus be summed up as follows:

- We introduce `IoT-Scan`, a universal tool for IoT network reconnaissance. `IoT-Scan` consists both of a collection of efficient and practical IoT scanning algorithms and of their implementations using a single commercial off-the-shelf software-defined radio device, namely a USRP B200 SDR [21].
- We validate the performance of the algorithms through extensive experiments on a large collection of devices. We demonstrate multi-protocol, multi-channel scanning both on the 2.4 GHz band for Zigbee and BLE, and on the 900 MHz band for LoRa and Z-Wave. Our implementation allows to promiscuously listen to network traffic, even when the network ID is encoded at the PHY layer.
- We propose new active scanning algorithms and show an implementation for Zigbee, which cuts down the discovery time by 87% (from 365 seconds to 46 seconds) compared to a sequential passive scanning algorithm. We further demonstrate a similar performance gain in terms of energy savings (from 770 J with passive scanning to 98 J with active scanning).
- We evaluate the efficiency of the scanning algorithm implementation on the SDR through a theoretical benchmark based on the non-uniform coupon collector problem [19], [20]. We show that passive scan algorithms for Zigbee and BLE perform near that benchmark.

**Threat model.** The purpose of `IoT-Scan` is to enumerate IoT devices and their properties at a given location (e.g., an office, a hospital room, etc.). This can be used to detect hidden unauthorized devices, some of which may have been intentionally planted by an adversary for malicious purposes (e.g., eavesdropping). We assume that these devices transmit packets, such as beacons, and/or respond to queries according to their respective wireless protocols. Note that it is hard to detect devices that do not transmit at all. `IoT-Scan` can also be used to identify missing devices which may have been deactivated or stolen by a malicious party (these devices would appear in scans up to some point, but disappear afterward).

The rest of this paper is structured as follows. Section II discusses related work. Section III presents the scanning methods and algorithms forming the core of `IoT-Scan`. Section IV discusses performance metrics for the algorithms, as well as a theoretical model for benchmarking device discovery. Section V provides background on each of the IoT protocols covered in this paper, and elaborates on how `IoT-Scan` discovers the addresses of devices in each case. Section VI presents our experiments, including implementation aspects, experimental setup, and results. Section VII provides an additional in-depth study of the active scan performance of Zigbee, including a discussion with respect to energy consumption. Section VIII concludes our findings, discusses ethical issues, and presents an outlook on future work.

An earlier and abbreviated version of this paper appeared in [22]. The main differences between the journal version and the conference version are the following: (i) we rewrote Section III, providing more detail on each algorithm, including providing pseudo-code for supporting functions (cf. Algorithms 1, 3, 5, and 6) and for an additional scanning algorithm, namely active multiprotocol scan (cf. Algorithm 8); (ii) we rewrote Section V to explain in much more detail how `IoT-Scan` discovers device addresses for each of the IoT protocols covered in this paper; (iii) we expanded Section VI to discuss implementation challenges and parameter optimizations, including experimental evaluation of different channel dwell times (cf. Figure 11); (iv) we added an entirely new section, Section VII, which provides additional experimental data for Zigbee scanning, including per-device scan

times, likelihood of discovering short versus long addresses; and energy consumption comparison of active versus passive scans; (v) we added discussion of future work and inherent constraints of IoT-Scan in Section VIII-A; (vi) we updated the abstract, introduction, conclusion and bibliography.

## II. Related Work

This section presents related work. Most existing work focuses on *protocol-specific* techniques. In contrast our work introduces several *cross-protocol* algorithms for IoT scanning, and further benchmarks their performane both theoretically and experimentally.

Heinrich et al. presents BTLEmap [23], a BLE-focused device enumeration and service discovery tool inspired by traditional network scanning tools like Nmap [12]. While BTLEMap supports both Apple's Core Bluetooth protocol stack and external scanner sources, it is limited to Bluetooth LE by design and does not aim to support multiple protocols. In contrast, `IoT-Scan` is not tied to a particular vendor as a host device, and supports multiple protocols simultaneously, with one radio source.

Sharma et al. propose Lumos [24], a system that identifies and further localizes hidden devices, using commodity hardware (e.g., a MacBook or an iPhone). Lumos is currently limited to Wi-Fi devices and does not support other IoT protocols.

Tournier et al. propose IoTMap [25], which models interconnected IoT networks using various protocols, and deduces network characteristics on multiple layers of the respective protocol stacks. IoTMap requires dedicated radios for each protocol in order to operate, whereas `IoT-Scan` achieves device detection across multiple protocols with a single software-defined radio transceiver.

Mikulskis et al. present Snout [26] and showcase scanning of BLE and Zigbee devices under a common SDR platform. `IoT-Scan` encompasses additional protocols, namely LoRa and Z-Wave. Furthermore, our work introduces novel scanning algorithms and conducts extensive evaluation of these algorithms, both theoretically and empirically with dozens of IoT devices. In contrast, the work in [26] does not present scanning algorithms and has no evaluation contents (either theoretical or empirical).

Bak et al. [27] optimize BLE advertising scan (i.e., device discovery) by using three identical BLE dongles. This approach is not scalable since it requires a new hardware receiver for each new channel, and equally does not scale beyond the BLE protocol. In contrast, our SDR-based approach uses the same SDR hardware to receive multiple protocols.

Kilgour [28] presents a multi-channel BLE capture and analysis tool implemented on a field programmable gate array (FPGA). This multi-channel BLE tool allows receiving data from multiple channels in parallel. However, the focus is on BLE PHY receiver implementation and related signal processing rather than actual scanning and enumeration of devices. In contrast, our work extends beyond Bluetooth LE, and crucially performs practical device enumeration scans to quantify scanning performance.

Park et al. describe a Wi-Fi active scan technique performed using BLE radio using cross-protocol interference [29]. The active scan algorithms in `IoT-Scan` are motivated by similar ideas, but require judicious use of protocol-specific mechanisms (i.e., sending beacon request packets in Zigbee).

Hall et al. [30] describe a tool, called EZ-Wave that can discover Z-Wave devices passively and actively. The EZ-Wave tool actively scans a Z-Wave device by sending a "probe" packet with acknowledgement request flag set. In the older version S0 of the Z-Wave protocol, it was compulsory for a Z-Wave device to reply with acknowledgements to such packets. By getting this acknowledgement back, the EZ-Wave tool learns about a device's presence. However, the EZ-Wave tool only supports older versions of Z-Wave protocol. In the new version (S2) of the Z-Wave protocol, acknowledgements are not compulsory and this is not a reliable active scan mechanism. The old Z-Wave protocol uses only the R1 (9.6 kbps) and R2 (40 kbps) physical layers. Our work adds R3 (100 kbps PHY) as well as multi-protocol capabilities. The R1, R2, and R3 rates are defined in [4, Table 7-2].

Choong [31] implements a multi-channel IEEE 802.15.4 receiver using a USRP2 software-defined radio. Choong describes a channelization method similar to the receive chain used in this work (see Section VI) that extracts multiple channels from a wider raw signal stream. However, Choong's work focuses on the performance impact of the SDR host computer, and is a Zigbee-specific implementation, whereas our work focuses on device enumeration in a multi-channel as well as multi-protocol context.

Our Zigbee, BLE, and Z-Wave GNU Radio receiver implementations are based on scapy-radio [17] flowgraphs. Our LoRa GNU Radio receiver flowgraph is based on a work by Tapparel et al. [5]. A similar multi-channel LoRa receiver was implemented by Robyns in [32]. In order to support multi-radio, multi-channel capabilities, `IoT-Scan` implements several changes to these GNU Radio receiver implementations. In general, these changes pertain to the signal path between the SDR source and the receive chains for individual channels and protocols (i.e., frequency translation, filtering, and resampling, see Section VI-A). Additionally, our LoRa receiver can listen to LoRa packets promiscuously.

We summarize the capabilities of existing tools and their limitations associating them with different wireless protocols they support, as shown in Table I. Related works supporting Wi-Fi are not listed as Wi-Fi devices generally have scannable IP addresses in contrast to the other IoT protocols listed in the table.

## III. Scanning Algorithms

In this section, we introduce SDR-based scanning algorithms that form the core of `IoT-Scan`. The notion of *channel* in this section refers to a 3-tuple containing the center frequency of the channel, the channel bandwidth (i.e., a range of frequencies delineated by the lower and upper frequencies of the channel), and the protocol type. The concept of *instantaneous bandwidth* refers to the range of frequencies captured by the SDR at any given point of time. The *center frequency* corresponds to the frequency at the middle of the range.

Table I: Related Works and Supported Protocols

| Research Work | Zigbee | BLE | Z-Wave | LoRa |
|---|---|---|---|---|
| BTLEMap [23] | | ✓ | | |
| Snout [26] | ✓ | ✓ | | |
| EZ-Wave [30] | | | ✓ | |
| Bak [27] | | ✓ | | |
| Kilgour [28] | | ✓ | | |
| Choong [31] | ✓ | | | |
| Tapparel [5] | | | | ✓ |
| Robyns [32] | | | | ✓ |
| IoT-Scan [22] | ✓ | ✓ | ✓ | ✓ |

**Algorithm 1:** Listen($ch, dwell\_time$)

▷ Receive packets on channel *ch* for duration *dwell_time* and return a list of discovered devices

1 $t_{start} \leftarrow$time() ▷ Store current time
2 $device\_list \leftarrow \{\}$ ▷ Initialize device list
3 **while** time()$-t_{start} \leq dwell\_time$ **do**
4  Listen on channel $ch$
5  Get packet and extract address $dev\_addr$
6  $device\_list = device\_list \cup dev\_addr$
7 **end while**
8 **return** $device\_list$

### A. Single-channel methods

The key building block to any of the following scanning algorithms is the function **Listen()** (Algorithm 1). It takes two input parameters, namely a channel $ch$ (defined by a center frequency, bandwidth and protocol) and a time period $dwell\_time$ after which the procedure terminates listening to channel $ch$. During execution of this procedure, the SDR decodes any packet received on the channel, and extracts address information $dev\_addr$ that identifies a device (line 5). Note that some packets (e.g., acknowledgements in Zigbee) may have no address information, in which case $dev\_addr$ is an empty set. Next, the device address is added to the list of discovered devices $device\_list$ (line 6). By definition, if $dev\_addr$ already appears in $device\_list$, then the union operation does not change the contents of the list. Upon the expiration of the channel dwelling time, the procedure returns the list of discovered devices.

Algorithm 2 presents a simple sequential scanning procedure **Passive_Scan** that can be used in conjunction with any IoT protocol. This algorithm represents a baseline against which the performance of more advanced algorithms can be compared. The algorithm invokes the **Listen** procedure in a round-robin fashion on each channel of a given channel list $ch\_list$, which is provided as an input to the procedure. The total scan time is set by the $scan\_time$ input parameter. Note that generally $scan\_time \gg dwell\_time$, and hence each channel is visited several times during the scan. The algorithm returns the list of discovered devices.

Sequential passive scanning can be slow, especially if an IoT protocol supports many channels, but only a few channels are used. In order to speed up device discovery, Algorithm 4, which we refer to as **Active_Scan**, implements a two-phase

**Algorithm 2:** Passive_Scan($ch\_list$, $dwell\_time$, $scan\_time$)

▷ Enumerate devices by repeatedly listening for duration *dwell_time* on each channel in *ch_list* and stop after *scan_time*

1 $t_{start} \leftarrow$time() ▷ Store current time
2 $device\_list \leftarrow \{\}$ ▷ Initialize device list
3 $i \leftarrow 0$ ▷ Set channel counter to zero
4 **while** time()$-t_{start} \leq scan\_time$ **do**
  ▷ *ch_list*($i$) is the $i$-th element in *ch_list*
5  $new\_dev \leftarrow$ **Listen**($ch\_list(i), dwell\_time$)
6  $device\_list = device\_list \cup new\_dev$
7  $i \leftarrow (i+1) \bmod |ch\_list|$
8 **end while**
9 **return** $device\_list$

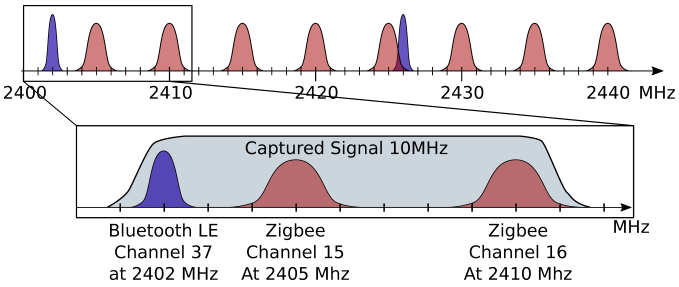


Figure 1: **Find_Channels_In_Range** (Algorithm 5) starts at the lowest channel in the provided list and returns all channels that are in range of the SDR hardware based on the provided instantaneous bandwidth parameter.

approach. During the first phase (line 4), it invokes a helper function **Probe_Channels** (Algorithm 3), which sends a probe packet on each channel $ch$ in the provided $channel\_list$ and waits for a response. If one or more devices respond, then channel $ch$ is added to the $active\_channels$ list. During the second phase (lines 5–7), Algorithm 4 performs passive scanning only on channels appearing in the $active\_channels$ list for the remaining scan time. Algorithm 4 is especially useful for protocols such as Zigbee, which defines 16 different channels, not all of which may be in use. For Zigbee, `IoT-Scan` implements the probe packet using a *beacon request* frame, to which Zigbee devices respond with a *beacon* frame (see also Section V).

### B. Multi-channel methods

The subsequent algorithms expand from single channel scanning to handling multiple channels and multiple protocols, at the same time.

Prior to discussing multi-channel and multi-protocol scanning algorithms, we need a method of grouping channels within the range of the instantaneous bandwidth of the SDR. The function **Find_Channels_In_Range** (Algorithm 5) identifies all channels in an input channel list (ordered by ascending frequency) by selecting all channels that fit the instantaneous bandwidth under consideration of their respective center frequencies and channel bandwidths, see Fig. 1. Note that while some channels overlap, transmissions on these channels do not

**Algorithm 3:** Probe_Channels($ch\_list, dwell\_time$)

▷ Actively probe each channel *ch* from the channel list *ch_list* for duration of *dwell_time*.

$active\_channels \leftarrow \{\}$
$device\_list \leftarrow \{\}$
**for** $ch \in ch\_list$ **do**
  Send probe on channel $ch$ ▷ Trigger responses
  $new\_dev \leftarrow$ **Listen**$(ch, dwell\_time)$
  **if** $new\_dev \neq \{\}$ **then**
    $device\_list \leftarrow device\_list \cup new\_dev$ ▷ Add found devices
    $active\_channels \leftarrow active\_channels \cup ch$ ▷ Add channel to active channel list
  **end if**
**end for**
**return** $active\_channels, device\_list$

**Algorithm 4:** Active_Scan($ch\_list, dwell\_time, scan\_time$)

▷ Enumerate devices by first identifying the list of *active_channels* in *ch_list* and then performing passive scanning only on those *active_channels*

$device\_list \leftarrow \{\}$ ▷ Initialize list of found devices
$active\_channels \leftarrow \{\}$ ▷ Initialize list of busy channels
$t_{start} \leftarrow$time() ▷ store current time
▷ Phase 1: Scan active devices
$active\_channels, device\_list \leftarrow$ **Probe_Channels**$(ch\_list, dwell\_time)$
▷ Phase 2: Passive-scan known active channels for the remaining time
$t_{scan} \leftarrow scan\_time - (\text{time}() - t_{start})$ ▷ Compute remaining scanning time
$new\_dev \leftarrow$ **Passive_Scan**$(active\_channels, dwell\_time, t_{scan})$ ▷ Run passive scan on active channels
$device\_list \leftarrow device\_list \cup new\_dev$ ▷ Add devices found during passive scanning
**return** $device\_list$

**Algorithm 5:** Find_Channels_In_Range($ch\_list, bandwidth$)

▷ Identify all channels in *ch_list* (ordered by ascending frequency) that can fit in the instantaneous *bandwidth*, starting from the first element *ch_list*(0).

$ch\_range \leftarrow \{\text{all } ch \text{ in } ch\_list \text{ such that } (ch.{}_{freq} + ch.{}_{bw}/2) - (ch\_list(0).{}_{freq} - ch\_list(0).{}_{bw}/2) \leq bandwidth\}$
**return** $ch\_range$

occur continuously. Hence, it is possible to decode packets if they do not collide, which is usually the case. For a channel to be considered in range, the entire bandwidth of the signal must be contained in the captured instantaneous bandwidth of the SDR. In practice, the center frequency of each group of channels is set such that the first channel from the channel list (i.e., the one with the lowest frequency) is at the far left end of the instantaneous bandwidth. If none of the other channels' bandwidths overlap with the current instantaneous bandwidth, the function will return the first element of the input channel list, i.e., it will default to single-channel selection.

We further define a helper function **Listen_In_Parallel** (Algorithm 6) which simultaneously listens to multiple channels by calling **Listen** (Algorithm 1) on all provided channels. Note that implementing this algorithm requires extracting multiple signal streams by frequency-shifting, filtering, and resampling the incoming signal relative to its center frequency and the parameters of the channel. This procedure is called *channelization*. The implementation aspects of this procedure are described in Section VI-A.

**Multiprotocol_Scan** (Algorithm 7) describes a parallel multi-protocol scan that can be used with any number of IoT protocols. Based on a list of channels to consider (ordered by ascending frequencies), the algorithm starts at the lowest frequency and determines all channels within range of the first channel by calling **Find_Channels_In_Range** (Algorithm 5). It subsequently listens to those channels by invoking **Listen_In_Parallel** (Algorithm 6). Note that if only one channel is in range at given step of the while loop (line 12), then the algorithm's behavior becomes identical to **Passive_Scan** (Algorithm 2). Each such channel hop is scanned for the defined channel $dwell\_time$. Once all unscanned channels are exhausted, the algorithm restarts from the lowest channel until the desired $scan\_time$ has elapsed. Note that the total $scan\_time$ is typically much longer than the channel $dwell\_time$. Depending on the frequency allocation of the protocols involved, the multi-protocol scan algorithm can significantly speed up IoT device discovery process by receiving multiple protocols simultaneously, as demonstrated in Section VI-C.

**Algorithm 6:** Listen_In_Parallel($ch\_range, dwell\_time$)

▷ Scan in parallel all channels in *ch_range* for a duration *dwell_time*. Note that all channels are assumed to be within the instantaneous bandwidth of the SDR, e.g. as produced by Find_Channels_In_Range()

**do in parallel**
  $new\_dev \leftarrow$ {**Listen**$(ch, dwell\_time)$ for all $ch$ in $ch\_range$}
  $device\_list \leftarrow device\_list \cup new\_dev$
**end parallel**
**return** $device\_list$

**Algorithm 7:** Multiprotocol_Scan($ch\_list, dwell\_time, scan\_time, bandwidth$)

▷ This algorithm enumerates devices by scanning as many channels as can fit in the instantaneous bandwidth of *bandwidth* for a duration *dwell_time* in each iteration.

$ch\_unscanned \leftarrow ch\_list$ ▷ All channels in the list are unscanned
$ch\_groups \leftarrow \{\}$ ▷ Initialize list of channel groups
**while** $ch\_unscanned \neq \{\}$ **do**
  ▷ Find channels that can be scanned simultaneously as they fit the instantaneous *bandwidth*.
  $ch\_range \leftarrow$ **Find_Channels_In_Range**(ch_unscanned, bandwidth)
  ▷ Scan channels that fit in instantaneous bandwidth BW around center freq.
  $ch\_groups \leftarrow ch\_groups \cup \{ch\_range\}$ ▷ Add this group to the list of channel groups
  $ch\_unscanned \leftarrow ch\_unscanned \setminus ch\_range$ ▷ Remove channels from unscanned list
**end while**
$t_{start} \leftarrow$time() ▷ Store current time
$device\_list \leftarrow \{\}$ ▷ Initialize list of found devices
$i \leftarrow 0$ ▷ Set channel counter to zero
**while** time()$-t_{start} \leq scan\_time$ **do**
  ▷ Scan all channels of the i'th channel group in parallel
  $new\_dev \leftarrow$ **Listen_In_Parallel**(ch_group(i), dwell_time)
  ▷ Add newly found device(s) to the device list
  $device\_list \leftarrow device\_list \cup new\_dev$
  $i \leftarrow (i+1)$ mod $|ch\_groups|$
**end while**
**return** $device\_list$

**Algorithm 8:** Active_Multiprotocol_Scan($ch\_list, ch\_probe\_list, dwell\_time, scan\_time, bandwidth$)

▷ Enumerate devices by first identifying list of *busy_channels* from *ch_probe_list* and then performing multi-protocol scanning only on those active channels and on other channels provided in *ch_list*.

$device\_list \leftarrow \{\}$ ▷ Initialize list of found devices
$active\_channels = \{\}$ ▷ Initialize list of busy channels
$t_{start} \leftarrow$ time() ▷ Store current time
▷ Probe all channels in *ch_probe_list* to identify active channels
$active\_channels, device\_list \leftarrow$ **Probe_Channels**$(ch\_probe\_list, dwell\_time)$
$t_{scan} \leftarrow scan_time - ($time$() - t_{start})$ ▷ Compute remaining scanning time
▷ Merge successfully probed channels with regular channels and sort by ascending frequency
$active\_channels \leftarrow$ sort$(active_channels \cup ch_list)$
▷ Run passive scan on all active channels for the remaining time $t_{scan}$
$new\_dev \leftarrow$ **Multiprotocol_Scan**$(active\_channels, dwell\_time, t_{scan}, bandwidth)$
$device\_list \leftarrow device\_list \cup new\_dev$ ▷ Add devices found during passive scanning
**return** $device\_list$

Finally, **Active_Multiprotocol_Scan()** (Algorithm 8) is a combination of the aforementioned active scanning and multi-protocol scanning capabilities. It is useful for scanning multiple protocols, some actively and some passively (such as a combination of Zigbee and BLE). Note that Algorithm 8 receives two lists of channels: $ch\_probe\_list$ and $ch\_list$. Active channels (e.g., Zigbee channels) are only sought among channels in the $ch\_probe\_list$. This step is skipped for channels (e.g., BLE channels) in the $ch\_list$.

## IV. Performance Metrics and Analysis

In this section, we define metrics to benchmark the various algorithms. We further formalize IoT device discovery as a variation of the non-uniform (weighted) coupon collector problem [19], [20]. Under appropriate statistical assumptions, the coupon collection time can be computed numerically and serve as a baseline against which the performance of the algorithms can be compared.

### A. Metrics

Our main metric is the discovery time of IoT devices, which we aim to minimize. Assume there are $N$ devices in total, with corresponding discovery times $T_1, T_2, \ldots, T_N$. We are interested in characterizing the *order statistics* of these random variables, i.e., the time elapsing till one device is discovered, which is denoted $X_{1:N}$, then till two devices are discovered which is denoted $X_{2:N}$, and so on till all devices are discovered, which is denoted $X_{N:N}$. We thus have

$$\begin{aligned} X_{1:N} &= \min(T_1, T_2, \ldots, T_N), && (1)\\ X_{2:N} &= \min(\{T_1, T_2, \ldots, T_N\} \setminus X_{1:N}), && (2)\\ &\ldots \\ X_{N:N} &= \max(T_1, T_2, \ldots, T_N). && (3) \end{aligned}$$

In our experiments, we estimate the expectation of the $n$-th order statistics $E[X_{n:N}]$, for $n = 1, 2, \ldots, N$. To obtain these estimates, we run each scanning algorithm $M$ times and denote by $x_{n:N}^{(m)}$ the time till $n$ devices are discovered at the $m$-th iteration, where $m = 1, 2, \ldots, M$. We then compute the *sample mean* for the $n$-th order statistics as follows:

$$\bar{x}_{n:N} = \frac{\sum_{m=1}^{M} x_{n:N}^{(m)}}{M}. \quad (4)$$

We also provide $(1-\alpha)100\%$ *confidence intervals* for our estimates

$$[\bar{x}_{n:N} - e_{n:N}, \bar{x}_{n:N} + e_{n:N}], \quad (5)$$

based on computing the sample standard deviation $s_{n:N}$ and the confidence interval parameter $e_{n:N}$ as follows:

$$\begin{aligned} s_{n:N} &= \sqrt{\frac{1}{M-1}\sum_{m=1}^{M}(x_{n:N}^{(m)} - \bar{x}_{n:N})^2}, && (6)\\ e_{n:N} &= t_{\alpha/2,M-1} \times \frac{s_{n:N}}{\sqrt{M}}, && (7) \end{aligned}$$

with $t_{\alpha/2,M-1}$ denoting the $1-\alpha/2$ quantile of the $t$-distribution with $M-1$ degrees of freedom [33]. In our experiments, described in Section VI, we run $M = 10$ independent iterations for each algorithm and consider 95% confidence intervals (i.e., $\alpha = 0.05$), hence $t_{\alpha/2,M-1} = 2.262$. [33, Table 1].

### B. Theoretical Model

We next propose a theoretical model to estimate the expectations of order statistics of the discovery time, under appropriate statistical assumption. The analysis further assumes an idealized channel environment where no packet loss occurs (in practice such losses could occur due to imperfect receiver implementation or interference). In Section VI-C, we show that the performance of the scanning algorithms approaches that predicted by the theoretical model, which demonstrates the efficiency of the algorithms.

*1) Statistical assumptions:* To model device enumeration, we need statistics of the inter-arrival times of packets generated by each device. For the sake of *analytical tractability*, we assume that devices transmit in a memoryless fashion, i.e., the inter-arrival times of their packets follow an exponential distribution. Note that the mean and standard deviation of an exponential random variable are equal. Hence, we expect that this model can provide a reasonable approximation, if for each device $i$, its mean inter-arrival time $\mu_i$ and standard deviation of inter-arrival times $\sigma_i$ are roughly equal. We stress that this assumption is not needed for the implementation of the scanning algorithms, only for their analysis.

To check this assumption, we collected statistics of the inter-arrival times of packets of the Bluetooth and Zigbee devices listed in Table II below. Specifically, for each device $i$, we measure the times of packet arrivals with $K+1$ timestamps. We then calculate the $K$ inter-arrival times $\tau_{i,k} = t_{i,k+1} - t_{i,k}$, where $k = 1, 2, \ldots, K$. Based on this data, we obtain estimates of the expectation for each device $i$

$$\mu_i = \frac{1}{K}\sum_{k=1}^{K}\tau_{i,k}, \quad (8)$$

as well as the standard deviation

$$\sigma_i = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(\tau_{i,k} - \mu_i)^2}. \quad (9)$$

Table II indicates that indeed for all tested BLE devices and most Zigbee devices $\mu_i \approx \sigma_i$.

*2) Analysis of order statistics:* Enumerating devices shares similarities with the non-uniform coupon collector's problem [19], albeit with certain modifications. The coupon collector's problem assumes a probability distribution in which each draw results in a coupon (i.e., a discovered device). This cannot be applied directly to a scenario in which devices' transmission characteristics may result in *null coupons*, i.e., a scan iteration in which no new device is discovered. Anceaume et al. [20] provide a method of calculating the expectation of the non-uniform coupon collector problem which accounts for a null coupon. Define the probability vector $\boldsymbol{p}$ in which $p_0$ is the probability of no device transmitting, and $p_i$ is the probability of device $i$ transmitting, $i = 1, 2, \ldots, N$. The

expectation for the $n$-th order statistics $X_{n,N}$ (i.e., the time to to discover $n$ out of $N$ devices) is then given by

$$\mathbb{E}[X_{n:N}(\boldsymbol{p})] = \sum_{h=0}^{n-1} R_{N,n,h} \sum_{J \in S_{h,N}} \frac{1}{1 - p_0 - P_J} \quad (10)$$

where

$$R_{N,n,h} = (-1)^{n-1-h} \binom{N-h-1}{N-n}. \quad (11)$$

Here, $S_{h,N}$ denotes all $\binom{N}{h}$ subsets containing exactly $h$ devices. Denote by $J$ any subset of $S_{h,N}$ that contains exactly $h$ devices. Then, $P_J = \sum_{j \in J} p_j$ is the summation of the transmission probabilities of all devices belonging to $J$. Note that the second summation term in Eq. (10) works out to a summation over all possible subsets $J$ of cardinality $h$.

Assuming all $N$ devices send packets in an independent and identically distributed memoryless fashion as discussed above, the device traffic can be modeled as $N$ independent Poisson processes with rate $\lambda_i = 1/\mu_i$. The combined influx of packets from all the devices then follows a Poisson process with rate $\lambda = \sum_{i=1}^{N} \lambda_i$. By selecting a small interval $\Delta t$ such that either zero or one packet arrives during any interval $\Delta t$, we can use Eq. (10) to compute the expectation of the order statistics of the discovery time of devices. Let $Z$ be a Poisson random variable with mean $\lambda \Delta t$ that counts the number of packets arriving from all devices during a time interval $\Delta t$. We have

$$\Pr(Z = 0) = e^{-\lambda \Delta t}, \quad (12)$$

$$\Pr(Z = 1) = (\lambda \Delta t) e^{-\lambda \Delta t}, \quad (13)$$

$$\Pr(Z \geq 2) = 1 - Pr(Z = 0) - Pr(Z = 1). \quad (14)$$

In order to determine a suitable $\Delta t$, we select it such that $Pr(Z \geq 2)$ becomes negligible, as discussed in Section VI-C. If all devices transmit on one channel that is continuously monitored, the probability $p_i$ that device $i$ transmits during an interval $\Delta t$ is then

$$p_i = (\lambda_i \Delta t) e^{-\lambda_i \Delta t} \approx \lambda_i \Delta t. \quad (15)$$

Note that if all devices are randomly distributed on any of $C$ available channels, a randomly channel-hopping radio scanner would receive a transmission from device $i$ with probability $p_i/C$. This can also be used as an approximation when the scanner visits channels in a round-robin rather than in a random fashion.

## V. Protocol Device Enumeration

In the previous sections, the concepts of "listening to a channel" and "extracting device addresses" were presented in a generic way. We now discuss these aspects for all the IoT protocols implemented in `IoT-Scan`.

### A. Zigbee

Zigbee is a network and application layer protocol which uses the IEEE 802.15.4 physical layer specification [3]. It is widely used in home and commercial building automation applications such as lighting, climate, and access control [34]. Zigbee operates on 16 channels on the 2.4GHz ISM band.

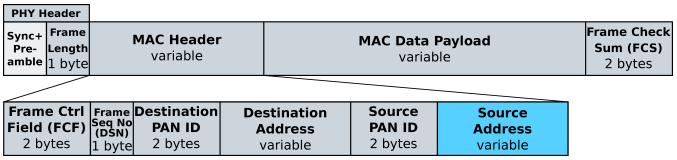


Figure 2: Zigbee medium access control (MAC) layer frame structure [34].

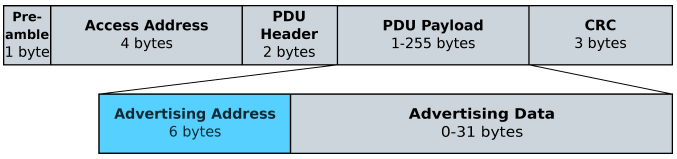


Figure 3: The BLE advertising packet format (top) and the Advertising Channel PDU Header (bottom) for common advertising messages (PDU Types ADV_IND, ADV_NONCONN_IND, ADV_SCAN_IND) [2, p. 2562, 2567].

Each channel is 2 MHz wide and centered at $f_c = 2405 + 5(k - 11)$ MHz for channels $k = 11...26$ [3, p. 387].

Zigbee defines three types of devices: *coordinators*, *routers*, and *end-devices*, each of which behave differently on the network. End-devices do not route traffic, and are typically mobile and battery-powered, i.e. energy-constrained. As a result, end devices are frequently sleeping, i.e., remain inactive in order to save power. Routers route traffic, receive and store messages for their children (i.e., end devices that they route traffic from and to), and communicate with new nodes requesting to join the network. Therefore, routers cannot sleep and are typically mains-powered devices. A Zigbee coordinator is a special router which, in addition to all of the router capabilities, also forms a network. Before creating a network, Zigbee coordinators scan available channels to select a good, i.e. low interference, channel for the network.

**Address Information.** `IoT-Scan` enumerates Zigbee devices by both their *short* (16 bits) and *long* (64 bits) address (whichever of those address types is present in a given packet), ensuring no device is counted double despite these two address formats. While short addresses are unique within a network, long addresses are globally unique and assigned by the manufacturer. Short and long addresses can be parsed from the "Source Address" variable-length field in Figure 2. Some Zigbee packets (e.g. acks) include no address. Note that the PAN ID (personal area network identifier), shown in the same figure, is a network specific identifier. While we do not use it in our scanning, it could be useful to differentiate between two devices with the same short address but on different networks.

**Implementation.** `IoT-Scan` implements both passive and active scans for Zigbee. A passive scan listens on each channel for a certain amount of time (i.e., the *channel dwell time*) repeatedly until the total scan time expires. With active scanning, channels with network activity are discovered by sending beacon requests on each channel. Receiving a beacon frame in response to a beacon request indicates that there is a network on the current channel.Subsequent passive scanning

rounds can then be limited to these active channels (line 6 in Algorithm 4), in order to detect any further devices that did not respond to active scanning.

### B. Bluetooth Low Energy (BLE)

Bluetooth LE [2] is a popular short-range wireless protocol on the 2.4GHz ISM band. Its physical layer comprises 40 channels (0-39), three of which are so-called *advertising channels* which are used to broadcast device information using *advertising packets*. Bluetooth LE operates on 40 RF (radio frequency) channels in the 2.4GHz band. Each channel is 1 MHz wide with center frequency $f_c = 2402 + 2k$ MHz where $k = 0...39$. The 40 RF channels are mapped to either data channels or advertising channels (see [35]). The advertising channels are centered at 2402 MHz (channel 37), 2426 MHz (38), and 2480 MHz (39) to ensure coexistence with other wireless protocols, i.e. minimize interfering with the most populated Wi-Fi channels 1, 6, and 11.

**Address Information.** Advertising BLE packets contain two address-related data fields in the packet structure of their most common packet types: the *access address* and the *advertising address* (Fig. 3. We use the advertising address (AdvA, 6 bytes long) to enumerate BLE devices. For advertising, the access address is set to the constant `0x8E89BED6` and is used as a sync word for frame synchronization. As it is the same for all advertisers, it cannot be used for device enumeration.

The advertising addresses of the BLE devices tested in this work did not change over time. Hence, we focused on identifying scanned devices by their advertising address. Note, however, that some devices, such as Apple devices, randomize their advertising addresses over time [35], [36]. In such cases, to infer the identity of BLE devices, one could use the data payload of BLE advertising messages, which include device identifiers, counters, or battery levels [37].

**Implementation.** In BLE, data channels are used for communication after a connection has been established, whereas advertising channels are used between devices that are in range to discover one another and exchange metadata. Therefore, `IoT-Scan` only scans the three advertising channels (i.e., there is no need to monitor data channels). Typically, advertising packets are sent on all three advertising channels for any given advertising event. This redundancy makes device discovery more resilient in cases where some of the channels experience interference. This means that scanning for BLE devices on any one of the three advertising channels is as good as a multi-channel scan (sequentially scanning each advertising channel), a fact that we also verified experimentally.

### C. LoRa

LoRa is a proprietary physical layer wireless protocol powering network layer protocols, such as LoRaWAN [38] long-range wide-area networks (LP-WAN) and Sidewalk [39]. LoRa defines all supported modulations and physical layer signaling. On the other hand, LoRaWAN defines a subset of all possible modulations and signal parameters, such as frequency allocations and channel widths. The physical layer of the LoRa protocol was patented by Semtech[1], and the specification of

[1]https://www.semtech.com/

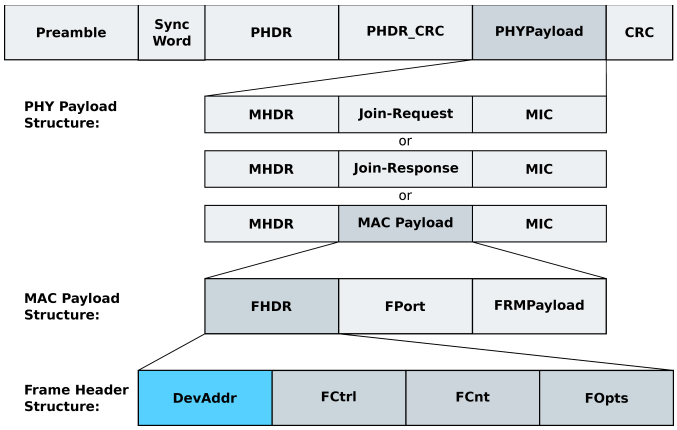


Figure 4: The LoRaWAN packet structure [38].

LoRaWAN is governed by the LoRa Alliance [38]. Work by security researchers have yielded SDR implementations of the physical LoRa layer [5], [32].

LoRaWAN uplink channels consists of 64 125 KHz wide channels (centered around $f_c = 903.2 + 0.2k$ MHz where $k = 0...63$) and 8 500 KHz wide channels (centered around $f_c = 903 + 1.6(k - 64)$ MHz where $k = 64...71$). LoRaWAN downlink channels consists of 8 500 KHz wide channels (centered around $f_c = 923.3 + 0.6k$ MHz where $k = 0...7$) [40].

**Address Information.** We use the third byte after the sync word to enumerate the LoRa devices under test. Indeed, from the traffic we collected, the value of the third byte changed between four values, corresponding to the IDs of the four YoLink devices under test. Incidentally, this third byte of the payload is part of the 32-bit device address (DevAddr) as specified in LoRaWAN frame format [38]. Furthermore, the first byte of DevAddr is used as a network identifier (NwkID), which is fixed for all devices in the same network. In this context, a network consists of a LoRa gateway and end-devices connected to that gateway.

**Implementation.** In our implementation, we scan Yolink devices listed in Table II. The major challenge in receiving any Yolink traffic is in determining the PHY-layer network sync word or network ID. We overcome the challenge of fixed sync word by modifying the LoRa receiver of [5]. Our implementation allows one to promiscuously listen for all sync words (as in [41]), as well as configure the bandwidth, the center frequency, the bit rate, and other parameters. A key advantage of scanning LoRa using an SDR implementation is that all sync words can be monitored simultaneously, whereas certified LoRa transceiver chips are programmed to receive a specific sync word.

### D. Z-Wave

Z-Wave is another proprietary physical layer wireless protocol, based on the ITU-T G.9959 specification [4]. It is used in smart home applications, most notably Ring Home Security Systems.

In the US, Z-Wave operates on the 900 MHz ISM band and comes in a few physical layer (PHY) variants, most importantly differentiated by its center frequency and bit rates

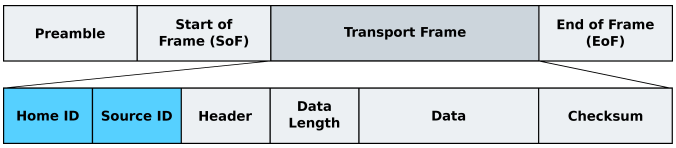


Figure 5: Z-Wave physical (PHY) and medium access control (MAC) layer frame structure [4].

(channel widths): 908.4 MHz at 9.6 Kbps (R1) and 40 Kbps (R2); 916 MHz at 100 Kbps (R3). Z-Wave long range PHYs at 912 MHz and 920 MHz are less common. For Z-Wave devices listed in Table II, `IoT-Scan` discovered traffic on the R2 and R3 PHYs, with corresponding channel widths of 40 KHz and 100 KHz respectively.

**Address Information.** Z-Wave packets contain two address-related data fields: the home (network) identifier, and the node (device) identifier. We use the single byte Source ID [42] to enumerate Z-Wave devices. Z-Wave supports up to 232 devices per network, hence this byte is sufficient to distinguish between devices on the same network. The Z-Wave primary controller (Z-Wave gateway) has a Source ID of 1. A Z-Wave device which has not been connected to a controller must use a Source ID of 0 before obtaining an actual non-zero Source ID. The Z-Wave network identifier or Home ID consists of 4 bytes that precede the node ID, as shown in Fig. 5. In the case of multiple overlapping networks, the Home ID can be used to distinguish between devices on different networks.

**Implementation.** `IoT-Scan` uses the Source ID [42] in the MAC header to enumerate Z-Wave devices. R1 and R2 Z-Wave PHY implementations are based on scapy-radio [17]. We built the R3 Z-Wave PHY receiver flowgraph based on the existing R2 PHY implementation, the main difference being the bit rate/sampling rate which is 2.5 times larger.

## VI. Experimental Evaluation

In this section, we perform an experimental evaluation of the scanning algorithms of `IoT-Scan`. We detail SDR implementation aspects, the experimental set-up (including the list of tested devices), and the experimental results.

### A. Algorithm Implementation

The main software components of our implementation consist of GNU Radio 3.8 [16] and Scapy-radio 2.4.5 [17]. Scapy-radio is a pentest tool with RF fuzzing capabilities. Note that Scapy-radio is based on GNU Radio version 3.7. We ported receiver flowgraphs to GNU Radio 3.8 gaining great reception improvements due to the automatic gain control (AGC) inside the USRP Source block.

*1) Flowgraph control:* We implement the scanning algorithms described in Section III in Python. Signal processing parameters, such as the SDR center frequency, the channel frequency offsets, and the channel bandwidths, are managed by a GNU Radio flowgraph. The GNU Radio flowgraph is imported into the main application as a Python module and is controlled with its native Python API. This allows for dynamic control of flowgraph parameters during the runtime of the flowgraph. Controlling the flowgraph in this way is crucial for correct time-keeping of the experiments, as it allows to compensate for seconds of startup delays due to the initialization of the USRP hardware driver library.

Table II: Tested IoT devices.

| (a) Zigbee Device | $\mu_i$ [s] | $\sigma_i$ [s] |
|---|---|---|
| #1 Wink hub 2 | 6.0 | 7.7 |
| #2 Amazon Echo 4.0 | 7.5 | 6.1 |
| #3 Ring Base Station | 7.9 | 7.5 |
| #4 Philips Hue Lamp1 | 8.4 | 7.1 |
| #5 Philips Hue Lamp2 | 8.3 | 7.0 |
| #6 Philips Hue Lamp3 | 8.4 | 7.0 |
| #7 Philips Hue Lamp4 | 8.4 | 7.1 |
| #8 Quirky PLINK-HUB | 8.5 | 4.7 |
| #9 Osram Lightify 73674 | 10.2 | 7.8 |
| #10 IKEA Gateway | 14.0 | 14.6 |
| #11 CREE Lightbulb A19 | 14.8 | 1.2 |
| #12 GE LAMP1 4VE8 | 14.9 | 1.4 |
| #13 GE LAMP2 4VE8 | 14.9 | 1.4 |
| #14 IKEA LED1732G11 | 15.9 | 1.2 |

| (b) BLE Device | $\mu_i$ [s] | $\sigma_i$ [s] |
|---|---|---|
| #15 Fit2 fitness tracker | 4.1 | 4.0 |
| #4 Philips Hue Lamp1 | 4.6 | 5.6 |
| #5 Philips Hue Lamp2 | 4.1 | 4.0 |
| #6 Philips Hue Lamp3 | 3.9 | 3.8 |
| #7 Philips Hue Lamp4 | 4.1 | 4.1 |
| #16 Mi Smart Band 5 | 11.0 | 10.4 |
| #17 AMIR Thermometer | 19.5 | 18.6 |
| #18 Tile tracker | 25.7 | 25.2 |
| #19 Tile tracker | 47.3 | 42.0 |
| #20 Tile tracker | 23.1 | 22.7 |
| #21 Tile tracker | 20.7 | 18.5 |
| #22 WIT Motion sensor | 119.5 | 96.1 |

| (c) LoRa (Yolink) |
|---|
| #26 Water Leak Sensor |
| #27 Door Sensor |
| #38 Smart Plug |

| (d) Z-Wave (Ring) |
|---|
| #3 Base Station |
| #23 Keypad |
| #24 Contact Sensor |
| #25 Motion Detector |

*2) Signal processing:* The process of converting an unfiltered full-bandwidth signal from an SDR source into the channel-wide receive chain (i.e., the sequence of DSP blocks connected serially starting with radio source, demodulator, filter, and clock recovery) of a particular protocol is referred to as *channelization* [43]. Channelization is particularly important in multi-protocol scanning, since it selects (filters out) a few narrow band signals (receive chains) from the raw wide band signal. Multi-protocol scans require parallel decoding of two or more receive chains which can overwhelm the capabilities of a typical host computer if the processing chain in the flowgraph is not correctly optimized.

Channelization in `IoT-Scan` comprises three signal processing steps: frequency translation (from the center frequency of the raw radio signal to the center frequency of the desired channel), channel filtering (filtering out other channels, protocols and potential interference), and re-sampling (down conversion) to reduce the computational load. Reducing the sample rate relies on Nyquist's theorem, which dictates that the sample rate of a signal be at least twice the signal's bandwidth, in order to not lose any information.

### B. Experimental Setup

We implemented all the scanning algorithms described in Section III on a single SDR device, namely a USRP B200 device [21], with a PC capable enough to handle data processing in real-time without dropping samples (i.e., overflowing buffers). Thus, all our experiments were run on a ThinkCentre 8 Core Intel i7 running Ubuntu 20.04.

The devices used in the experiments are listed in Table II. All scanning experiments were based on IoT devices under our control, which were placed on the same office desk as the SDR. Any foreign device from the environment was filtered out. In order to only account for our devices, we initially enumerated them with a passive scan inside an RF shielded box (Ramsey box STE3500) to determine their addresses.

Traffic of the BLE and Zigbee devices were statistically analyzed to derive the parameters of the theoretical traffic model introduced in Section IV-B. Note that we did not analyze transmission statistics of low-power Z-Wave and LoRa devices due to their periodic transmission patterns (devices transmit once every hour or so).

We conducted all scanning experiments using the default network configuration of the respective devices and protocols. In all experiments, the tested devices were in an idle state, i.e., not actively used by an operator. Manually operating devices in a way that generates network communication, e.g., actuating Zigbee lights via the Amazon Alexa smartphone app, would impact scanning performance. We expect the results presented in this section to be conservative estimates of the scanning time, since generating additional traffic from the devices should speed up the discovery of the devices.

Additional network configuration consisted of the following:

- The Amazon Echo 4.0 device was paired to all smart light bulbs.
- Most BLE devices in our experiments come with an associated application. Specifically, Philips Hue lights were connected to the Amazon Echo 4.0 using the Amazon Alexa app. Tile trackers, Fit2, and the Mi smart band were activated using their respective companion smartphone apps.
- The YoLink LoRa devices were left in their default configuration after pairing the end devices (moisture sensor, smart plug, door sensor) to the YoLink LoRa hub, using the YoLink app.
- The Z-wave devices (motion sensor, door sensor, keypad, and range extender) were paired to the Ring Security Base Station using the Ring app.

When `IoT-Scan` discovers a new device name, it also saves its associated frequency channel information. Devices in this study form three Zigbee networks on channels 11, 15, and 20. No device appears on more than a single channel. Regarding the parameters of the algorithms, the channel dwell time (i.e., the scanning time of each channel in each round) was set to 1 second. We also tried channel dwell times of 0.1 second and 3 seconds, and found that the scanning times did not differ significantly. The channel dwell time during active scan of Zigbee was set to 0.2 second. When scanning each individual protocol, we set the instantaneous bandwidth parameter according to the protocol's bit rate. Specifically, BLE's channel bandwidth was set to 1 MHz, Zigbee to 2 MHz, LoRa to 125 KHz, and Z-Wave to 40/100 KHz. When implementing multiprotocol scanning algorithms, we used wider bandwidth to fit the bandwidth of each protocol and channel spacing in between. Both the Zigbee/BLE and Z-Wave/LoRa and multiprotocol experiments used 8 MHz of bandwidth.

### C. Results

In this section, we discuss experimental results of the scanning algorithms. The figures show the sample means and 95% confidence intervals of the order statistics of the discovery time of the $n$-th device (see Eqs. (4) and (5)). Each point represents an average over 10 experiments with identical parameters.

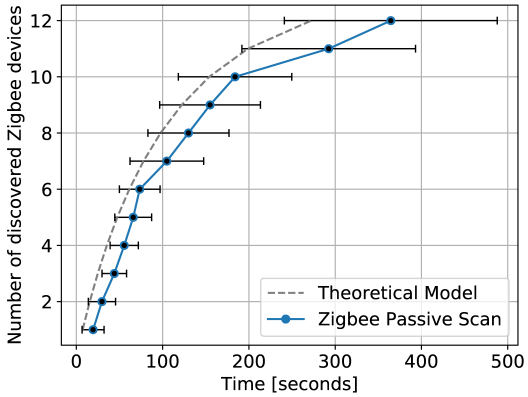


Figure 6: Zigbee theoretical model and experimental passive scan results. The 95% confidence intervals indicate a good fit.

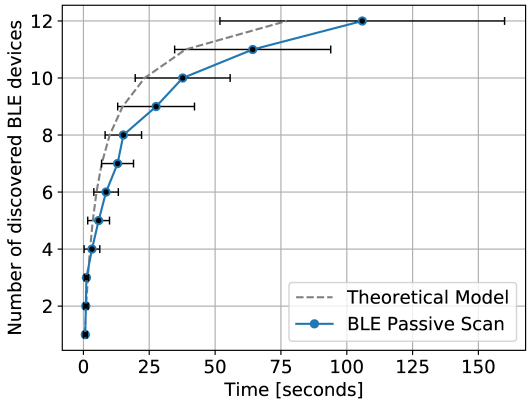


Figure 7: BLE passive scan results align closely with the theoretical model.

*1) Passive Zigbee and BLE Scans and Comparison with Theoretical Model:* We first evaluate the performance of the passive scanning algorithms (Algorithm 2) for Zigbee and BLE devices, and compare those with the expected discovery times based on the theoretical model described in Section IV-B.

To build the theoretical traffic model (see Section IV-B), we measured device characteristics of our tested devices by running one long continuous scan of 100 minutes on every Zigbee channel and on every BLE advertising channel, in order to collect a baseline of traffic for each device. The traffic statistics are shown in Table II. We set $\Delta t = 0.1$s in Eq. (15) to compute $p_i$ for each device. We then use Eq. (10) to compute the expectation of the order statistics of the discovery time of devices. Note that for Zigbee, we replace $p_i$ by $p_i/16$, since with Algorithm 2, the SDR listens to only one out of the 16 Zigbee channels at a time.

Fig. 6 shows curves for the experimental results of Zigbee passive scanning and the theoretical model. The model fits inside most of the 95% confidence intervals. This shows that our passive scan implementation is close to the best performance possible, and our testbed has minimal packet losses. The deviation from the model could be attributed to interference (e.g., from Wi-Fi) and the fact that transmissions of some Zigbee devices are not memoryless.

Fig. 7 shows experimental results for BLE passive scanning and the theoretical benchmark. The measured discovery times

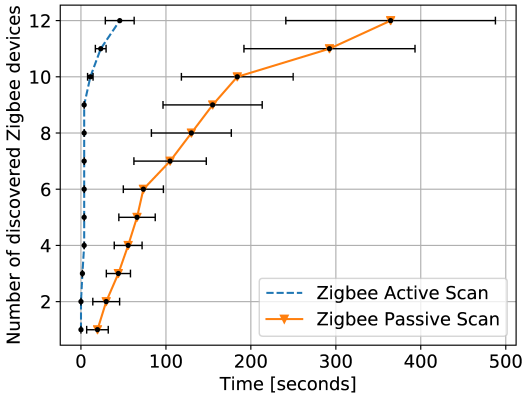


Figure 8: Zigbee active versus passive scan.

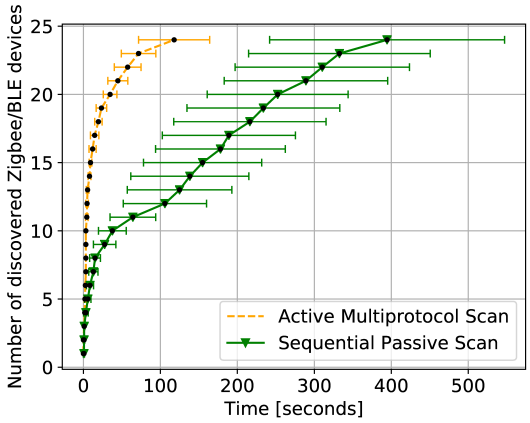


Figure 9: Zigbee/BLE multiprotocol active scan vs. sequential passive scan.

again fit the model well. Since all BLE advertising channels are equivalent, scanning is performed on channel 37 only. Note that BLE device discovery can only be performed as a passive scan, since BLE does not allow for broadcast-type scan requests as performed in Zigbee. While BLE scan requests could be a useful active scanning technique for gathering additional device data, they are always directed scans, i.e., they require knowledge of the target device's address.

*2) Active Zigbee Scan:* We next evaluate the performance of active scanning (Algorithm 4) and compare it to passive scanning in the context of Zigbee. Fig. 8 shows that the passive discovery of 12 Zigbee devices takes 365 seconds on average while active Zigbee discovery takes only 46 seconds, i.e., a reduction of 87% in the scan time. While active scanning discovers the 12 devices within one minute, passive scanning discovers only 4 devices within one minute.

Note that Zigbee supports up to 64,000 nodes per network. It is conceivable that the improvement of active scan over passive scan would be even more significant with a larger number of nodes. Zigbee routers and coordinators (mains-powered devices) are typically continuously active and will reply to beacon requests, which contributes to the discovery of several devices almost immediately during the active scan, whereas end-devices are usually battery-powered and optimized for power saving, and may not respond to beacon requests. However, since end devices are on the same channel as their coordinator, limiting the second phase of the active scan to the known active channels significantly speeds up discovery by virtue of spending more time on each relevant channel. Among our tested devices, we have three Zigbee coordinators occupying three channels: GE Link/Quirky hub on channel 11, IKEA Gateway on channel 15, and Amazon echo 4.0 on channel 20. As a result, the second phase of the active scan (cf. Algorithm 4, line 6) cycles through 3 instead of 16 channels, shortening the detection speed by a factor of roughly $16/3 = 5.333$ for the remaining end-devices.

*3) Zigbee and BLE Multiprotocol Scan:* We next evaluate the performance of active multiprotocol Zigbee and BLE scan and compare it to sequential passive scan. Sequential passive scan consists of passive BLE scan followed by passive Zigbee scan. Sequential passive scan enumerates the 24 considered devices in 395 seconds on average, while active multiprotocol Zigbee and BLE scan takes 118 seconds on average, which corresponds to a 70% improvement (Fig. 9). Within 1 minute active multiprotocol scan discovers 22 devices while sequential scan discovers only 10. Breaking down sequential passive scan into two: the first 106 seconds corresponds to a BLE passive scan, followed by 289 seconds of Zigbee scan, which is consistent with the results shown in Figs. 6 and 7. The speed-up is achieved because of two aspects: active scan and multiprotocol scan. Zigbee active scan narrows the search down from 16 to only 3 channels. Multiprotocol scan supports reception of one Zigbee and one BLE channel in parallel. Note that parallel reception is possible only if the two channels fit within the instantaneous bandwidth. As mentioned earlier, the instantaneous bandwidth for multiprotocol scan was set to 8 MHz. Three Zigbee active channels were identified, namely channel 11, 15, and 20. BLE has three well-known advertising channels, namely 37, 38, and 39. BLE channel 37 and Zigbee channel 11 can be received in parallel as well as BLE channel 38 and Zigbee channel 15. However, Zigbee channel 20 and BLE channel 39 are scanned separately since they do not fit within the same instantaneous bandwidth.

*4) Z-Wave and LoRa Multiprotocol Scan:* We next evaluate the performance of passive multiprotocol LoRa and Z-Wave scan on 900 MHz band (Algorithm 7) and compare it to sequential passive scan (Algorithm 2). Passive multiprotocol scan consists of scanning each of 3 frequency channels (2 Z-Wave and 1 LoRa) in a round robin fashion. The passive scanning operation visits the LoRa and Z-Wave channel in a round-robin fashion, one at a time. Due to having 2 Z-Wave channels (908.4 and 916 MHz) and only 1 LoRa channel (910.29 MHz), Z-Wave has an advantage in passive scanning.

Fig. 10 shows that sequential LoRa and Z-Wave scan takes about 8.1 hours on average while multiprotocol Z-Wave and LoRa scan takes 2.5 hours, which represents a reduction of about 70% in the discovery time. Within a single hour passive scan discovers less than 1 device on average while multiprotocol scan discovers 5 out of the 7 devices. This significant speed-up is achieved because multiprotocol scan receives all three channels (from the two protocols) in parallel, namely 908.4 MHz (Z-Wave R2 PHY), 910.23 MHz (LoRa uplink), and 916 MHz (Z-Wave R3 PHY).

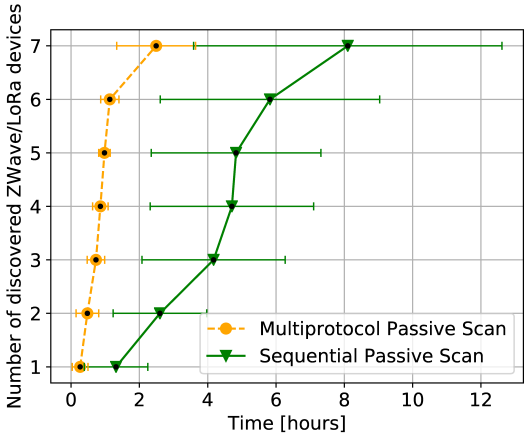


Figure 10: Multiprotocol passive scan (Z-Wave, LoRa)

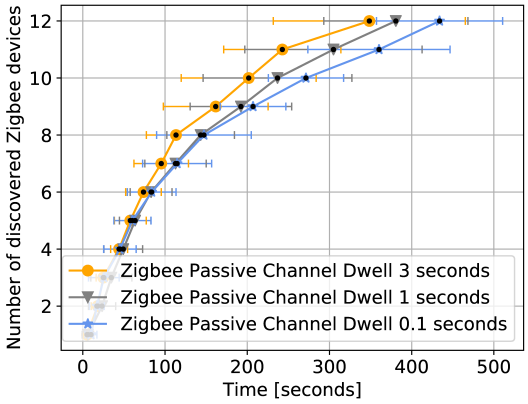


Figure 11: Zigbee passive scan discovery times for channel dwell times of 0.1, 1, and 3 seconds.

*5) Channel Dwell Time:* We last examine the impact of properly setting the value of the channel dwell parameter, which is an input to several of the algorithms. Each trial measures the time to passively scan 12 BLE and 12 Zigbee devices. Passive Zigbee scan hops between 3 active Zigbee channels. A passive BLE scan involves channel hopping between three advertising BLE channels. Our experiments indicate that the scanning times do not differ significantly for channel dwell times of 0.1, 1, and 3 seconds. Fig. 11 shows the channel dwell time experiment for Zigbee, but similar results hold for BLE. Thus, in all our experiments, we chose a value of 1 second for the channel dwell times, the only exception being in the first phase of Zigbee active scan, where the dwell time was set to 0.2 seconds (which was sufficient to receive beacon responses and move on).

## VII. In-depth Zigbee Scanning Evaluation

In this section, we provide more details regarding per-device discovery times and energy considerations, focusing on Zigbee. We aim to answer the following questions: Is active scan effective for all or only some devices? Which address types (i.e., short vs. long) are most easily discoverable? How does energy consumption of active scan compare to that of passive scan?

*1) Setup:* The IoT testbed setup is slightly different from that in Section VI, namely, there are four active Zigbee channels, not three, namely channels 11, 15, 20, and 25. The Ring Base Station and GE Quirky operate on channel 11. The Wink hub and Ikea hub are on channel 15. The Amazon Echo and Hue lights are on channel 20. The remaining Ikea, Cree and GE light bulbs are on channel 25.

The Ring Base Station, Wink hub and Echo act as Zigbee coordinators. The Ikea, Cree and GE lights are not connected to any Zigbee router nor coordinator. Nevertheless, we will show in the following that they are still discoverable.

Average values from the figures in this section are based on 100 independent trials. Figures show 95% confidence intervals based on 100 trials, except Figure 14 which is based on 10 trials.

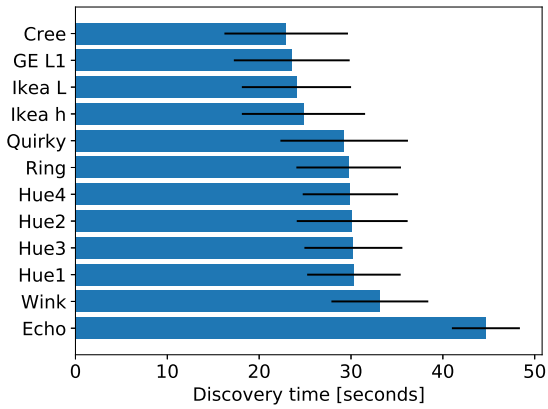


Figure 12: Zigbee device average discovery times during active scan with 95% confidence intervals.

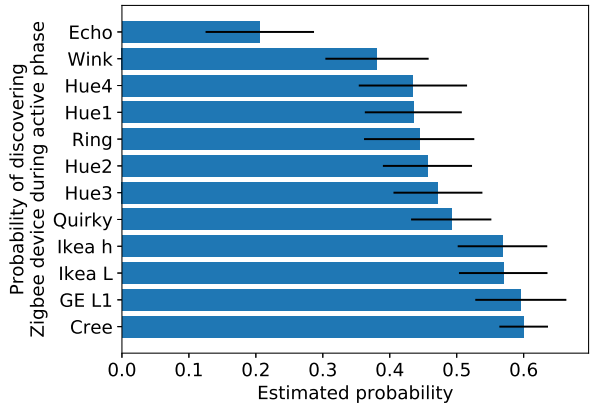


Figure 13: Estimated probability of discovering Zigbee device during phase 1 of active scan with 95% confidence intervals.

*2) Per-device discovery time:* Figure 12 shows the average per-device discovery times with active scan (Algorithm 4). Per-device discovery time is the average time until a specific device is discovered, through either its short or long address. A device can be discovered either during phase 1 of active scan (namely the probing stage) or during phase 2 (namely the passive stage) of active scan. Figure 12 shows that the Cree light bulb takes on average the shortest time to be discovered and while the Echo takes the longest. This figure is consistent

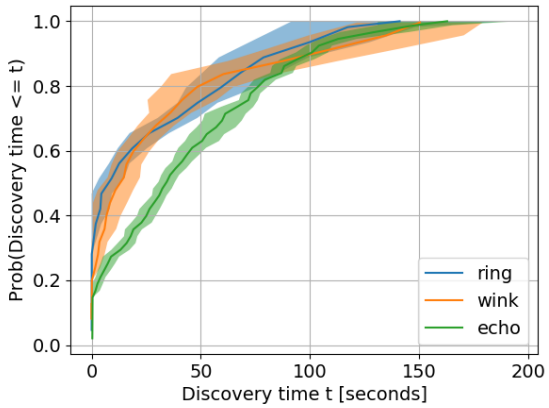


Figure 14: Cummulative distribution function (CDF) of discovery time of Zigbee devices with active scan.

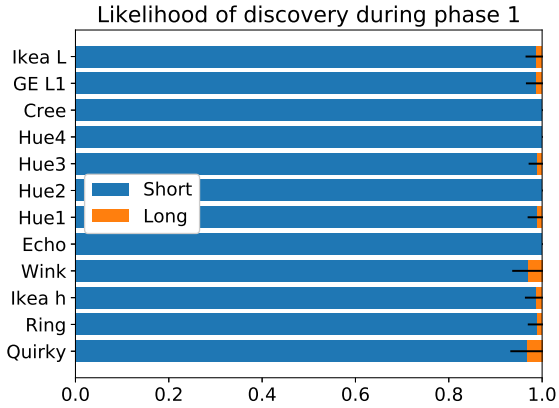


Figure 15: Likelihood of discovery of short Zigbee device addresses during phase 1 of active scan.

with the Zigbee active scan curve shown in Figure 8 in that it takes on average about a minute to discover all devices.

Figure 13 shows the estimated probability of discovering Zigbee devices during phase 1 of active scan. We observe that the Cree, GE Lamp1, Ikea light, and Ikea hub are the most likely to be discovered during phase 1 of active scan while the Echo and Wink devices are the least likely. We note that Zigbee coordinators (Wink, Quirky, and Echo) are no more likely to be discovered during phase 1 than other types of devices.

To cross-validate our results, we plot the cumulative distribution function (CDF) of the discovery time of the Ring, Wink, and Echo devices, as shown in Fig. 14. The figure confirms that either one of the devices are quickly discovered during the probing stage of active scan, or more slowly during its passive stage.

While the Ikea, Cree and GE lights are orphan Zigbee devices (not connected to a router or coordinator), they are still discovered by `IoT-Scan`. Furthermore, Wink and Ikea hub are Zigbee coordinator and router on channel 15 without any end-devices, yet `IoT-Scan` is still effective. Of course, device discovery with `IoT-Scan` also works in fully connected Zigbee network, e.g., the Echo hub coordinating Hue lights.

*3) Discovery time by address type:* Remember that Zigbee devices map to different addresses, namely 16-bit short and 64-bit long address. During the device discovery process, we will mark a device as discovered after seeing any of the two addresses. In practice, devices tend to be more readily discovered by their short addresses.

Figure 15 shows that, during phase 1 of active scan, devices are almost always discovered by their short Zigbee address. Note that Cree, Hue4, Hue2, and Echo devices were only discovered via short addresses, hence the lack of confidence intervals. While during the entirety of active scan, devices can also be discovered via their long addresses (see Figure 16). Nevertheless, in aggregate, discovery through the short address is still more likely.

We note in passing that Zigbee coordinators may have

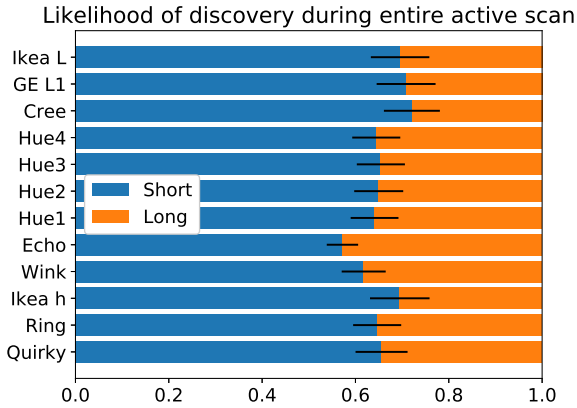


Figure 16: Likelihood of discovery of short Zigbee device addresses during the entire active scan.

the same short address 0x00000000. However, this is not a problem in our set up, because there is only one network per channel (each network has a unique coordinator). If multiple networks operate on the same channel, the coordinators could be distinguished by their long addresses or PAN IDs.

*4) `IoT-Scan`'s Energy Consumption:* Our next goal is to assess the energy consumption of active scan and compare it to that of passive scan. Specifically, we show that, while active scan relies of the transmission of probing packets that expend some energy, the energy savings achieved by reducing the scanning time ends up being much more significant.

We obtain the energy consumption by measuring the SDR transmission and reception powers. Specifically, the power consumption of the SDR is measured using a MakerHawk USB power meter [44] using a USB3 cable (see Figure 17).

In more detail, the SDR's transmission power is measured on the power meter by having the host computer run a GNURadio flowgraph that constantly transmits a constant carrier at Zigbee's 2.4GHz band, across different transmit gains. The SDR's received power is measured on the power meter while the host runs a flowgraph that constantly receives

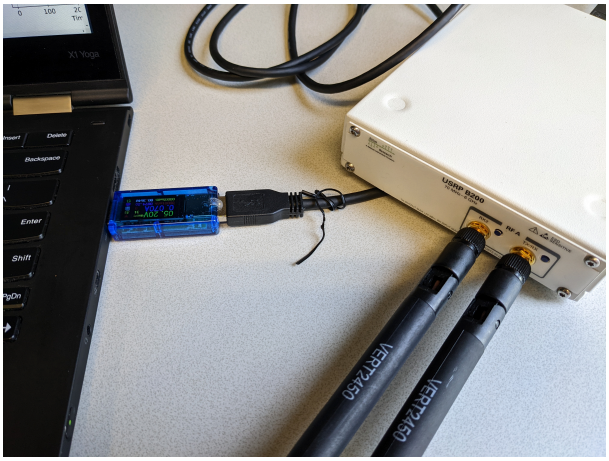

Figure 17: SDR power consumption is measured with MakerHawk a USB power meter.

samples (with receive gain configuration set to automatic gain control or AGC) across different sample rates. The measured transmission and reception powers are shown in Table III. We choose these sample rates (and their multiples) in accordance with the protocols `IoT-Scan` supports. Table III shows that the transmission gain and the sample rate have relatively limited impact on the power consumed during transmission and reception, respectively.

Our active scan energy calculation is based on the measured transmission power of $P_{tx} = 2.456$ W at a 80 dB transmit gain. The passive scan energy calculation is based on the measured receive state power consumption of $P_{rx} = 2.109$ W at 4 Msps. Thus, we conclude that transmission and reception powers do not differ by much. Therefore, the energy consumption of different types of scans (i.e., active vs. passive) mostly depends on the total scan time and less on the radio's state (i.e., transmission vs. reception).

We calculate the total cumulative energy consumption of scans as the power multiplied by the scanning time. The average passive and active scan times for Zigbee are listed earlier in this section (Subsection 2). We assume that Phase 1 of active scan takes around $T_{tx} = dwell\_time * |channel\_list| = 0.2 * 16$ s= 3.2 s during which SDR transmits and receives on all 16 channels. During phase 1, the radio is in both transmission and reception state. In this case, we assume that the radio consumes the higher of the two measured powers, which is the transmission power.

Through this computation, we obtain that active scan consumes on average $P_{act} = P_{tx} * T_{tx} + P_{rx} * (t_{scan} - T_{tx}) = 98$ J, while passive scan consumes $P_{pas} = P_{rx} * t_{scan} = 770$ J. This shows that Zigbee **Active_Scan** saves energy, in addition to saving time, compared to **Passive_Scan**.

## VIII. Conclusion

We presented `IoT-Scan`, an extensible multi-protocol network reconnaissance tool for the Internet of Things that can be employed for security auditing and network monitoring. `IoT-Scan` leverages the capabilities of SDRs to process multiple streams in parallel. Accordingly, we introduced several scanning algorithms and evaluated them both theoretically and experimentally. Using the theoretical model, we showed that our implementation is efficient and achieves minimal packet loss in reception. We implemented multi-protocol, multi-channel scanning both on the 2.4GHz band for Zigbee and BLE, and on the 900 MHz band for LoRa and Z-Wave, and demonstrated significant improvement over sequential passive scanning.

Table III: SDR transmit (TX) and receive (RX) power consumption vary little with the output gain and the sample rate.

| Output gain | Tx Power |
|---|---|
| [dB] | [Watt] |
| 0 | 2.254 |
| 20 | 2.259 |
| 40 | 2.254 |
| 60 | 2.265 |
| 70 | 2.265 |
| 80 | 2.383 |
| 90 | 2.383 |

| Sample rate | Rx power |
|---|---|
| [Msps] | [Watt] |
| 0.125 | 1.916 |
| 0.25 | 1.921 |
| 0.5 | 1.926 |
| 1 | 1.931 |
| 2 | 1.955 |
| 4 | 1.997 |
| 8 | 2.077 |

Our SDR implementations should prove especially useful in overcoming the incompatibility of different protocols based on the same PHY layer. For instance, besides Zigbee, there exist several IoT protocols based on the IEEE 802.15.4 standard, such as Thread [13] and WirelessHART [45]. We expect that these protocols could readily be integrated into `IoT-Scan`.

The design of `IoT-Scan` does not raise ethical issues in itself. However, like other penetration testing tools, usage of this tool does require explicit consent from the owners of the devices under test. Specifically, active scanning, while brief, may interfere with existing network traffic and delay time-sensitive communication. A major advantage of `IoT-Scan` versus a tool like Nmap is that it also supports a passive scanning mode, which does not generate traffic.

### A. Discussion and Future Work

This paper opens several avenues for future work. First, one could explore FPGA implementations of `IoT-Scan` to increase the number of channels and protocols that can be decoded in parallel and further speed up the discovery of IoT devices. While this should yield useful performance improvements, we expect that such implementations would still rely on the algorithms introduced in Section III.

Another interesting research avenue lies in the design of active scanning methods for LoRa and Z-Wave, as devices in these protocols transmit sparingly. We have publicly released data traces obtained with `IoT-Scan` in [46]. We envision that these traces should be useful for the design and evaluation of scanning algorithms and other IoT-related research (see, e.g., [47]).

While IoT-Scan hold promise for enhancing device discovery and communication in IoT networks, it faces several inherent constraints. Overcoming these limitations in the future will requires interdisciplinary research efforts, combining aspects of communications engineering, signal processing, and operating systems. These limitations encompass various aspects such as:

- **Limited bandwidth:** Many SDR frameworks utilize

general-purpose OS, which often have limited processing power and and not designed for real-time operations. This can lead to issues such as overlows and underflows in buffer management, affecting reliability of data transmission and reception. To overcome this, future work could focus on developing a lightweight multi-protocol real-time scheduling mechanisms [48].

- **Packet collisions:** Number of devices that can be scanned simultaneously by active scan is limited due to collisions in response messages. Future work could investigate techniques such as directed or non-broadcasting probe requests [49].
- **Ineffective Adaptive Algorithms:** Ineffective carrier frequency compensation and timing recovery algorithms' implementations can result in signal distortions, leading to increased bit error rates and reduced overall network performance. Future research could explore more advanced algorithms that address these challenges such as [50].
- **SDR Non-linearity:** The non-linearity of SDR components can introduce signal distortions, affecting the quality of communication and exacerbating energy efficiency of SDRs. Researchers could delve into developing advanced signal processing techniques to mitigate non-linear effects, potentially incorporating adaptive compensations, such as digital predistortion [51].

## ACKNOWLEDGEMENTS

This research was supported in part by the US National Science Foundation under grants CNS-1717858, CNS-1908087, CCF-2006628, EECS-2128517, AST-2229104, and by an Ignition Award from Boston University.

## REFERENCES

[1] Ericsson. (2020) Internet of Things Forecast. [Online]. Available: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast

[2] Bluetooth Special Interest Group (SIG). (2016) Bluetooth Core Specification. v5.0. [Online]. Available: https://www.bluetooth.com/specifications/specs/core-specification/

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

[4] International Telecommunication Union. (2015) G.9959: Short range narrow-band digital radiocommunication transceivers - PHY, MAC, SAR and LLC layer specifications. [Online]. Available: https://www.itu.int/rec/T-REC-G.9959

[5] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming, and A. Burg, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," in *SPAWC 2020*, 2020, pp. 1–5.

[6] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: network device behavior modeling for identifying unknown IoT devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*. New York, NY, USA: ACM, Apr, pp. 106–117.

[7] D. Y. Huang, N. Apthorpe, F. Li, G. Acar, and N. Feamster, "IoT Inspector," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–21, Jun 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3397333

[8] Palo Alto Networks Inc., "The Connected Enterprise: IoT Security Report 2020," Palo Alto Networks, Inc., Santa Clara, CA, USA, Tech. Rep. [Online]. Available: https://www.paloaltonetworks.com/resources/research/connected-enterprise-iot-security-report-2020

[9] B. Burns, D. Killion, N. Beauchesne, E. Moret, J. Sobrier, M. Lynn, E. Markham, C. Iezzoni, P. Biondi, J. S. Granick, S. Manzuik, and P. Guersch, *Security power tools*. O'Reilly Media, Inc., 2007.

[10] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: a comprehensive survey," *IEEE communications surveys & tutorials*, vol. 16, no. 3, pp. 1496–1519, 2013.

[11] M. Klein. (2020) What is Z-Wave Long Range and How Does it Differ from Z-Wave? [Online]. Available: https://z-wavealliance.org/what-is-z-wave-long-range-and-how-does-it-differ-from-z-wave/

[12] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*, 2nd ed. Sunnyvale, CA, USA: Insecure.Com LLC, 2008.

[13] Thread Group. (2023) What is Thread? [Online]. Available: https://www.threadgroup.org/What-is-Thread/Overview

[14] Connectivity Standards Alliance. (2023) Matter: Smart Home Device Solution. [Online]. Available: https://csa-iot.org/all-solutions/matter/

[15] T. Ulversoy, "Software Defined Radio: Challenges and opportunities," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 531–550, 2010.

[16] GNU Radio Project. (2022) GNU Radio. [Online]. Available: https://www.gnuradio.org

[17] Bastille Research. (2015) scapy-radio. [Online]. Available: https://github.com/BastilleResearch/scapy-radio

[18] Y. He, J. Fang, J. Zhang, H. Shen, K. Tan, and Y. Zhang, "MPAP: Virtualization Architecture for Heterogenous Wireless APs," *ACM SIGCOMM Computer Communication Review*, no. 4, pp. 475–476, Aug.

[19] P. Flajolet, D. Gardy, and L. Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, no. 3, pp. 207–229, Nov 1992.

[20] E. Anceaume, Y. Busnel, and B. Sericola, "New Results on a Generalized Coupon Collector Problem Using Markov Chains," *Journal of Applied Probability*, vol. 52, no. 2, p. 405418, 2015.

[21] Ettus Research. (2022) USRP B200. [Online]. Available: https://www.ettus.com/all-products/ub200-kit/

[22] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski, "IoTScan: Network Reconnaissance for the Internet of Things," in *IEEE Conference on Communications and Network Security (CNS) 2022*, Austin, TX, USA, October 2022.

[23] A. Heinrich, M. Stute, and M. Hollick, "BTLEmap: Nmap for Bluetooth Low Energy," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec '20. Association for Computing Machinery, 2020, p. 331333. [Online]. Available: https://doi.org/10.1145/3395351.3401796

[24] R. A. Sharma, E. Soltanaghaei, A. Rowe, and V. Sekar, "Lumos: Identifying and localizing diverse hidden {IoT} devices in an unfamiliar environment," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1095–1112.

[25] J. Tournier, F. Lesueur, F. Le Mouël, L. Guyon, and H. Ben-Hassine, "IoTMap: A protocol-agnostic multi-layer system to detect application patterns in IoT networks," in *10th International Conference on the Internet of Things (IoT 2020)*, Malmö, Sweden, Oct. 2020.

[26] J. Mikulskis, J. K. Becker, S. Gvozdenovic, and D. Starobinski, "Poster: Snout - An Extensible IoT Pen-Testing Tool," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2529–2531.

[27] S. Bak and Y.-J. Suh, "Designing and Implementing an Enhanced Bluetooth Low Energy Scanner with User-Level Channel Awareness and Simultaneous Channel Scanning," vol. 102, no. 3. The Institute of Electronics, Information and Communication Engineers, 2019, pp. 640–644.

[28] C. D. Kilgour, "A Bluetooth low-energy capture and analysis tool using software-defined radio," Master's Thesis, Simon Fraser University, 2013. [Online]. Available: http://summit.sfu.ca/item/12931

[29] W. Park, D. Ryoo, C. Joo, and S. Bahk, "BLESS: BLE-aided Swift Wi-Fi Scanning in Multi-protocol IoT Networks," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[30] J. Hall, B. Ramsey, M. Rice, and T. Lacey, "Z-wave Network Reconnaissance and Transceiver Fingerprinting Using Software-Defined Radios," in *ICCWS 2016*.

[31] L. Choong, "Multi-Channel IEEE 802.15.4 Packet Capture Using Software Defined Radio," *UCLA Networked & Embedded Sensing Lab*, vol. 3, pp. 1–20, 2009.

[32] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A Multi-Channel Software Decoder for the LoRa Modulation Scheme," in *IoTBDS 2018*, 2018, pp. 41–51.

[33] B. Schmeiser, "Batch Size Effects in the Analysis of Simulation Output," *Operations Research*, vol. 30, no. 3, pp. 556–568, 1982.

[34] Connectivity Standards Alliance. (2021) Zigbee – The Full-Stack Solution for All Smart Devices. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/

[35] N. K. Gupta, *Inside Bluetooth Low Energy*, 2nd ed. Boston, London: Artech House, 2016.

[36] J. K. Becker, D. Li, and D. Starobinski, "Tracking Anonymized Bluetooth Devices," in *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, Jul. 2019, pp. 50–65.
[37] G. Celosia and M. Cunche, "Discontinued privacy: Personal data leaks in apple Bluetooth-low-energy continuity protocols," in *Proceedings on Privacy Enhancing Technologies*, vol. 2020, 2020, pp. 26–46.
[38] LoRa Alliance. (2018, July) Lorawanő specification v1.0.3. [Online]. Available: lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf
[39] Jon Callas. (2022) Understanding amazon sidewalk. [Online]. Available: https://www.eff.org/deeplinks/2021/06/understanding-amazon-sidewalk
[40] LoRa Alliance, "LoRaWANő Regional Parameters," 2021. [Online]. Available: lora-alliance.org/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf
[41] C. Bernier, F. Dehmas, and N. Deparis, "Low complexity lora frame synchronization for ultra-low power software-defined radios," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3140–3152, 2020.
[42] C. W. Badenhop, S. R. Graham, B. W. Ramsey, B. E. Mullins, and L. O. Mailloux, "The Z-Wave routing protocol and its security implications," *Computers & Security*, vol. 68, pp. 112–129, 2017.
[43] Marija Dimitrijevic. (2018) Replacing many RF receivers with only ONE using Channelization. [Online]. Available: ettus.com/wp-content/uploads/2018/12/Channelization_-_Article_.pdf
[44] Tina Zhu. (2023) MakerHawk USB 3.0 Tester. [Online]. Available: https://www.makerhawk.com/products
[45] FieldComm Group, "WirelessHART: HART Without The Wires," 2021. [Online]. Available: https://www.fieldcommgroup.org/technologies/wirelesshart
[46] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski. (2022) IoT-Scan Traces. [Online]. Available: https://github.com/nislab/iot-scan
[47] J. K. Becker and D. Starobinski, "Optimizing Freshness in IoT Scans," in *8th IEEE World Forum on Internet of Things (IEEE WF-IoT) 2022*, Yokohama, Japan, November 2022.
[48] J. Bush. (2019) Radio scheduling in dynamic multiprotocol iot applications. Aug 29, 2023. [Online]. Available: https://www.electronicspecifier.com/industries/wireless/radio-scheduling-in-dynamic-multiprotocol-iot-applications
[49] M. Jaakkola, A. Suomi, and J. Poyhonen, "Usage of multiple ssids for doing fast wlan network discovery," Jan. 4 2007, uS Patent App. 11/372,037.
[50] E. Faulkner, Z. Yun, S. Zhou, Z. J. Shi, S. Han, and G. B. Giannakis, "An advanced gnu radio receiver of ieee 802.15. 4 oqpsk physical layer," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9206–9218, 2021.
[51] R. Marsalek and M. Pospisil, "Evaluation of digital predistortion using the usrp n200 software defined radio transceiver," in *2014 NORCHIP*. IEEE, 2014, pp. 1–4.