Will Dynamic Spectrum Access Drain My Battery?

Przemysław Pawelczak  p.pawelczak@tudelft.nl
Nihan Çiček           n.cicek@tudelft.nl
Niels Brouwers        n.brouwers@tudelft.nl
Koen Langendoen       k.g.langendoen@tudelft.nl
Will Dynamic Spectrum Access Drain my Battery?

Przemysław Pawelczak, Nihan Çicek, Niels Brouwers, and Koen Langendoen
Department of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
Email: {p.pawelczak, n.cicek, n.brouwers, k.g.langendoen}@tudelft.nl

Abstract—To satisfy the ever-increasing demand for wireless bandwidth, Dynamic/Opportunistic Spectrum Access (DSA/OSA) will become increasingly important in the years to come. Since most of the wireless traffic is generated by smartphones and other mobile devices, it is vital to determine the energy footprint of DSA/OSA, an aspect that has so far been ignored by the research community. To remedy this void, we present an extensive experimental evaluation of both spectrum sensing (using an Android-based low-cost Software Defined Radio, ‘RLT-SDR’) and database querying (using operational white space databases by Google, SpectrumBridge, and Microsoft Research). The resulting insight into the basic (access) delay and energy consumption of these competing techniques has driven our subsequent modeling and analysis of long-term DSA/OSA with a mobile secondary user roaming in-between various fixed primary users. We conclude that—despite the belief that spectrum sensing is energy hungry—cellular-based WSDB querying is many times more demanding; only querying over IEEE 802.11 does indeed consume less energy than spectrum sensing.

I. INTRODUCTION

It is a truism to say that the policy of radio frequency management leaves parts of the spectrum heavily utilized [1] Sec. IV, while keeping other spectrum bands virtually empty [2] Fig. 11. Dynamic/Opportunistic Spectrum Access (DSA/OSA) (in its many flavors) is thus a strong candidate to alleviate the “spectrum crunch” [3]. An essential element of DSA/OSA is a (spectrum) white space detector, implemented either locally on the device through spectrum sensing, or through access to a central repository, i.e. White Space Database (WSDB). However, while spectrum sensing attracted a great level of interest from the academic community, it has not been considered for first DSA/OSA trials for TV white space access. Spectrum sensing lacked the guarantees for channel availability [4] p. 9, and instead, WSDBs were used, e.g. [5], [6].

Interestingly, the use of WSDBs goes far beyond aiding in accessing TV white spaces. For example, spectrum databases are also considered for helping in regular spectrum sensing for radar activity detection (in L, S, C bands) [7], refer also to US government plans for 3.5 GHz Spectrum Access System [8]. More importantly, the use of WSDBs becomes increasingly relevant with the advent of Licensed Shared Access (LSA) [9]. Also, in Program Making and Special Events (PMSE) [10] pp. 23–24, where many independent stakeholders compete for the common spectrum for wireless video links and wireless microphones, WSDB would immensely automate spectrum allocation.

A. Problem Statement

Intuitively, and as [11] notes “spectrum sensing is expensive—in cost, energy consumption and complexity of the circuitry.” However, to the best of our knowledge there have been no studies thus far that would confirm this observation—especially from the energy consumption perspective. Verifying this statement (in most realistic conditions, preferably using contemporary smartphones) becomes crucial as one notices that WSDB querying must be frequent to minimize interference and every WSDB access costs energy of battery-operated secondary user (SU) devices. As soon as SU device roams on a large swaths of area, frequent sensing and/or querying becomes inevitable[11]. In addition, for PMSE we speculate that querying frequency to be at least in the order of seconds—refer also to [12, Fig. 7] for typical query rates in non-LSA/PMSE scenarios. In other words, the energy cost of WSDB access by end devices is unknown and highly relevant to investigate. This paper will try to shed a light on this aspect.

B. List of Important Contributions

In this paper we make the following advancements.

1) Accurate measurement of white space detection: Based on an extensive, two weeks spanning, measurement campaign we have characterized the WSDB access energy cost for an Android-based smartphone platform for (i) two internet access techniques: IEEE 802.11-based and cellular-based, (ii) three European cellular network operators, and (iii) three US-based commercial WSDB implementations. In addition, we have compared the WSDB energy querying cost with local spectrum sensing based on software defined radio (SDR)—connected to the same embedded platform;

2) Analytical modeling of of white space detection energy consumption: Based on our experiments, we have proposed a simple analytical energy consumption model for WSDB access using cellular-based Internet access, as well as local smartphone-based SDR-enabled sensing. We conclude that (i) [contrary to the intuition] considering energy consumed by SU’s smartphone WSDB-based white

This work has been supported by the Dutch Technology Foundation STW under contract 12491.
space detection might be either on par with SDR-based sensing, i.e. for two cellular operators amassing to \( \approx 6J \) for both fully USB connected SDR-sensor and WSDB querying, or for other network operator \( \approx 3.5 \times 3.5 \) times more energy-hungry compared to SDR-based sensing; and (ii) WSDBs are slower by orders of magnitude compared to Internet search services, differing significantly among themselves in terms of response time (in extreme case difference reaching \( \approx 5 \) seconds)—with their response times being query location-dependent.

The rest of the paper is organized as follows. Section [II] reviews the related work, while Section [III] introduces our experimental setup. Experimental results are presented in Section [IV] while subsequent analyses are presented in Section [V]. An analytical case study considering total energy consumption through roaming is given in Section [VI]. The paper concludes in Section [VII] preceded by discussion in Section [VIII].

II. BACKGROUND AND RELATED WORK

A. WSDBs and White Space Estimation

1) Existing Commercial WSDBs: Many commercial WSDBs already offer their services, including (in the US): (i) Google [6], (ii) Microsoft Research, Redmond, WA, USA [13], denoted later as MSR (also described in [14]), (iii) SpectrumBridge [5], and (iv) University of Washington [15]. Except for University of Washington’s WSDBs, all above databases offer open application programming interfaces (APIs) allowing for non-commercial experimentation (see for example the research of [12] based on SpectrumBridge WSDB querying). It is thus logical to use a set of them in our experiments.

2) WSDB Performance Evaluation: A very preliminary consideration of TV WSDB has been presented in [16]. One of the first complete local, sensing-free, WSDB implementation and evaluation is [14] where a set of different performance metrics (e.g. response time, database update time, white space computation time) have been presented. Indoor TV white space exploitation based on local sensors reporting to a WSDB has been evaluated in [11]. A very recent evaluation of LSA using an incumbent WSDB is presented in [17]. All in all, considering the existing WSDBs presented in Section II-A1 the knowledge on various WSDB performances is fragmented and comparative performance studies are missing. Most importantly, the energy querying cost imposed on the SU end device has never been considered in these studies.

3) WSDB-based White Space Estimation Process: A separate stream of research considers improving the white space estimation process of WSDBs itself. For a recent discussion on that topic we refer to [12], [18] (and the references therein).

Logically, local spectrum sensing is conceptually entangled with WSDB querying. Relevant results considering the implementation of spectrum sensing algorithms aiding in white space selection are e.g. [19] Sec. 3, with the respective spectrum sensing algorithm of [20] Sec. III-A) based on off-the-shelf (non-low power) SDR platform. Nevertheless, no receiver operating characteristics of the detector have been shown in [19], [20], nor the actual cost of local sensing in terms of consumed energy.

4) Interaction of Mobile Devices with WSDBs: In the literature, there are several works that focus on the performance of applications in mobile devices using different Internet access techniques. For instance, in [21] by conducting massively crowdsourced (\( \approx 30,000 \) users in total) measurements on US carrier’s UMTS/HSPA and EVDO Radio Access Techniques (RATs), delay characteristics were measured [21] Figs. 4–7 (among others) of accessing popular Web services. However, no WSDBs were considered as service of interest therein.

B. Energy Profiling of Wireless Embedded Devices

1) Energy Cost of Internet Access Use: Here we focus on systems-related research pertaining to this topic. One of the first such studies can be found in [22]. Therein, energy consumption measurements in three networking technologies, (i) UMTS/WCDMA, (ii) GSM/EDGE/GPRS and (iii) IEEE 802.11b, are performed and it was concluded that (i) and (ii) have a significant tail energy overhead. A more detailed investigation of this overhead (including state machine modeling), i.e. characterization or RRC in 3GPP-based networks, is discussed in [23].

Power characteristics of LTE (post-3G) RAT are studied empirically with data collected from 20 LTE-enabled smartphones and compared with IEEE 802.11g and UMTS CDMA in [24], where LTE is found to be less power efficient than networks in comparison due to its long high power tail [24] Fig. 12.

A very recent work pertaining to energy consumption measurements and modeling of IEEE 802.11x and 3GPP-based systems (as it is performed in this paper) has been presented in [25]. Therein, energy traces from three mobile platforms were correlated with operating system logs to get high granularity information on the RAT connection/disconnection and RAT transmission/reception process [23] Sec. 2. Another recent, but less detailed process of RAT access energy cost of Web services access has been presented in [26]. In all the above mentioned works, energy profiling is based on popular web services only, thus energy cost of WSDB access has never been considered. Naturally, intrinsic features of WSDBs, e.g. large response delays [14] Fig. 13, ask for re-evaluation of energy profile studies.

2) Energy Consumption of Spectrum Sensing in Embedded Platforms: Finally, we remark that except for [27] we are not aware of studies evaluating smartphone-based spectrum sensing platforms. Therein, unfortunately (i) sensing cost has been evaluated by considering cost of all radio features of the sensor [27] Fig. 1 (which was a low-power ZigBee transceiver connected through a Secure Digital port), and (ii) evaluation was performed based on software-only energy measurement which is drawing energy from the smartphone itself. In addition, we remark that the only study considering connecting SDR with smartphones can be found in [28]. Nevertheless, the energy cost of such connection, or spectrum sensing implementation, was not considered.
III. EXPERIMENTAL SETUP

To realistically assess the energy costs associated with WSDB access we developed a purpose-built application that periodically executes a WSDB query. We chose to carry out experiments using a Samsung Galaxy Nexus (GT-I9250) smartphone running CyanogenMod 10.1.3 (i.e. a modified Android 4.2.2). Experiments were run under different WSDB/network operator/network types, while energy consumption measurements were done using a custom-built high resolution power meter hardware described first in [29] and built by [30]. In addition, we have performed local sensing on the same embedded platform using an external hardware SDR dongle. Energy consumption of the SDR-based sensing on the smartphone was also measured using the same hardware/software tool chain [29], [30].

A. Energy Consumption Measurement Platform [29], [30]

1) Energy Measurement Hardware: The smartphone was equipped with an external high-resolution custom built power meter called NEAT [29] as shown in Fig. 1. The fully portable, phone battery-independent, power meter was embedded into an enlarged back cover, capable of sampling battery voltage and current draw. Data is recorded onto a micro SD card. A summary of the NEAT hardware capabilities is presented in Table I. We note that NEAT schematics, firmware and layout files are provided upon request by contacting [30].

2) Energy Measurement Software: A logging application running on the smartphone records events from the Android kernel and user-space programs. Events are later overlaid onto the collected power trace, using synchronization points established through the hardware trigger. Fig. 1. The actual correlation of flanking and operating system events is described in [29] Sec. 4.1 and 5.3. The application that performs the WSDB querying and spectrum sensing (see Section III-B1) emits events whenever operations are started and completed, so that the energy consumption of an operation to the area under the curve between those two points can be taken. The screen was turned off during the experiments.

B. White Space Spectrum Database Access

1) WSDB Querying Process: For the WSDB querying experiment we have developed an Android OS application, allowing the user to manually select (i) which WSDB to query ((a) Google [6], (b) MSR [13], or (c) SpectrumBridge [5]), (ii) with what frequency to query ((a) 10 s, (b) 30 s, (c) 1 m, (d) 2 m, (c) 5 m), (iii) and what location information to pass to WSDB. Two selected predefined locations were (i) Scipio, OH, USA, and (ii) Manhattan, NY, USA, while coordinates of both locations are given in Fig. 2. Those allowed to test for the effect of calculation of white space availability depending on two distinct query delay profiles, as it will become apparent in Section IV-B1.

We need to emphasize that for each query we resided outside the US, i.e. in the Netherlands. We feel nevertheless that the results provided here are fully representative for the US-located experiments—refer again to Section IV-B1.

a) Google WSDB Query Format: The IETF PAWS-compliant parameters passed to Google’s WSDB are summarized in Table II. Note that due to a non-commercial API license we possessed, a cap of 1000 queries per day with a maximum query rate of one per second, has been imposed by Google.

b) MSR WSDB Query Format: The parameters that were passed in a MSR WSDB request are summarized in Table II.


TABLE I NEAT HARDWARE PARAMETERS (TAKEN FROM [29])

<table>
<thead>
<tr>
<th>Feature</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casing</td>
<td>3D-printed, fitted on top of phone’s battery</td>
</tr>
<tr>
<td>Micro-controller</td>
<td>STM32F373xx (32-bit ARM Cortex-M4 architecture)</td>
</tr>
<tr>
<td>NEAT battery supply</td>
<td>1.5 V, 50-600 mA, 12.2 mA average current draw (10.5-14h of continuous operation)</td>
</tr>
<tr>
<td>Current measurement</td>
<td>MAX9922 current probe, 75 mΩ shunt resistor</td>
</tr>
<tr>
<td>Sampling and range</td>
<td>2 kHz at [-0.5,2] A with 38.15 μA resolution</td>
</tr>
<tr>
<td>USB port</td>
<td>NEAT battery charge, calibration, RTL control</td>
</tr>
<tr>
<td>Data storage</td>
<td>Micro SD card, e.g. ≤98h measurement data at 6kHz rate for 4GB card</td>
</tr>
<tr>
<td>Dimensions</td>
<td>39.3 x 16.3 x 3.3 mm</td>
</tr>
</tbody>
</table>

Fig. 1. Measurement setup used in the experiments. Important components: S1: 3D-printed chassis, S1a: battery supporting NEAT, S1b: NEAT hardware, S2a: antenna connector, S2b: Realtek RTL2832U-based TV dongle (“RTL-SDR”), see Section III-C1. S2c: USB 2.0 OTG mode cable, S3: antenna (connected through S2a), F1: external battery connector, F2: micro USB port (NEAT battery charge), F3: status LED, F4: I²C port, F5: reset button, F6: Phone-to-NEAT connectors (BAT+ connection to smartphone battery ports; TRIG: connection to smartphone vibrator pad, LOAD: connection to shunt resistor, GND: ground), F7: Micro SD slot, B1: STM32F373xx microcontroller, B2: MAX9922 current-sense amplifier.

TABLE II PARAMETERS PASSED TO ALL WSDB IN QUERY REQUEST

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Google</th>
<th>MSR</th>
<th>SpectrumBridge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>PAWS 1.0</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Device type</td>
<td>Mod. 1</td>
<td>NA</td>
<td>Mod. 2 (40 mW)</td>
</tr>
<tr>
<td>Antenna</td>
<td>30 m AGL</td>
<td>NA</td>
<td>30 m</td>
</tr>
<tr>
<td>Propagation</td>
<td>NI</td>
<td>Longley-Rice</td>
<td>NI</td>
</tr>
<tr>
<td>Calling</td>
<td>NI</td>
<td>-114 dBm</td>
<td>NI</td>
</tr>
</tbody>
</table>


2Although in our query we also pass other parameters, they are ignored by Google. First, operational frequency ranges of the secondary device we provide (i.e., [800, 850] MHz and [900, 950] MHz), due to a Google bug we found (see https://groups.google.com/forum/#topic/google-spectrum-db-discusses/14sdoxgQvQe) are ignored and full spectrum response is given. Also, despite passing the secondary user antenna height in the query (30 m), this also has no effect on white space calculation, as confirmed by Google.
and provided in the source code of our application [31]. Note that in contrary to the Google WSDB, there was no limit on the number of queries. Also, it is important to state that MSR’s WSDB is an experimental, non-US regulation compliant, system.

c) SpectrumBridge WSDB Query Format: Via courtesy of SpectrumBridge we have obtained the access to the demonstration version of company’s WSDB, which according to the company is ‘an exact replica of the certified [US-based regulator] system’. The query parameters passed in the experiments are described in Table II. Note that for the SpectrumBridge WSDB we used a different white space device type than for Google WSDB, i.e. Mode 2 (with 40 mW transmission power limit). Nevertheless, Mode 1 and Mode 2 devices are mobile and both are allowed to query WSDBs for free channels.

C. SDR Local Spectrum Sensing Platform

1) Spectrum Sensing Hardware: Investigating the SDR capabilities of modern smartphone platforms, [28] concluded that “the most difficult hurdle in realizing the SDR app approach is the open radio front [end]”, suggesting an external front-end as a workaround. In our experiments we use therefore an inexpensive USB DVB-T dongle based on a Realtek RTL2832U DVB-T OFDM demodulator [32]. The demodulator is able to directly output raw 8-bit I/Q samples at a maximum (stable) 2.56 MS/s, enabling inexpensive SDR experimentation [33]. In our experiments we connect the dongle to the USB port of the smartphone (see Fig. 1).

Table III

<table>
<thead>
<tr>
<th>Tuner</th>
<th>Sampling Rate (MHz)</th>
<th>Power Draw (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitpower FC0013</td>
<td>— (idle mode)</td>
<td>321.72</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>609.66</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>655.24</td>
</tr>
<tr>
<td>Rafael Micro R820T</td>
<td>— (idle mode)</td>
<td>332.09</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1222.26</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1243.96</td>
</tr>
</tbody>
</table>

TABLE III

POWER CONSUMPTION OF TWO RTL2832U-BASED DONGLES, MEASURED USING A MONSOON POWER METER [34]

Idle mode: device plugged in but no sampling; Measurement type: USB pass-through mode.

RTL-SDR Tuner Selection: Several different RTL2832U-based dongles are available, and we found that choice of tuner chip has an influence on the power consumption. Table III shows the power consumption of two dongles containing different tuners. Measurements we carried out using a Monsoon power monitor in USB-passsthrough mode [34]. We observe that the R820T-based dongle draws nearly twice as much current as the FC0013-based one. For the rest of this paper we therefore proceed with more energy-efficient FC0013 tuner.

Physical Connection: We tested two connections between smartphone and RTL-SDR. In first, most straightforward way (denoted as OTG), we used a so-called ‘USB on the go’ (OTG) cable, which forces the phone to act as a USB host. In this setup, smartphone supplies +5 V on the USB port and powers RTL-SDR. However, the device’s battery voltage is significantly lower (3.6–4.2 V depending on charge) and will need to stepped up—a process causing significant power draw.

To avoid this overhead, we measured a second setup (denoted as BTR) in which the RTL-SDR is powered directly from the phone’s internal battery. To accomplish this we modified the USB host driver in the Android kernel so that no power is supplied on the USB bus when the dongle is plugged in. We then spliced an OTG cable and connected its voltage and ground lines to that of the phone battery, while leaving the USB D+/D- lines intact. We found no ill effects for RTL-SDR from being run at a slightly lower voltage.

2) Spectrum Sensing Software: To experiment with the SDR spectrum sensing on the smartphone, we have adapted rtl_power, a spectrum scanner provided with RTL-SDR software [33], to sweep through all UHF TV channels in the Netherlands: 48 channels, 8 MHz wide, with no spacing, starting at 470 MHz.

a) Spectrum Sensing Implementation: Observing that UHF digital TV channels have well specified masks, we can detect the presence of TV signal with one tune by tuning to a center frequency at the edge of two adjacent channels and compare it against the threshold $\gamma$. We tune to a 1 MHz channel and consider 256 FFT with a rectangular window, with 8.2 ms integration time translating to $2^{14}$ I/Q samples buffer which is passed to FFT only when full. Before new frequency tune, a 5 ms delay required for I/Q buffer flush is imposed. The reader is referred to the source file of the scanner [51] for more details of our sensing implementation. Note that evaluation of spectrum sensing for other FFT sizes and sampling rates is straightforward and left for future investigation.

b) Incumbent Detection: We emphasize that we do not perform an actual incumbent detection in this study (including wireless microphone detection), i.e. we do not optimize the threshold $\gamma$ for a certain detectability. The focus here on the energy profiling of a fundamental operation leading to incumbent detection, i.e. FFT, and radio front end tuning.

D. Data Collection

WSDB querying experiments were collected via three Dutch mobile operators: (i) T-Mobile, (ii) Vodafone and (iii) KPN, and through IEEE 802.11 Eduroam network. In total 10549 query results at a fixed indoor location were collected, within two weeks, of which 3695 for T-Mobile, 3143 for Vodafone, 935 for KPN and 2975 for Eduroam. The WSDB querying interval is chosen to be 30 s for T-Mobile, 3143 for Vodafone, 935 for KPN and 2975 for Eduroam. The WSDB querying interval is chosen to be 30 s for T-Mobile, 3143 for Vodafone, 935 for KPN and 2975 for Eduroam. The WSDB querying interval is chosen to be 30 s for T-Mobile, 3143 for Vodafone, 935 for KPN and 2975 for Eduroam.

E. Replicability of the Results

For replicability of the results, latest versions of all application source codes (WSDB query app, RTL-SDR sensing implementation, NEAT processing files, MATLAB processing scripts) and traces used in this paper are available upon request or via http://www.es.ewi.tudelft.nl/reports/support_files/tdsdbsp-tech-report-source-files-2014.zip.

1Different querying intervals were chosen for each operator experimentally due to large differences in the so-called tail energy. This will be discussed further in Section IV-B.2.
Fig. 2. MATLAB boxplot of delay and energy consumption of Smartphone-based RTL-SDR TV UHF channel spectrum sensing: connection (BTR), {red ‘x’ outlier sign and circle marker}; connection (OTG), {red ‘x’ outlier sign and dash marker}; connection (SB), see Section IV-A for details.

IV. EXPERIMENTAL RESULTS

A. Spectrum Sensing Energy Profile: Linear and Predictable

Result of the experiment is presented in Fig. 2. Looking at the scanning length results, Fig. 2(a) delay increases linearly with the number of channels to be scanned and its independent of the RTL-SDR/Smartphone connection type.

Observing energy consumption results, Fig. 2(b) we see that connection (BTR) consumes significantly more power then connection (OTG). Furthermore, the energy cost of sensing increases linearly with number of channels scanned and is highly predictable, with very little variation. We then propose the following intuitive model relating consumed energy with number of re-tunings done, n, as

$$E(n) = a_n + b_n \forall n \in \mathbb{Z}^+,$$  \hspace{1cm} (1)

where $a_n$ is the energy consumption of a single tune, $b_n$ is the cost of running spectrum sensing circuitry and $t_s$ is the time of single re-tune. Using MATLAB’s R2013a polyfit function the values of $a_n$ and $b_n$ are 0.2 J and 0.84 J for (OTG) and 0.1 J and 0.47 J for (BTR), respectively.

B. WSDB Querying Energy Profile

1) General Observations: Before proceeding with smartphone-based queries, we profiled all WSDBs using cURL v7.30.0 from a stationary PC connected to Eduroam.

a) WSDB Response Delay Distribution—Location Varying and WSDB-dependent: We have first measured the query response time, i.e. a total time from sending a cURL request to a full WSDB reply, as a function of queried for location. Results are presented in Fig. 3. We observe that for all WSDBs query response times are location dependent. We speculate that (i) the flatter the terrain, the faster the response from a database (ii) and/or the more surrounding TV transmitters from the queried location, the more time it takes to calculate the white space, nevertheless we have no definite explanation for such behavior. Then, for all WSDB implementations, response times compared to web server response times of the respective organizations (BG, SB and GL as in Fig. 3) the response times are much larger (in extreme case of MSR larger than 1 second).

b) WSDB Response Size Distribution—Small Messages, Large inter-WSDB Differences: In the second experiment, as in the case of Fig. 2 we have queried all WSDBs in a straight line from US west coast to US east coast (LA location, see Fig. 3) to US east coast (CB location, see Fig. 3) and plotted WSDB expected response time distribution for all tested databases in Fig. 4. We immediately observe huge differences between delay profiles of each databases, Google being most delay efficient, while MSR extremely slow and variable. Interestingly, MSR’s and SpectrumBridge delay probability density function (PDF) is non-regular (note a bimodal-like shape of the PDF). Also, note that SpectrumBridge delay profile is almost twice as large as Google’s.

c) WSDB Response Error Rate—Some WSDBs are Erroneous: Finally, it is important to remark that in our experi-
imments Google provided response for every query we sent, while MSR had a failure response rate of 8.55\% (with a variance of 0.06) for 20 sets of 100 queries randomly selected between LA and CB location as given in Fig. 4, while for SpectrumBridge error response rate was 2\% (with a variance of 0.02). In the following sections (unless otherwise stated) we thus proceed with analyzing Google and SpectrumBridge WSDB querying process only.

2) Energy Cost of WSDB Querying: We first present results on WSDB query excluding tail energy and later excluding wake-up and query energy.

a) Query Process over Various Access Networks—Try Always Query Through IEEE 802.11: Results are presented in Fig. 6 considering for cellular access only energy cost for smartphone’s radio front end ramp-up from sleep and actual WSDB querying. We confirm that IEEE 802.11-based access is far more energy efficient than cellular-based. Not only it is faster to receive a WSDB reply for all WSDB/location combinations, but also it by one Joule more efficient than cellular-based WSDB query in the worst case. Then, all cellular operators tend to be clustered one on top of another. The only operator that has slightly higher energy use is KPN.

This experiment confirms further the observation from Fig. 4 that Google WSDB is faster than SpectrumBridge, compare Fig. 6(a) and Fig. 6(b) with Fig. 6(c) and Fig. 6(d) however the reply delay between NYC and Ohio locations discussed in Section IV-B1a vanishes due to RAT response variability.

b) Investigation of Tail Energy—Large Overhead to WSDB Query, Operator-dependent: Querying any public API over the Internet via cellular network involves two main sources of energy draw: (i) the main central processing unit (CPU), and (ii) the radio front end. As discussed in Section IV-B2a IEEE 802.11-based query the energy cost is simply equal to the total amount of energy spent between the start- and completion of the query. For cellular networks, however, the so-called tail energy has to be taken into account as well, i.e. the energy spent by the radio front end as it remains in a high power state after the query has completed.

As seen in Fig. 7 the amount of energy spent on such tail depends on (i) the length of the Radio Resource Control (RRC) time-outs set by the network operator, and (ii) the power draw of the modem in the Dedicated Channel (DCH) and Forward Access Channel (FACH) state The state transitions from DCH to FACH, and FACH to idle can be seen in Figure 7 as discrete power drops.

Flanking Tail Energy in Measurement Trace: Because these state transition happen when the main CPU is in suspend mode, they do not appear in the event log. We therefore extract the locations in the trace where these transitions occur as follows. We first apply a median filter with a window size of 50 ms to the power trace to remove noise and power spikes. An error function is computed over a sliding window, representing the sum-of-squares distance to a step function centered on the current sample, with a left-hand side power level corresponding to source state (e.g. \( \approx 520 \) mW for the DCH state), and a right-hand side power level corresponding to the target state (e.g. \( \approx 19 \) mW for idle). Transition events are inserted at local minima that pass a lower threshold. Parameters such as window size and detection threshold have been found empirically.

Extraction of Tail Energy Data: We selected a single trace from each of the networks. Note that we are interested only

\footnote{The power consumption during FACH and DCH tails was already investigated in \[29\]. A detailed discussion on the trade-offs associated with timings of RRC states can be found in \[25\].}

\footnote{Detection could be improved by manually annotating a large number of transitions and using machine learning techniques to find the best parameters. We leave this for future work.}
in the power consumption of the radio front end, and not the CPU. We therefore compute the energy consumed in the DCH state as the power level in that state multiplied by the time spent in that state, rather than taking the area under the curve, as that would include the power draw of the CPU during the short time between finishing the query and going back into suspend mode. Because the detection algorithm is imperfect, and might occasionally miss particularly noisy transitions, we removed the outliers using MATLAB boxplot function. Moreover, it is worth noting that the T-Mobile network transitions from DCH directly back to idle without first passing through FACH.

**Findings**: A boxplot summarizing the energy consumption for all performed experiments is presented in Fig. 8—note that the summary is performed for European setup (as RTL-SDR senses only these channels). First, looking at the median energy consumed in Fig. 8(a) (all energy cost except for tail energy), SpectrumBridge incurs ≈0.1J more on the smartphone than Google WSDB, demonstrating that just a selection of WSDB has an effect on smartphone energy consumption.

Then, looking at tail energy—for KPN it is almost four times larger than for the other two operators, see Fig. 8(b) which contributes significantly to the overall querying cost, see Fig. 8(c) where all experiment results are plotted for a conclusive comparison. It is clearly seen that SDR-based sensing on smartphone is actually on par with all WSDB connections for all operators except for KPN for USB connection. For RTL-SDR in BTR mode, energy cost drops significantly, but its still larger than for Eduroam-based WSDB querying.

V. Modeling Energy Cost of WSDB Querying

A. Proposed Energy Consumption Model of WSDB Querying

We have fitted the measurements to a linear model as in Section IV-A and demonstrated in Fig. 6. Based on the fitting we propose the energy consumption model of cellular-based WSDB querying as

\[ E_q(t) = c_q t + a_q, \]  

where \( t \) is the querying delay and \( a_q \) and \( c_q \) are the ramp-up energy consumption of the smartphone (which takes a different value for different networks) and energy consumption due to received signal strength, respectively. Table IV summarizes the values for \( a_q \) and \( c_q \), together with the coefficient of determination \( R^2 \) test indicating the accuracy of model fitting. A good fit is observed for Google WSDB for all three cellular operators and Ohio location, \( R^2 > 0.9 \), while slightly less for NY location, minimum \( R^2 = 0.6 \). SpectrumBridge WSDB was fitted with less accuracy to a linear model, i.e. only one \( R^2 > 0.9 \). Considering cross-fitting of \([a_q, c_q]\) coefficient pair estimated from one WSDB/operator/location triple into another (refer to [31] for an accompanying MATLAB script) reveals that differences between parameters are small, justifying a one common set of \([a_q, c_q]\) for all scenarios in case of back-of-the-envelope calculations.

Finally, we observe that \( R^2 \) values are very small for Eduroam. This is the reason we do not propose a model for IEEE 802.11-based networks and we leave its modeling for future study. Summarizing, we will later use \( R^2 \) in finding the total energy spent on querying in roaming in Section VI.

B. WSDB Response Time Distribution—Long-Tail Presence

To find the PDF for the WSDB response delay we have used the Kolmogorov-Smirnov (K-S) test with 99% critical value. We have tested the querying delay time data for seven distributions, namely (i) exponential, (ii) log-normal, (iii) gamma, (iv) gaussian, (v) Rayleigh, (vi) uniform and (vii) Weibull. Considering the output of the K-S test, log-normal, gamma and Weibull distributions are found to fit querying delay time for each scenario. Among these three distributions, we have chosen Weibull distribution as an example distribution and gave its shape and scale parameters, \( \lambda \) and \( k \), respectively, for each scenario in Table V to help the readers visualize the distribution. In addition, mean and variance of the querying delay time for each scenario are also listed in Table V.

VI. Energy Consumption of White Space Detection in a Mobile Scenario: A Case Study

Finally, we demonstrate the usefulness of our proposed models of (1) and (2). We perform this by introducing a

### Table IV

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dataset</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
<th>( \lambda )</th>
<th>( k )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMB/GGL (NY, Ohio)</td>
<td>2.94, 2.91</td>
<td>0.12, 0.14</td>
<td>3.098, 3.082</td>
<td>8.26, 7.28</td>
<td></td>
</tr>
<tr>
<td>TMB/SBI (NY, Ohio)</td>
<td>3.20, 3.16</td>
<td>0.17, 0.19</td>
<td>3.398, 3.373</td>
<td>6.32, 5.44</td>
<td></td>
</tr>
<tr>
<td>TMB (all)</td>
<td>3.07</td>
<td>0.17</td>
<td>3.263</td>
<td>5.96</td>
<td></td>
</tr>
<tr>
<td>KPN/GGL (NY, Ohio)</td>
<td>3.07, 2.72</td>
<td>0.39, 0.12</td>
<td>3.324, 2.923</td>
<td>5.19, 6.18</td>
<td></td>
</tr>
<tr>
<td>KPN/SBI (NY, Ohio)</td>
<td>3.16, 2.94</td>
<td>0.16, 0.15</td>
<td>3.342, 3.117</td>
<td>7.72, 7.44</td>
<td></td>
</tr>
<tr>
<td>KPN (all)</td>
<td>2.98</td>
<td>0.26</td>
<td>3.197</td>
<td>6.01</td>
<td></td>
</tr>
<tr>
<td>VDF/GGL (NY, Ohio)</td>
<td>3.08, 2.98</td>
<td>0.16, 0.23</td>
<td>3.281, 3.205</td>
<td>6.32, 6.26</td>
<td></td>
</tr>
<tr>
<td>VDF/SBI (NY, Ohio)</td>
<td>2.93, 2.79</td>
<td>0.06, 0.06</td>
<td>3.054, 2.917</td>
<td>10.63, 9.86</td>
<td></td>
</tr>
<tr>
<td>VDF (all)</td>
<td>2.94</td>
<td>0.15</td>
<td>3.12</td>
<td>6.33</td>
<td></td>
</tr>
<tr>
<td>EDR/GGL (NY, Ohio)</td>
<td>1.02, 1.06</td>
<td>0.06, 0.04</td>
<td>1.130, 1.071</td>
<td>4.53, 1.16</td>
<td></td>
</tr>
<tr>
<td>EDR/SBI (NY, Ohio)</td>
<td>1.45, 1.25</td>
<td>0.01, 0.01</td>
<td>1.511, 1.312</td>
<td>9.65, 8.74</td>
<td></td>
</tr>
<tr>
<td>EDR (all)</td>
<td>0.26</td>
<td>0.29</td>
<td>1.058</td>
<td>1.51</td>
<td></td>
</tr>
<tr>
<td>GGL (all)</td>
<td>2.27</td>
<td>1.36</td>
<td>2.467</td>
<td>1.62</td>
<td></td>
</tr>
<tr>
<td>SBI (all)</td>
<td>2.61</td>
<td>0.65</td>
<td>2.903</td>
<td>3.75</td>
<td></td>
</tr>
</tbody>
</table>

### Table V

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( \mu )</th>
<th>( \sigma^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMB/GGL</td>
<td>NY, Ohio</td>
<td>2.56, 3.082</td>
</tr>
<tr>
<td>TMB/SBI</td>
<td>NY, Ohio</td>
<td>3.398, 3.373</td>
</tr>
<tr>
<td>TMB (all)</td>
<td>3.263</td>
<td></td>
</tr>
<tr>
<td>KPN/GGL</td>
<td>NY, Ohio</td>
<td>3.324, 2.923</td>
</tr>
<tr>
<td>KPN/SBI</td>
<td>NY, Ohio</td>
<td>3.342, 3.117</td>
</tr>
<tr>
<td>KPN (all)</td>
<td>3.197</td>
<td></td>
</tr>
<tr>
<td>VDF/GGL</td>
<td>NY, Ohio</td>
<td>3.281, 3.205</td>
</tr>
<tr>
<td>VDF/SBI</td>
<td>NY, Ohio</td>
<td>3.054, 2.917</td>
</tr>
<tr>
<td>VDF (all)</td>
<td>3.12</td>
<td></td>
</tr>
<tr>
<td>EDR/GGL</td>
<td>NY, Ohio</td>
<td>1.130, 1.071</td>
</tr>
<tr>
<td>EDR/SBI</td>
<td>NY, Ohio</td>
<td>1.511, 1.312</td>
</tr>
<tr>
<td>EDR (all)</td>
<td>1.058</td>
<td></td>
</tr>
<tr>
<td>GGL (all)</td>
<td>2.467</td>
<td></td>
</tr>
<tr>
<td>SBI (all)</td>
<td>2.903</td>
<td></td>
</tr>
</tbody>
</table>
formal example procedure to calculate the expected energy consumption due to white space detection during a large-area roaming of the SU of the spectrum—without the need for large scale mobility experiments.

A. System Model

We directly adapt a system model of [35, Sec. III-A]. We assume that the primary user (PU) distribution follows a Poisson process with the average density of $\rho$ PUs per m². SUs are mobile, moving with an average speed of $v$ m/s within an area encapsulating PUs. Each PU has a transmission range of $r$ and operates on the same channel. SU query WSDB with an uniform time interval of $T_q$. Based on the last WSDB query SU selects one PU channel for transmission. As SU moves, white space information becomes stale causing potential interference to PU. This, in-turn, imposes frequent WSDB re-querying on SU.

B. Analysis

1) Optimal Query Rate: First goal is to find a $T_q$ minimizing both interference to PU and energy consumption related with WSDB querying by decreasing querying frequency. Following [35, Sec. III-A], probability of inducing $T_q$'s of interference to PU is [35, (1)]

$$f_i(T_q) = 2\rho^2 v \left(1 - e^{-v r T_q}\right).$$

Then, we want to find $\min_{0<T_q\leq T_{\text{max}}}$ $f_i(T_q)/T_q$, where $T_{\text{max}} \triangleq d_{\text{max}}/v$ is the minimum query interval imposed, i.e. after SU moved $d_{\text{max}}$ m from the previous location it queried, e.g. $d_{\text{max}} = 100$ m following US regulation. We introduce the following lemma.

Lemma 1: The optimal sensing interval for the considered system model is given as

$$T^*_q = \min\{-r(W_m(-e^{-1}) + 1)v^{-1}, T_{\text{max}}\}, \forall m \in \mathbb{Z}$$

where $W_m(\cdot)$ is the Lambert W function of branch $m$.

Proof: Directly by calculating minimum trough $df_i(T_q)/T_q$.

It is important to remark that solution to (4) is a complex number. As a sidetone, we remark that [35, Proposition 2] is presumably incorrect, as the calculation step from [35, (11)] to [35, (12)] if faulty resulting in a wrong conclusion.

2) Energy Distribution: Second, we calculate the consumed energy due to white space search during time $H$ and we define $Q = H/T_q$, i.e. number of queries during $H$.

a) WSDB Query: We denote random variables representing energy consumption associated with WSDB querying and its querying delay as $J_q$ and $D_q$, respectively and define PDF of $D_q$ as $f_{D_q}(d)$. From (2) we can write $J_q = c_q D_q + a_q$. Then, using the transformation random variables such as $f_X(x) = a^{-1} f_Y \left(\frac{x}{a}\right)$ where $X$ and $Y$ are random variables with a relation $X = a Y + b$, $\forall a > 0$. Thus, $J_q$ can be defined with the PDF $f_{J_q}(J_q) = f_{D_q} \left(\frac{J_q - a_q}{c_q}\right) c_q^{-1}$, expected value $E[J_q] = c_q E[D_q] + a_q$ and variance $\text{Var}[J_q] = c_q^2 \text{Var}[D_q]$. Then, the energy consumed for querying within $H$ s is $U_q = Q J_q$, whose PDF is

$$f_{U_q}(u_q) = f_{J_q} \left(\frac{u_q}{Q}\right) Q^{-1} = f_{D_q} \left(\frac{u_q - a_q Q}{c_q Q}\right) \left(c_q Q^{-1}\right)$$

where $E[U_q] = Q (c E[D_q] + a_q)$, $\text{Var}[U_q] = (c Q)^2 \text{Var}[D_q]$. 

b) Local Spectrum Sensing: As the time it takes to sense the channels is deterministic, see (1), the amount of energy consumed for sensing within $H$ s is $U_s = E_s(n) Q$.

C. Numerical Example—DSA/OSA Might Drain (Smartphone’s) Battery

Let us consider the energy consumption of a mobile SU moving though an area consisting of PUs with PU constant transmission range $r = 200$ m, e.g., a IEEE 802.11-like access point. For a fair comparison of WSDB and local sensing, the number of channels scanned by SDR-based sensing is $n = 24$. The values of $a_s$ and $b_s$ are defined as in Section IV-A. In addition, the mean WSDB querying delay values are taken from Table IV while $a_q$ and $c_q$ are taken from Table IV. In the experiment we do not consider tail energy of WSDB query, thus the numerical results are underrepresentation of energy consumption for this case.

Considering typical human movement speeds, refer to Fig. 9 for a movement delay epoch of $H = 1$ h, we observe the total SU energy consumption where the optimal querying rate $T^*_q$ is chosen with $T_{\text{max}} \triangleq d_{\text{max}}/v$ for $d_{\text{max}} = 100$ m—a

---

**Fig. 8.** Boxplot of energy consumption for all considered experiments with white space detection techniques, network types, network operators and WSDBs. Abbreviations—EDR: Eduroam, GGL: Google, RTL: RTL-SDR, SBI: SpectrumBridge, TMB: T-Mobile, VDF: Vodafone.
current US-based regulation. We first observe that that this imposed constraint is strict as \( T_{\text{max}} \) as for all conditions considered in this example \( T_{\text{max}} \). Then as the speed of SU increases energy cost of white space search increases exponentially. Fig. 9 clearly shows that IEEE 802.11-based WSDb query is the most energy effective (however questionable system-wise considering very large SU speeds). Result in Fig. 9 proves that the battery of a smartphone, which typically stores around 20 kJ of energy (e.g. 19.62 kJ for iPhone 5s), will be drained within hours when queried for white spaces through WSDBs.

VII. DISCUSSION

Topic of energy efficiency of spectrum white spaces acquisition is by no means exhausted. Further points of consideration are as follows.

- **Larger smartphone set**: In WSDB context, other 3G-based and LTE-based smartphones should be measured;
- **Real human mobility traces**: Both analytical models should be applied to a large set of human mobility traces;
- **Optimization of SDR-based sensing**: Other SDR/Smartphone connection techniques reducing energy cost should be investigated, e.g. where an external micro-controller would steer the SDR to sleep;
- **Other WSDBs**: Further tests of other commercial WSDBs (including those outside the US) or non-TV white space databases (including those for LSA) are required.

Finally, numerous parallel lines of research that touch upon our findings. We list important ones here.

- **Local spectrum sensing and WSDB interaction**: Mobile SU-based sensing can improve the future WSDB replies by providing feedback to the WSDB. It is important to consider the overall energy cost of such operation;
- **WSDB response caching**: Caching WSDB replies might improve database responsiveness and in-turn reduce energy burden of SU;
- **Proactive WSDB replies**: Energy consumption reduction for the mobile SU terminal can be achieved further by providing feedback by WSDB for a given radius of expected SU path (instead of a single position).

VIII. CONCLUSIONS

We have performed a detailed comparative study of local spectrum sensing and white space database (WSDB) access on a smartphone and focused on energy/delay tradeoff. First, we have found large differences in WSDB quality for all databases tested. In addition, we demonstrate for the first time an Android-based low-cost Software Define Radio platform allowing to assess an energy detector-based white space search. Most importantly, we conclude that the cost of WSDB query is non-negligible and contrary to many beliefs, sometimes on par regarding energy cost with local spectrum sensing.

REFERENCES