Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 155 (2019) 51–58

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 19-21, 2019, Halifax, Canada

Connection-less BLE Performance Evaluation on Smartphones

Joshua Siva[a,*], Jian Yang[a], Christian Poellabauer[a]

[a]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, United States*

Abstract

Previous studies concerning theoretical discovery latency for Bluetooth Low Energy (BLE) have been repeatedly validated on dedicated BLE Systems on Chips (SoCs), but to a much lesser extent on multi-radio SoCs such as those found in smartphones. In this paper, we evaluate neighbor discovery latency and energy consumption across various parameters available on an Android smartphone under API 24 for Bluetooth 4.2. We further investigate the effect of concurrent Wi-Fi traffic on energy consumption and BLE packet reception over advertising channels. We found that average discovery latency can be comparable to what state of the art models predict, while exhibiting higher variability across advertising intervals. Energy consumption deviates from that predicted by the chosen models and Wi-Fi interference tests show a much greater impact on energy consumption and discovery latency when interference traffic involves the BLE scanning activity.

Keywords: Bluetooth Low Energy (BLE); Android; Discovery Latency; Energy Consumption; Wi-Fi Interference

1. Introduction

Bluetooth Low Energy (BLE) is a wireless technology that was first released in 2010 and has since found a home in many devices around the world. The 2018 market update provided by the Bluetooth Special Interest Group (SIG) predicted that 100% of phones, tablets, and laptops shipped in 2018 would have Bluetooth installed and 97% of all Bluetooth devices shipped in 2022 would have LE capabilities [1]. BLE stands out among IEEE 802.15.4, SimpliciTI, ZigBee, and ANT [15, 6] with its lower power consumption and wide installation in smartphones.

Evaluations of various aspects of BLE performance have focused disproportionately on dedicated BLE SoCs with a few notable exceptions such as the works of Fürst et al. [7], Sikora et al. [18], and Giovanelli et al. [8]. Fürst et al. provides a cross sectional review of basic BLE performance metrics across various Android phone models, SoCs, and APIs. In this paper we seek to expand on this type of investigation by evaluating BLE performance across numerous metrics on Android 7.0 devices. We evaluate these results against theoretical models and empirical evidence from

* Corresponding author
E-mail address: jsiva@nd.edu

1877-0509
10.1016/j.procs.2019.08.011

investigations into dedicated BLE SoCs. Moreover, we evaluate the effect that concurrent Wi-Fi operations have on BLE discovery latency and energy consumption. Our contributions can be summarized as:

- Evaluation of neighbor discovery latency and energy consumption of BLE on Android devices, with comparison to existing evaluation models.
- Assessment of the effect of concurrent Wi-Fi operations on connection-less BLE communications.

This paper is structured as follows: we review the most relevant connection-less BLE communication configurations in Section 2, followed by related work in Section 3. The evaluation methodology and results are explained in Sections 4 and 5, respectively. We conclude the paper and discuss future work in Section 7.

2. BLE Communications

BLE was first introduced in the Bluetooth 4.0 standard and then updated with the Bluetooth 4.1, 4.2, 5.0, and 5.1 specifications. Our evaluation focuses on Bluetooth 4.2 because it has been available long enough that we assume it has become commonplace in the market. For this paper, we focus on the connection-less BLE communication that relies on *advertisers* and *scanners*. Briefly, this refers to when one device (an advertiser) broadcasts a short packet across three channels sequentially while another device (the scanner) listens on these same three channels, one at a time. The actions of these two devices are not synchronized, so there is uncertainty in the amount of time it takes for a packet to be received. Further, the advertiser can broadcast whether it will allow connections or not, which changes whether the advertiser will listen for responses and how frequently advertisements can be sent. For both advertiser and scanner, there is freedom to adjust the device duty cycle.

In theory the channel changes instantaneously, but real BLE radios deviate quite widely from the expected behavior, as shown by Perez-Diaz de Cerio et al. [4]. They found that the tested devices fell into two groups which experienced different types of delays when changing the scan channel or receiving an advertisement. These “non-idealities” suggest that if there are deviations from expectations with simple, dedicated BLE radios then there might be more unexpected behavior from a multi-radio SoC such as those found in smartphones.

3. Related Work

3.1. Discovery Latency

The *neighbor discovery latency*, has been widely studied. To provide a ground truth for our investigation of Android BLE behavior, we refer to the one-way communication latency developed by Kindt et al. [12] as this model closely matches the behavior of a real-world BLE radio: the NRF51822 running the BLESSED Bluetooth networking stack. We use Kindt et al.’s general model for determining the average and upper bound for discovery latency. This model has no closed form but can be computed using their provided software [11]. To offer a secondary point of comparison we include the model developed by Liu et al. [13] as this takes channel hopping into account. In both models we add 5ms to the advertising interval to account for the average behavior of the random delay as done in [12].

3.2. Energy Consumption

The energy consumption models for connection-less communications can be roughly split into two groups: those that break the advertisement and scanning processes down into the chip-level events as found in [10, 14] and those that view the energy consumption from a packet level as in [15]. Since all cases base measurements on some particular device, there is the risk of being unable to generalize results to other chips. It is not clear which radio from the previous works would provide comparable measurements to the multi-radio SoC from the Android smartphones in our experiments, so we will compare our results to multiple models, i.e., the models of Kindt et al. [10] and Mikhaylov et al. [15]. These models show agreement with each other even though the former is based on adding up the constituent sources of drain during an advertising event while the latter provides a packet level energy consumption. Additionally, the energy consumption models are based on measurements from two different SoCs (BLE112 and CC2540, respectively).

3.3. BLE on Android

Similar work has also been conducted to evaluate BLE performance on Android smartphones. A study by Bronzi et al. [3] on iPhones evaluated the effect of nearby Wi-Fi interference on connection-oriented communications and found that while the discovery latency increased in the presence of up to three Wi-Fi access points, the communication over the data channels was resistant to the interference. This traffic does not involve the phone being tested, which is something that we examine in Section 5.4.

An evaluation of BLE performance on three different Samsung Android devices by Radhakrishnan et al. [17] found that BLE scanning can consume a fair amount of energy. The authors' explanation was that the BLE portion of the SoC could not be turned on separately from the Bluetooth Classic portion and that accessing the wake-lock when the screen was asleep increased power consumption. This second part is derived from the observation that the average BLE power consumption increased when the screen was turned off. However, the investigation by Czurak et al. [5] found the screen being on *increased* the power consumed by BLE. Although, another important point to note from this study is the great variability in battery consumption between the two devices studied. Variability in BLE performance metrics across Android devices is a feature further reinforced by the previously mentioned investigation by Fürst et al. [7].

4. Methods

4.1. Code Review

To determine the limitations imposed by Android on BLE performance, we reviewed developer documentation from Android and Nordic (a producer of BLE chips and software) as well as the Android source code for Android 7.0 and 8.0. We found that scans that last longer than 30 minutes are stopped by the Android OS and restarted "opportunistically", which means that results are only reported when another app performs a BLE scan. To sidestep this issue, we run our tests no longer than 10 minutes. Another constraint is that Android 7.0 and 8.0 do not allow control over the duration of scanning and advertisement parameters at the granularity mentioned in the specification (0.625 ms) [2]. Android allows setting advertising or scanning to LOW_POWER, BALANCED, or LOW_LATENCY. For scanning, these translate to a scan window of 500 ms, 2000 ms, and 5000 ms, respectively, with a scan interval of 5000 ms. For the advertising interval, the settings translate to 1000 ms, 250 ms, and 100 ms.

4.2. Collection Methods

We developed an Android BLE application that accepts flags and arguments over the Android Debug Bridge (ADB) v1.0.35 [9] to set the advertising and scanning parameters. We used a Raspberry Pi 3 Model B with Python 2.7+ and the USB hub per-port power control tool (uhubctl) [16]. A Python script resets all logs, runs jobs on multiple phones, programmatically unplugs the phones, and collects the Android bug report and custom log files.

The Android devices were set to fall asleep after 30 minutes, screen brightness set to minimum, and Wi-Fi turned off (for the non-Wi-Fi interference tests). Furthermore, we added a Bluetooth adapter reset function that is called when the app starts. After numerous experiments it was found that the Bluetooth adapter could get stuck in a state where BLE scans and advertisements would never start successfully. This is a known problem, but there is no clear solution beyond disabling and re-enabling Bluetooth. We also added the acquisition of a wake lock in the app to ensure that our test application was allowed to run without interference from the Android operating system. Our BLE testing application always runs in the foreground as running it in the background can greatly diminish performance; bringing a different application to the foreground caused the average number of received advertisements to drop from approximately 4000 to less than 20 over the same duration of 10 minutes. For the Wi-Fi interference tests we used the Magic iPerf app and the command line utility iPerf3. The app is started on the appropriate device before the BLE app and kicks off a UDP server as a daemon. The Raspberry Pi can then send traffic or tell the daemon to start sending traffic at a specified rate.

Using our Android app on the Moto g5 Plus running Android 7.0, we ran tests in one of the following configurations for 10 minutes and repeated each test at least five times:

- **idle**: do nothing but open app
- **scanner**: device scans as LOW_LATENCY, BALANCED, or LOW_POWER
- **advertiser**: device advertises as LOW_LATENCY, BALANCED, or LOW_POWER and either connectable or non-connectable for each advertising interval
- **scanner receiving**: one device scans while the other advertises; the same sets of parameters used in the tests above are used here in all combinations (18 total).
- **scanner receiving with Wi-Fi**: one device scans, the other advertises (each LOW_LATENCY and non connectable), and Wi-Fi traffic is either directed into or out of one of the devices; the Wi-Fi (802.11n) scheduling interference takes the form of the Wi-Fi being turned on and associated or UDP traffic varied across 1 Kbps, 512 Kbps, and 2 Mbps.

For every test, the advertiser uses the maximum power of 1 dBm.

4.3. Metrics

All power consumption data is extracted from the Android bug report that is captured using ADB. Since the logs and battery data are reset before each run, we can collect the overall charge consumed (in mAh) during each test as well as an estimate of the charge consumed by Bluetooth. These values are found in the bug report under the heading "Statistics since last charge" in the fields labelled "Discharge" and "Bluetooth Power drain". The energy consumption due to advertising or scanning is calculated by subtracting the average consumption while the app is sitting idle from the measured values during BLE operation. The discovery latency is estimated by collecting the nanosecond time stamps from the received advertisements (provided by the Android API) and calculating the inter-advertisement times. This same method was used by Perez-Diaz de Cerio et al. [4] due to the ease of collection. The average of these times is considered the average discovery latency.

5. Experimental Results

5.1. Discovery Latency

To evaluate the discovery latency achieved on Android devices,we provide the results of Kindt et al.'s and Liu et al.'s models (in orange and red, respectively) alongside our results (in blue) in Figure 1.

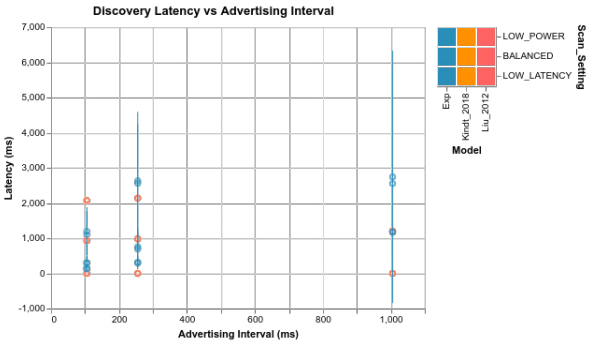

Fig. 1. Discovery latency between two devices by varying advertising and scan settings. Our results are in blue and labelled "Exp". Discovery models included for comparison.

The vertical lines show the standard deviation of our measurements. Note that the models from Liu et al. and Kindt et al. are essentially identical and therefore overlap. Figure 1 shows that the discovery latency on our Android devices were not far off from the times predicted by the models, particularly for shorter advertising intervals and larger scan windows. We found that the connectable advertisements had a slightly higher discovery latency on average than non-connectable advertisements. The standard deviation from these averages increases as the scan window shrinks (indicating a lower power scan setting) and also as the advertising interval increases.

5.2. Reception Efficiency

Sikora et al. [18] found that the efficiency of packet reception was rather poor. However, the authors accounted for every advertisement sent even though the same scanner is very unlikely to receive two advertisements from the same advertising event as it would need to change listening channels within a very narrow window of time. If the authors, instead, were to count the number of successfully received advertising *events* (which could include some very minor over-counting) then the reception rates look more like the left side of Table 1. Our results are found on the right side of Table 1. Like Sikora et al., we found that the scan setting primarily determines the packet reception performance. However, we also noticed a much larger increase in packet reception rate from BALANCED to LOW_LATENCY than did those authors. Further, there is a distinct pattern of increasing reception rate with decreasing advertising interval.

	Sikora et al. [18]			This Paper		
	ADV_LL	ADV_B	ADV_LP	ADV_LL	ADV_B	ADV_LP
SC_LL	46.2%	45.9 %	46.5%	81.4%	83.6 %	85.7%
SC_B	36.3%	35.1%	36.3%	35.8%	36.7%	39.5%
SC_LP	11.7%	13.8%	12.6%	9.5%	9.6%	0%

Table 1. Adjusted reception rates from [18] along with the packet reception rates from our experiments with non-connectable advertisements.

5.3. Energy Consumption

Before evaluating the energy consumption of communication we first established a baseline measurement of the energy expenditure of the scanning and advertising operations. The results of these experiments are shown on the left in Figure 2. The histogram illustrates how varied the measured idle Bluetooth energy consumption (orange) can

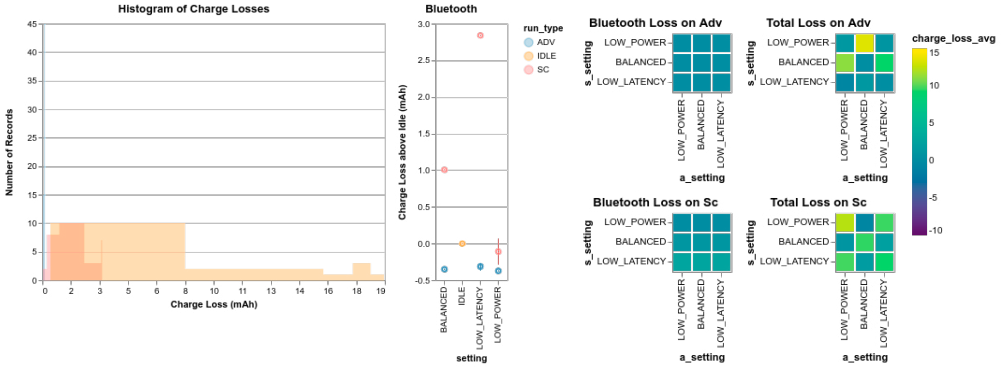

Fig. 2. Bluetooth energy consumption by varying advertising and scan settings *(left, middle)*. Average energy consumed (mAh) by Bluetooth and Overall for Advertiser and Scanner for a connectable advertisement exchange over 10 minutes *(right)*.

be. To the left side of the histogram (in the darker area) one can see that in 10 minutes, the scanner is unlikely to consume more than 3 mAh. The leftmost side of the histogram has a blue streak that represents the fraction of a mAh that the advertising operation is likely to consume in 10 minutes. The chart in the middle breaks down the average charge consumed by the Bluetooth chip along with the standard deviation (after subtracting the average idle charge). Interestingly, the charge consumed by each of the advertisement settings is negative. This could be indicative of the high variability found in the idle measurements or possible inaccuracies in the estimation of energy consumption in the Android bug report. Either way, the salient point from this chart is that the charge consumed by scanning is significantly higher than advertising and increases as the scan window increases.

We further evaluated the energy consumption when a scanner was listening to an advertiser. The results for the connectable case are shown on the right in Figure 2. Here, we can see that relative to the overall energy consumed, the Bluetooth consumption is fairly low. As expected, the Bluetooth scanner energy consumption is higher than the advertiser and higher, in particular, for larger scan windows. The overall energy consumption on the right of this chart

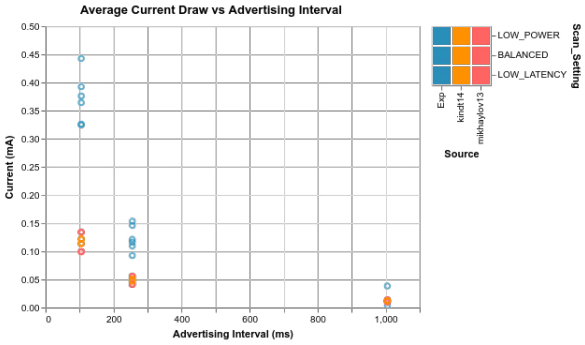

Fig. 3. Current draw of Android phones compared to the models derived from Kindt et al. [10] and Mikhaylov et al. [15].

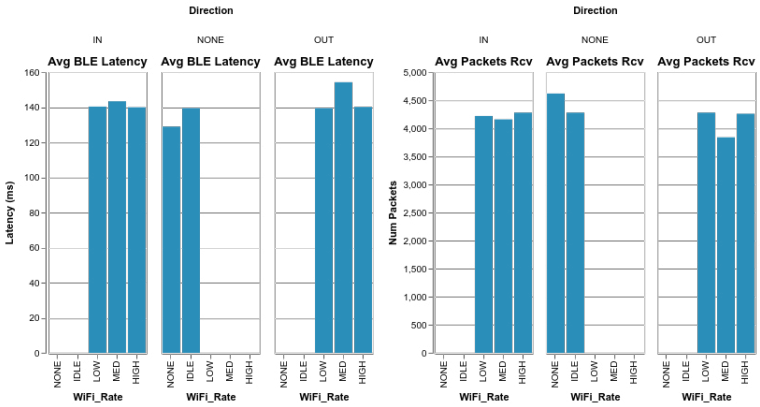

Fig. 4. Discovery latency *(left)* and number of received packets *(right)* for Wi-Fi IDLE (associated with AP, no traffic), LOW (1 Kbps), MED (512 Kbps), and HIGH (2 Mbps) involving the advertiser.

shows a checkerboard pattern of energy consumption for the advertiser and scanner where the energy consumption is never high for both at the same time.

To compare the energy consumption to the energy models from Kindt et al. and Mikhaylov et al. appropriately, we calculate the average current for the duration of the exchange. The results are shown in Figure 3. Here we see that the average Bluetooth current is higher for our tests than predicted by the models; however, the discrepancy between the expected and measured/calculated values decreases as the advertising interval increases.

5.4. Wi-Fi Interference

For this set of tests, we look at antenna time sharing for Bluetooth and Wi-Fi by examining advertiser and scanner separately and either sending packets to or receiving packets from the particular device while it operates. The advertiser's energy consumption was not significantly affected by the Wi-Fi traffic in either direction. There was a modest reduction in the number of received packets and an increase in the discovery latency (time between consecutive advertisements) as illustrated in Figure 4. Transmitting (Wi-Fi Out) appeared to have a greater impact on the discovery latency and packet loss than receiving. The scanner was much more heavily impacted by Wi-Fi traffic as illustrated by the energy consumption on the left in Figure 5 and the discovery latency and packet reception count in the middle and on the right, respectively. The energy consumption for Bluetooth is lowest when the Wi-Fi traffic rate is the highest and is lower for when the scanning device is transmitting rather than receiving. These results go hand-in-hand with the latency and packet count plots where the latency increases with the amount of traffic and the number of packets received decreases.

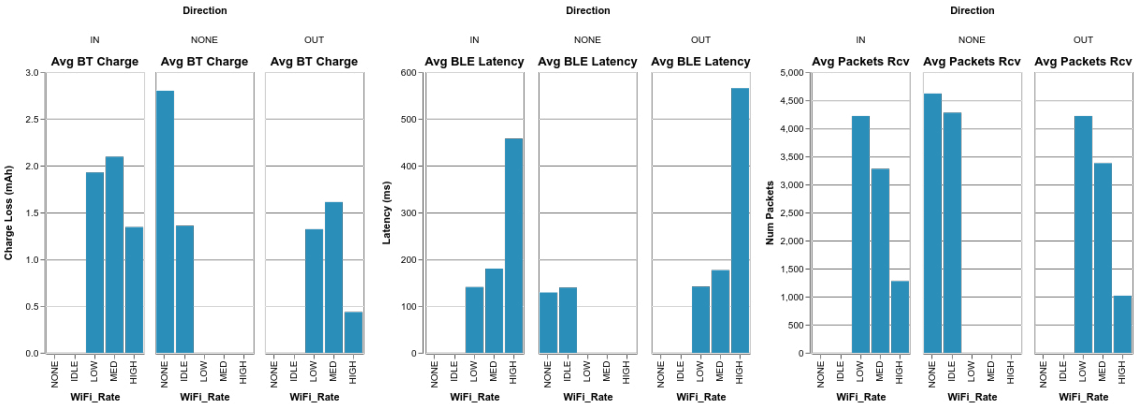

Fig. 5. Energy consumption for the scanner *(left)*, Discovery latency *(middle)*, and number of received packets *(right)* for Wi-Fi IDLE (associated with AP, no traffic), LOW (1 Kbps), MED (512 Kbps), and HIGH (2 Mbps).

6. Discussion

6.1. Discovery Latency

The discovery latency illustrates interesting behavior on the devices studied. First of all, the fastest advertising interval leads to a lower than expected advertising latency for when the scanner is set to BALANCED and LOW_POWER. Secondly, the increase in the standard deviation with the increasing advertising intervals hints that greater liberties are being taken with scheduling the advertising operation on the radio. This result echoes that of Sikora et al. [18] in which the advertisement interval, as measured by a device scanning a single channel at all times, has a higher standard deviation for lower power advertisement settings.

6.2. Energy Consumption

The energy consumption data suggests that the estimation of Bluetooth energy consumption in the Android bug report may not be as accurate as initially assumed. Further investigation could validate the accuracy of this feature of the bug report although it is possible that Bluetooth operations are being carried out by the Android operating system in the background. The energy consumption pattern during communication is intriguing because of the complementary structure to the overall energy consumption. The most readily available explanation for this is that the total energy consumption is accounting for the processing of scan requests and responses. It is plausible that in particular configurations, the scanner spends more time processing the received packets rather than simply listening. The high energy point for the advertiser could be when it is spending a lot of time listening for scan requests and issuing scan responses. Even though transmitting draws more current than receiving according to the data sheet of the WCN3680B/WCN3660B chip found in the Moto g5 Plus, the amount of time spent receiving can greatly surpass the amount of time spent transmitting—leading to greater energy expenditure.

6.3. Interference

The Wi-Fi interference affects the advertiser and scanner asymmetrically. This makes sense given that the amount of time it takes to send an advertisement is very small (N bytes $*8 * 10^{-6}$). From a scheduling point of view, there is plenty of time on the antenna available for Wi-Fi operations. For the scanner, the scheduling of the antenna for Wi-Fi operations appears to kick in as soon as the Wi-Fi is turned on as there is an immediate drop off in energy attributed to the Bluetooth (which we can think of as a proxy for the amount of time allocated for Bluetooth on the antenna). The impact of Wi-Fi traffic is somewhat lessened for the scanner when the traffic is directed into the device. This suggests that it may be easier to handle time sharing on the multi-radio SoC when the operation is the same (receive).

6.4. Limitations

This study faces a few limiting factors:

- The method of measuring energy consumption is potentially flawed, as illustrated through the negative charge consumption.
- The developed app is not representative of a real-world use case as it relies on energy draining features.
- We did not try to adapt the discovery latency algorithm from Kindt et al. to take multiple channels into account.
- The Wi-Fi interference tests do not specifically separate the effects of noise on the channel from the effects of time sharing the antenna.

7. Conclusions and Future Work

In this paper we evaluated the performance of BLE advertising and scanning (connection-less communication) on an Android smartphone and compared the results to models that have previously only been validated against dedicated BLE SoCs. We found that while the performance of BLE on Android with respect to the discovery latency does deviate from the expectations of the models, this is not always for the worse. Further, the average current attributed to the Bluetooth operations does exceed the expectations of the models. The effect of Wi-Fi interference in the form of UDP packets directed either into or out of the scanning and advertising devices at different rates had drastically different effects depending on the BLE role. In the future, we will investigate the effect of concurrent Bluetooth Classic communication during advertising and scanning as well as performance metrics for connection-oriented communication over GATT. We hope to apply lessons learned from these investigations toward the development of a multi-hop network that is deployable on common Bluetooth technology.

References

[1] Bluetooth SIG, a. Bluetooth market update 2018. URL: https://www.bluetooth.com/markets/market-report.
[2] Bluetooth SIG, b. Specification of the bluetooth system: Version 4.2.
[3] Bronzi, W., Frank, R., Castignani, G., Engel, T., 2016. Bluetooth low energy performance and robustness analysis for inter-vehicular communications. Ad Hoc Networks 37, 76–86.
[4] Perez-Diaz de Cerio, D., Hernndez, ., Valenzuela, J.L., Valdovinos, A., 2017. Analytical and experimental performance evaluation of BLE neighbor discovery process including non-idealities of real chipsets. Sensors 17. doi:10.3390/s17030499.
[5] Czurak, P., Maj, C., Szermer, M., Zabierowski, W., 2018. Impact of bluetooth low energy on energy consumption in android os, in: 2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH), IEEE. pp. 255–258.
[6] Dementyev, A., Hodges, S., Taylor, S., Smith, J., 2013. Power consumption analysis of bluetooth low energy, ZigBee and ANT sensor nodes in a cyclic sleep scenario, in: 2013 IEEE International Wireless Symposium, IWS 2013, pp. 1–4. doi:10.1109/IEEE-IWS.2013.6616827.
[7] Fürst, J., Chen, K., Kim, H.S., Bonnet, P., 2018. Evaluating bluetooth low energy for iot, in: 2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench), IEEE. pp. 1–6.
[8] Giovanelli, D., Milosevic, B., Farella, E., 2015. Bluetooth low energy for data streaming: Application-level analysis and recommendation, in: 2015 6th International Workshop on Advances in Sensors and Interfaces (IWASI), pp. 216–221. doi:10.1109/IWASI.2015.7184945.
[9] Gutschke, 2016. Adb fastboot tools. https://github.com/gutschke/adb-fastboot.
[10] Kindt, P., Yunge, D., Diemer, R., Chakraborty, S., 2014. Precise energy modeling for the bluetooth low energy protocol arXiv:1403.2919.
[11] Kindt, P.H., 2017. Pi-latencycomp - neighbor discovery in ble-like protocols. https://www.codeocean.com/. doi:https://doi.org/10.24433/CO.fec70c60-c265-4eea-9e37-8f7222ec5c92.
[12] Kindt, P.H., Saur, M., Balszun, M., Chakraborty, S., 2018. Neighbor discovery latency in BLE-like protocols. IEEE Transactions on Mobile Computing 17, 617–631. doi:10.1109/TMC.2017.2737008.
[13] Liu, J., Chen, C., Ma, Y., 2012. Modeling and performance analysis of device discovery in bluetooth low energy networks, in: 2012 IEEE Global Communications Conference (GLOBECOM), pp. 1538–1543. doi:10.1109/GLOCOM.2012.6503332.
[14] Liu, J., Chen, C., Ma, Y., Xu, Y., 2013. Energy analysis of device discovery for bluetooth low energy, in: 2013 IEEE 78th Vehicular Technology Conference (VTC Fall), IEEE. pp. 1–5.
[15] Mikhaylov, K., Plevritakis, N., Tervonen, J., Mikhaylov, K., Plevritakis, N., Tervonen, J., 2013. Performance analysis and comparison of bluetooth low energy with IEEE 802.15.4 and SimpliciTI. Journal of Sensor and Actuator Networks 2, 589–613. doi:10.3390/jsan2030589.
[16] MVP, 2017. Usb hub per-port power control. https://github.com/mvp/uhubctl.
[17] Radhakrishnan, M., Misra, A., Balan, R.K., Lee, Y., 2015. Smartphones and ble services: Empirical insights, in: 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, IEEE. pp. 226–234.
[18] Sikora, A., Krzysztoń, M., Marks, M., 2018. Application of bluetooth low energy protocol for communication in mobile networks, in: 2018 International Conference on Military Communications and Information Systems (ICMCIS), IEEE. pp. 1–6.