Delft University of Technology
Master's Thesis in Electrical Engineering

Better Mobility Support for Radio Spectrum White Space-enabled Devices

Amjad Yousef Majid

embedded software

TU Delft Delft University of Technology

Better Mobility Support for Radio Spectrum White Space-enabled Devices

Master's Thesis in Electrical Engineering

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Amjad Yousef Majid

amjadyousefmajid@student.tudelft.nl

18th August 2015

Author
Amjad Yousef Majid (amjadyousefmajid@student.tudelft.nl)

Title
Better Mobility Support for Radio Spectrum White Space-enabled Devices

MSc presentation
18th August 2015

Graduation Committee

prof. dr. Koen Langendoen	Delft University of Technology
dr. Przemysław Pawełczak	Delft University of Technology
dr. ir. Gerard Janssen	Delft University of Technology

This work has papers submitted for possible publications in IEEE TMC, and IEEE Sensors.

Parts of this work appeared earlier in the TU Delft technical report "Will Dynamic Spectrum Access Drain my Battery?" Number: ES-2014-01.

Abstract

Dynamic Spectrum Access (DSA) is a new spectrum sharing paradigm that aims at increasing the utilization of radio frequencies as well as mitigating the spectrum scarcity problem. It allows Secondary Users (SUs) to access idle channels in the licensed spectrum while protecting the signals of Primary Users (PUs) from harmful interference. Although local-sensing can detect an empty radio frequency band, it does not provide a sufficient level of protection to a PU's transmission. Therefore, DSA currently relies on online geo-location databases, White Space Databases (WSDBs), to distribute information about the availability of radio spectrum white spaces. In this thesis, we make three contributions to the field of Dynamic Spectrum Access (DSA).

First, the access to WSDBs in terms of response message size and response time is profiled using the state-of-the-art WSDB query technique *Single-location WSDB query*. This comparative study helps us in understanding the differences between the WSDBs' performances and shows us the capabilities and limitations of the current query technique. Our conclusion from this part of the study is that the current WSDB query method has poor support to white space-enabled mobile devices (MWSD).

Second, based on our previous conclusion, we propose a new WSDB querying technique *multi-location WSDB query* that optimizes the access of MWSD to a WSDB by reducing the number of needed queries to cover a particular path and as a results the energy consumption of the query process is reduced. Furthermore, We show that *multi-location query* has a much quicker response time than *single-location query*.

We introduced the world's first algorithm called *Nuna* that predicts the next direction of a movement path, estimates the required size of the multi-location WSDB query, and queries a WSDB. *Nuna* helps MWSD to use *multi-location query* without they need to know their path in advance. Moreover, we demonstrated by example that *Nuna* halves the required number of queries to cover a particular path, improving the energy efficiency by 50%.

Finally, we shifted our focus from optimizing the access of MWSD to a WSDB to optimize the information of a WSDB by providing local sensing readings to a WSDB. Therefore, we developed an energy efficient Portable Spectrum Sensing Platform (PoSSP) that can obtain its GPS position, sense the spectrum, and send its data to an online server. We show that PoSSP is by 60% more energy efficient than similarly designed sensing platforms.

"Knowledge is the root of all good."
—Ali ibn Abi Talib

Preface

I stopped my previous research because I was not enjoying working on it. It was a difficult decision to stop after months of working and to start, from zero, looking for a new opportunity. However, I knew it was the right decision because to do a great job you should love what you do.

This current study could never have been completed without the support of others. First and foremost, I would like to express my sincere gratitude to my supervisor, dr. Przemysław Pawełczak, for offering me such a great research opportunity, guidance, constructive criticism and encouragement. Moreover, his suggestions have helped me in opening new directions in my research.

Secondly, I would like to thank Peter Stanforth, Luzango Pangani, Ranveer Chandra, Pasquale Cataldi and Heikki Kokkinen for providing white space databases' APIs for the SBI, CSI, MSR, NOM, and FIR systems, respectively. Furthermore, I would like to thank Noor Assad for the car and Ahmed Assad, Yousef Assad and Jawad Hamid for driving it to test the implementation of the Multi-Location WSDB query technique. My thanks also go out to Ivar in 't Veen, Ioannis Protonotarios and Niels Brouwers for their help and suggestions, and to Nihan Cicek for the initial work that she did regarding White Space Databases.

Finally, I would like to thank the very special person in my life, my wife, for her support, patience and encouragement. I love her more than she could ever imagine.

Amjad Yousef Majid

Delft, The Netherlands
18th August 2015

Contents

Chapter 1

Introduction

The chapter begins with a short introduction that summarizes the required background as well as highlighting the main problems that are solved in this thesis. After stating the problem formally and showing the outline of the thesis, the reader is provided with an extended background and related work to make the newly introduced concepts in the following chapters easier to follow.

1.1 Introduction

The current spectrum management policy divides the frequencies into those that are licensed, with exclusive rights needed for utilization, and unlicensed, freely available frequencies [1]. As a result of this static way of dividing the spectrum and the growing demand for radio frequencies, a spectrum scarcity problem is starting to emerge [2,3]. Therefore, a new technology has been developed that changes the focus of spectrum-sharing policy from the exclusive usage right and publicly accessible to the used and unused spectrum called Dynamic Spectrum Access (DSA) [3]. DSA enables other users besides the primary user (PU) to access licensed spectrum provided that all PU services are protected from harmful interference. Furthermore, DSA relies originally on sensing technology to detect the free frequency bands.

However, it was verified by the Federal Communications Commission (FCC) that sensing-only spectrum detect methods are unable to provide a sufficient level of protection to the licensed services. Therefore, the PU's protection mechanism depends on online services called White Space Databases, which in turn depend on known information about the incumbent services and their exact protection requirements.

When a secondary user (SU) wants to access an empty band, it queries a WSDB for the available spectrum in the SU's location and the WSDB replies with a message containing a list of the available spectrum for secondary

usage. Mostly this communication is governed by a protocol called Protocol to Access a White Space (PAWS) [4].

In order to protect the PU's services, policymakers make SUs query a WSDB anew every time its current position is X meter (i.e. 50 m in the UK and 100 m in the US) away from the last indicated location in the previous query. This rule may not have a great impact on a fixed device, but it certainly does on a mobile device, since it requires a mobile device to access a WSDB very frequently, especially if it is moving at high speed,. This can have a big impact on the battery life of a device like a smartphone.

Targeting this challenge, our research focuses on optimizing the access of a mobile white space device (MWSD) to a WSDB. Furthermore, we exploit a smartphone, a widely used device, and a cheap RTL-SDR dongle to develop an energy-efficient and low-cost Portable Spectrum Sensing Platform (PoSSP). PoSSP sends its local spectrum readings to a WSDB, which can be used to validate WSDB information and improve its quality. However, our focus is on the energy footprint of a PoSSP on a smartphone.

1.1.1 Motivation

Mobile devices are widely spread, and they are an essential part of our daily life. Moreover, they constantly support more diverse functionality. Therefore, enabling DSA to have better mobility support may open up a whole new area of development and possibilities for even more diverse mobile services, especially with the unique characteristics of white spaces. For example, the authors of [5] propose connecting RTL-SDR dongles to smartphones to crowdsource the spectrum. Currently there is a very rapid growth in unmanned aerial vehicles (UAV). The white space in the TV bands (TVWS) will enable UAVs to communicate over longer distances than 2.4 GHz or 5 GHz, in a way that is cheaper than licensed bands.

1.1.2 Problem Statement

The information in an available spectrum WSDB response message is valid for a relatively short distance. This forces a mobile white space device (MWSD) to almost constantly exchange messages with a WSDB, especially when the device moves with a high speed. Furthermore, since the communication with a WSDB is governed by PAWS, optimizing the access process of MWSD without needing to modify PAWS itself is not trivial. In this study, first, we focus on optimizing the access of mobile devices to a WSDB. Second, the focus in on optimizing the information of WSDB by providing local sensing readings from mobile devices to a WSDB. In this thesis, we answer the following research questions:

- How can you provide white space spectrum to mobile devices in an efficient way compared to the current state-of-the-art WSDB querying technique?
- What is the energy cost of sensing the spectrum using a RTL-SDR dongle connected to a smartphone and sending the data to a WSDB?

1.1.3 Key Contributions

The main contributions of this thesis are as follows:

- A comparative study on WSDBs from an SU perspective. We specifically focus on the response message size and the WSDB response time;
- Energy consumption modeling of accessing a WSDB from a mobile device (a smartphone). Specifically, we propose models that capture the energy consumption of accessing a WSDB by a mobile device using the current query technique and our newly proposed one;
- We propose a new WSDB query technique called *Multi-location WSDB query* that requires less frequent access to a WSDB to cover a particular path in comparison with the current querying technique, *Single-location* query. *Multi-location WSDB query* enables WSDB to have better mobility support;
- We introduce *Nuna*, the world's first multi-location WSDB query algorithm. It enable mobile devices to utilize white space spectrum without they need to know their paths in advance and in an efficient, in terms of number of queries, way compare to the state-of-the-art query technique. Furthermore, the results of testing *Nuna* on an Android-based smartphone are presented;
- Developing an energy-efficient Portable Spectrum Sensing Platform (PoSSP). We connect a RTL-SDR dongle to a smartphone using our in-house developed On-The-Go switch, which enable us to reduce the energy consumption of PoSSP significantly.

1.1.4 Limitations

- We are only concerned with fetching spectrum information and not how to use it. Moreover, we are located in the Netherlands where white space spectrum still not available for secondary usage.

- This study is done from a SU's perspective. Therefore, the focus is on the WSDBs response messages and the WSDBs themselves are seen as black boxes.

1.1.5 Thesis Outline

Chapter 1
The rest of Chapter 1 presents relevant background information coupled with the related work. Its goal is to position the reader such that she/he can easily follow the new concepts in the following chapters.

Chapter 2
This chapter shows the results of querying the WSDBs using the current state-of-the-art query technique. Firstly, we compare WSDBs performance from an SU perspective. Then, the energy measurement tools are introduced and the chapter ends with energy consumption model.

Chapter 3
In this chapter, we first introduce our new querying technique *multi-location WSDB query*. Then we analyze its time and energy consumption and present relevant analytical models. Finally, *Nuna*, a multi-location WSDB query algorithm, is explained, and its performance is studied.

Chapter 4
This chapter presents our Portable Spectrum Sensing Platform (PoSSP) and its energy consumption analysis. Furthermore, PoSSP is used to show the benefit of the "in-house developed" On-The-Go Switch.

Chapter 5
The conclusions and future work are discussed in this chapter.

1.2 Background and Related Work

The multi-billion-dollar price for a 20 MHz band at the European 3G spectrum auction and the overly crowded U.S frequency allocation chart in Fig. 1.1 strengthen the belief that we are running out of usable radio frequencies. However, the spectrum usage measurements obtained by FCC's Spectrum Policy Task Force [6] tell a different story. At any moment and any location most of the licensed spectrum lies idle [2]. This shows that the spectrum scarcity problem comes from the frequency management policy and not the actual physical limit. Therefore, efficient spectrum management is vital to meet the ever-growing demand for wireless data traffic. Mobile data traffic reached 2.5 exabytes per month at the end of 2014, surpassing 1.5 exabytes

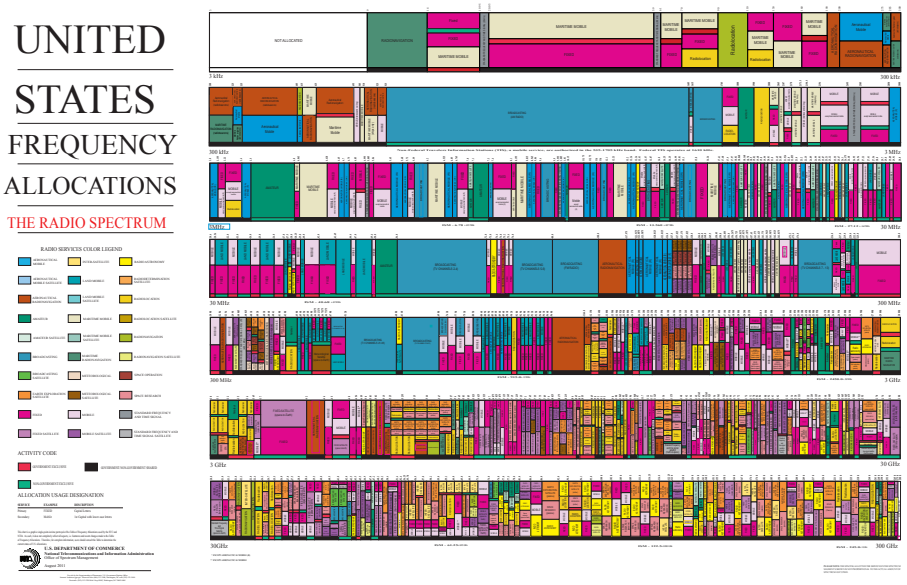


Figure 1.1: Frequency allocation chart of United States in 2011

per month at the end of 2013. It is expected that this growth will continue and will surpass 24.3 exabytes per month by 2019 [7].

1.2.1 Cognitive Radio

"*Cognitive radio is viewed as a novel approach for improving the utilization of a precious natural resource: the radio electromagnetic spectrum*" [8].

The recent advances in software and hardware make it possible to develop software-defined radio (SDR), which is a multiband radio that supports multiple air interfaces and protocols and is reconfigurable by software [9]. Cognitive radio, built on a software-defined radio platform, is an intelligent radio that can sense and autonomously reason about its communication environment and adapt its transmitter parameters accordingly [10]. Cognitive radio and Software Defined radio are the needed tools to access the spectrum dynamically.

1.2.2 Dynamic Spectrum Access

The term dynamic (opportunistic) spectrum access (DSA) is the opposite of the current static spectrum regulation policy. It aims at reusing sparsely occupied frequency bands while protecting the licensed services from harmful interference [11]. Dynamic spectrum access strategies can be basically categorized under three models [12], see Fig. 1.2.

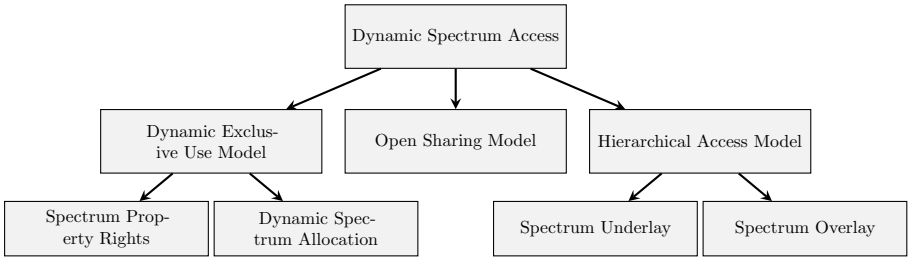


Figure 1.2: Dynamic spectrum access strategies classification [13].

Dynamic Exclusive Use Model The core idea of this model is to introduce flexibility to the current static spectrum management, and thus improve spectrum utilization. Two approaches have been proposed under this model.

- *Dynamic spectrum allocation* It assigns spectrum dynamically to different services based on the spatial and temporal traffic statistics [14]. Therefore, the overall efficiency of the spectrum usage will increase.
- *Spectrum Property Right* allows a licensee to sell or lease unoccupied portions of its own spectrum to another one [15], in a process referred to as spectrum trading [16, 17]. Based on this strategy, the market is supposed to select the most profitable use of this scarce resource.

Open Share Model This model gives equal rights to all users to access the spectrum, but they are subject to a certain protocol to manage the spectrum sharing process [12,18]. This model is supported by the phenomenal success of wireless services operating in the unlicensed industrial, scientific, and medical (ISM) frequency bands. Since the users have equal rights to access the spectrum, avoiding frequency interference is not trivial. Different distributed and centralized spectrum sharing strategies have been introduced to minimize the interference problem [19–21].

Hierarchical Access Model The hierarchical access model allows Secondary users to have access to the licensed spectrum, while limiting the interference to Primary users (licensees) [12]. There are two different approaches considered in this model:

- In the *Spectrum underlay* approach, an SU is permitted to transmit with extremely low power, i.e. below the noise floor of the PU. In this approach, high data rates for short-range communication are possible using the ultra-wide band (UWB) technique [16]. Furthermore, an SU does not need to detect the white spaces to transmit over them [12].
- *Spectrum overlay* does not impose severe restrictions on an SU's transmission power but on when and where an SU is allowed to transmit. In

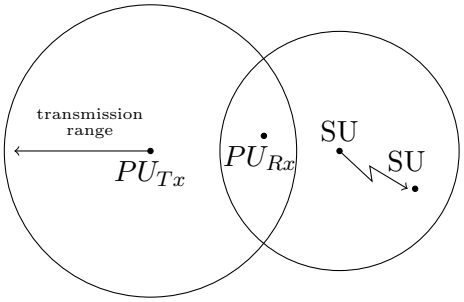


Figure 1.3: Hidden terminal problem [3].

other words, an SU is allowed to identify and exploit the white space spectrum [16]. However, the SU have to clear the frequency channel once a PU's activity is detected.

1.2.3 Spectrum Sensing

The first technique that enables a SU to access the idle spectrum (white space) is spectrum sensing. The aim of spectrum sensing is to detect idle bands and monitor PUs' activity. After detecting an empty band a SU can communicate over it.However, once a PU re-utilizes a channel, the cognitive radio should switch to a new spectrum hole without causing harmful interference to the PU. Detecting a spectrum hole, in the context of cognitive radio networks, involves many challenges: (i) According to IEEE 802.22 the detection accuracy should be at least 90% [22]. (ii) A wide spectrum bandwidth needs to be sensed to find a sufficient number of white spaces. Moreover, (iii) an SU needs to be able to sense the transmitted signals of "potentially" many PUs with different characteristics [22].

Spectrum-sensing techniques can generally be categorised as local sensing and cooperative sensing [23].

Local Sensing In the local sensing technique, each SU independently senses the spectrum, detects unoccupied channels, and then chooses a white space "empty band" to communication over it. This is called a listen-before-talk strategy. Furthermore, an SU is responsible for constantly monitoring the absence of PU signals to minimize any potential interference [23].

There are three dominant techniques to implement the local sensing strategy:

- *Energy Detection* is the most common spectrum sensing method because of its low computational and implementation complexities [24–27]. Additionally, the receiver does not need any knowledge about a PU's signals. The energy detector compares the sensed level of

energy to a pre-defined threshold. The band is idle if the level is below the threshold; otherwise the channel is occupied [28]. However, the energy detection approach cannot distinguish between a signal and noise and performs poorly under low signal-to-noise ratio (SNR) values. Moreover, the energy detector method does not work efficiently when a spread spectrum signal is involved [29].

- *Matched Filter* When the PU's signal is known in advance (i.e. modulation type, bandwidth, pulse shape and packet format), matched-filter detection is the optimum method to detect its presence [30]. It detects a single presence by correlating the features of the observed signal to the information that the filter possess. An advantage of a matched-filter detector is the short time it requires to achieve a certain false alarm or misdetection rate [31]. Its disadvantages are the high implementation complexity and large power consumption [13].

- *Cyclostationary feature detection* Most of the modulated signals can be characterized by the cyclostationary feature, because their means and autocorrelations exhibit periodicity. Therefore, it is possible to distinguish between a signal and noise, and distinguishing between PUs' signals is also possible [32].

In addition to the detection techniques discussed, there are other less common detection techniques such us eigenvalue-based [33] and waveform-based [34] detection techniques. Sensing-only detection methods do not possess sufficient information about the incumbent services. Therefore, they cannot guarantee enough protection [3, 23, 35] level to PUs' services as demanded by Federal Communication Commission (FCC) [36]. For example, local-sensing techniques suffer from the hidden terminal, problem see Fig 1.3 where a PU's receiver is within the range of an SU's transmitter while the SU is outside the range of the PU's transmitter or the SU can not sense the presence of the PU's transmitter due to shadowing or multipath fading effects.

Cooperative Sensing Cooperative sensing increases the accuracy of local sensing detection and mitigates the effects of multipath fading and shadowing, as well as reducing the probability of having a hidden terminal [3]. In cooperative sensing, a group of SUs share their observations/decisions to have a more general view about the communication environment. This can be implemented using a central or distributed approach [37]. In the distributed approach, the SUs exchange the local sensing decisions/observations and then each SU makes its own independent decision about the selection of an idle channel. In the centralized strategy, on the other hand, all SUs send their sensing results/decisions to a central station, i.e. a base station, and

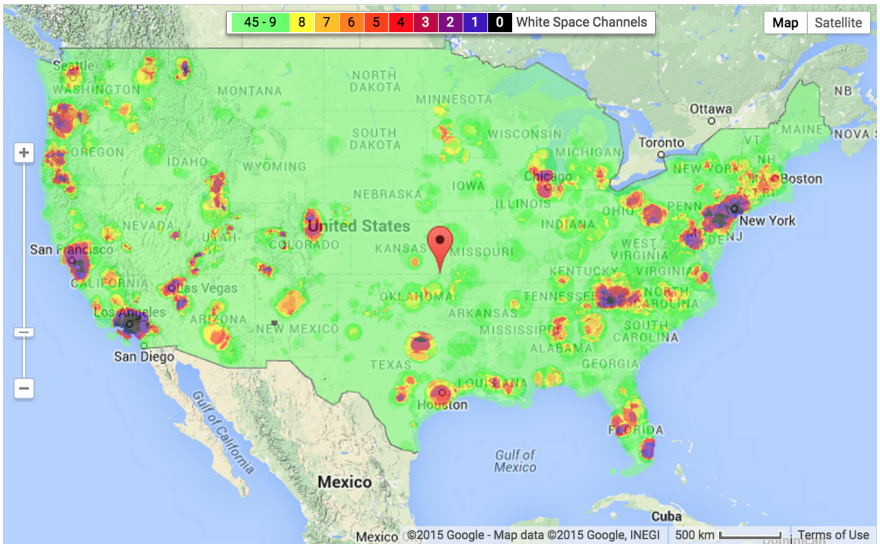


Figure 1.4: An example of the available TV white space spectrum, TV channels, for the secondary usage in the US [38] using Google WSDB.

this central station processes the data and makes the decisions about the network accessibility [13].

1.2.4 White Spaces

White spaces, or spectrum holes, are unused spectrum at specific times and locations that can be exploited through spectrum sharing technology [39]. TV white spaces (TVWS) are spectrum holes in the VHF/UHF band. In Europe, the TVWS ranges from 470–790 MHz [40, 41], while in the US it occupies a non-continuous range from 54 MHz to 698 MHz [42]. According to modeling studies commissioned by Ofcom, the total capacity associated with TVWS is significant. Over 50% of locations in the UK are likely to have more than 150 MHz of interleaved spectrum, and even at 90% of locations around 100 MHz of interleaved spectrum might be available for cognitive access [43]. TVWS' propagation properties make it a convenient and desirable spectrum for a wide range of wireless services [39]. It allows for non-line-of-sight coverage [44]. Since TVWS frequencies are below 1 GHz, they can better propagate through obstacles than unlicensed industrial, scientific, and medical (ISM) bands (2.4 and 5.7 GHz). Fig 1.4, from Google's spectrum database website, shows an example of the available TV channels for secondary users in the US.

1.2.5 White Spaces Database

Since sensing technology can not offer sufficient level of protection to the PUs' services, as concluded by FCC [36], the focus now is on ab online service know as White Space Database (WSDB). A WSDB, or Spectrum database, is an online database that contains geo-location information about the availability of the white space spectrum in the licensed/reserved spectrum [45]. One advantage of relying on a WSDB is that it handles the complexity of spectrum policy conformance on behalf of an SU. Another benefit of a WSDB is that it relies on known information about the bands, including the exact types of incumbent services and their specific protection requirements [45]. Although WSDBs are still in an early stage of development, there are already quite a number of them, for example Google Spectrum database [38], SpectrumBridge WSDB [46] and Nominet WSDB [47].

Interestingly, the use of WSDBs goes far beyond aiding in accessing TV white spaces (TVWSs). For example, spectrum databases are also considered for helping in regular spectrum sensing for radar activity detection (in L, S and C bands) [48] - also refer to US government plans for 3.5 GHz Spectrum Access System [49]. More importantly, the use of WSDBs becomes increasingly relevant with the advent of Licensed Shared Access (LSA) [50]. Also, in Program Making and Special Events (PMSE) [51, 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.

Generally, considering the existing WSDBs presented in Table 2.1, the knowledge of 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. A very preliminary consideration of TV WSDB has been presented in [45]. One of the first complete local, sensing-free, WSDB implementation and evaluation is [52], 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 [53]. A very recent evaluation of LSA using an incumbent WSDB is presented in [54].

1.2.6 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 [55], by conducting massively crowdsourced (≈30,000 users in total) measurements on US carrier's UMTS/HSPA and EVDO Radio Access Techniques (RATs), delay characteristics were measured [55, Figs. 4–7] (among others) of accessing popular Web services. However, no WSDBs

```
curl -XPOST https://www.googleapis.com/rpc -H "Content-Type:
     application/json" --data '{
  "jsonrpc": "2.0",
  "method": "spectrum.paws.getSpectrum",
  "apiVersion": "v1explorer",
  "params": {
    "type": "AVAIL_SPECTRUM_REQ",
    "version": "1.0",
    "deviceDesc": { "serialNumber": "your_serial_number",
    "fccId": "TEST", "fccTvbdDeviceType": "MODE_1" },
    "location": { "point": { "center": {"latitude": 42.0986,
    "longitude": -75.9183} } },
    "antenna": { "height": 30.0, "heightType": "AGL" },
    "owner": { "owner": { } },
    "capabilities": { "frequencyRanges":
    [{ "startHz": 800000000,"stopHz": 850000000 },
    { "startHz": 900000000, "stopHz": 950000000 }] },
    "key": "your_API_key"
  },
  "id": "any_string"
}'
```

Figure 1.5: Google implementation of "AVAIL_SPECTRUM_REQ" message [38]

were considered here.

1.2.7 Protocol to Access White Space Database

The Protocol to Access White Space Database (PAWS) is a protocol that enables a white-space device to communicate with a spectrum database to obtain information about the white spaces for a specific location and time. A device may be required to register with the database with some credentials prior to being allowed to query [56]. According to PAWS, a WSD should use the initialization messages, INIT_REQ and INIT_REPS, to exchange capability information with WSDBs. The protocol also specifies two messages for the registration process. Most importantly, a WSD queries a WSDB for white spaces using an "AVAIL_SPECTRUM_REQ" message, see Fig. 1.5. Furthermore, an SU can use batch query messages to ask for a frequency for a group of locations. However, to the best of our knowledge, no single WSDB supports this type of message. Finally, there are also messages for notifications and validations [4].

1.2.8 Energy Cost of Internet Access Use

Here we focus on the systems-related research pertaining to this topic. One of the first such studies can be found in [57]. Therein, energy consumption measurements in three networking technologies (i) UMTS/WCDMA, (ii)

GSM/EDGE/GPRS and (iii) IEEE 802.11b were performed and it was concluded that (i) and (ii) have a significant tail energy, it is the energy consumed after competing a ntwork task and before going to sleep mode, overhead. A more detailed investigation of this overhead (including state machine modeling), i.e. characterization or radio resource control in 3GPP-based networks, is discussed in [58].

Power characteristics of LTE (post-3G) RAT were studied empirically with data collected from 20 LTE-enabled smartphones and compared with IEEE 802.11g and UMTS CDMA in [59], LTE was found to be less power efficient than other networks due to its long high-power tail [59, Fig. 12].

A very recent piece of work about energy consumption measurements and modeling of IEEE 802.11x and 3GPP-based systems (as is performed in this paper) was presented in [60]. 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 [58, Sec. 2]. Another recent though less detailed process of RAT access energy cost of Web services access is presented in [61]. In all the above-mentioned works, energy profiling is only based on popular web services. Thus the energy cost of WEDB access has never been considered. Naturally, intrinsic features of WSDBs, e.g. large response delays [52, Fig. 13], demand a re-evaluation of the energy profile studies.

1.2.9 Outdoor Localization

There is a broad range of outdoor localization techniques. GPS is considered to be one of the most popular localization technologies [62]. However, since GPS consumes relatively lots of energy, there are a number of methods/techniques that try to replace GPS or rely on GPS only to enforce the estimation of their own techniques. i.e. a cellular phone can use cell-tower triangulation to localize itself [63], or city-wide WiFi, as investigated in [64, 65], trading accuracy to reduce power consumption. Another approach uses fingerprinting techniques, where a device trying to match captures signatures against a set of geotagged signatures to identify a device position [66]. We choose to use GPS for localization, since it is the dominant outdoor localization technique. Moreover, it is available on almost all smartphones.

This chapter started with introducing the essential tools, i.e. software-defined radio, and technologies, i.e. sensing, that make opportunistic access to the licensed spectrum possible. How these techniques have evolved, from relying on local sensing to cooperative sensing to WSDBs. WSDBs are regarded as the state-of-the-art technology that enables SUs to access white space spectrum. To the best of our knowledge there is no work that considers

optimizing the access of mobile white space-enabled devices to a WSDB. Furthermore, we are also missing a comparative study between the WSDBs. Therefore, the next chapter will compares the WSDBs' performance from an SU perspective.

Chapter 2

WSDB: Single-Location Query

The goal of this chapter is to provide the researcher/reader with a solid technical understanding about the performance of WSDBs. We take a comparative study approach to clarify the differences between the WSDBs.

This chapter begins by introducing the WSDBs involved in this study, then WSDBs' response messages sizes are shown in Section 2.1.1 and the response delays are evaluated in Section 2.1.2. Furthermore, Section 2.2 introduces the power measurement tools and shows the energy cost of querying a WSDB. Finally, Section 2.3 presents an analytical model for energy consumption.

2.1 Response Messages Characteristics

We begin by profiling the response messages in terms of size and delay using a stationary MacBook Air running Ubuntu 14.4, R2014 MATLAB and cURL v7.30.0. Table 2.1 summarizes high-level information about the WSDBs that are used in this study.

2.1.1 Size Distribution

Fig. 2.1 shows the cumulative distribution function (CDF) of available-spectrum WSDB response messages sizes. Each WSDB is queried, along a straight line, for 100 different locations, see Table 2.3. The reasons for the significant differences in message sizes are the amount of information being communicated back by a WSDB and how it is presented, i.e. MSR responds with an XML list. The order of the elements in this list represents the number of a TV channel and a true or false indicates the availability of the channel. FAI replies with a JSON message contain a list of frequencies to indicate the start of a channel and the maximum allowed transmission power. SBO responses have larger response message sizes than other WSDBs. SBO

Table 2.1: List of WSDBs involved in the experiments

Company	Abbr.[1]	Links	Land	Format	Com[2]	MLq[3]
Google	GGL	[38]	US	JSON	Yes	Yes
Microsoft	MRS	[67]	US	XML	No	No
SpectrumBridge	SBI	[46]	US	XML-URL	Yes	No
SpectrumBridge-Ofcom	SBO	[68]	UK	JSON	No	Yes
Nominet	NOM	[47]	UK	JSON	No	No
Csir	CSI	[69]	SA	JSON	No	No
Fairspectrum	FAI	[70]	UK	JSON	No	No

[1]Abbr: Abbreviation, [2]Com: Commercial, [3]MLq: Multi-Location query

Table 2.2: Response time and response messages size of WSDBs

WSDB	Size (kB)	Variance (kB)	Delay (s)	Variance (s)
GGL	2.60	450.00	0.88	0.06
MSR	1.57	0.04	2.30	0.05
SBI	0.31	0.26	1.63	0.03
SBO	8.70	0.163	24.2	8.00
NOM	5.10	0.78	0.92	0.002
CSI	1.33	210.00	1.12	0.10
FAI	1.70	0.02	1.65	0.02

responses use a JSON template and in each entry in their frequencies list, the response message specifies the start and stop frequencies, the maximum allowed transmission power and the resolution bandwidths of 8 MHz and 100 kHz. The response messages of SBI are the smallest in size. The numbers of the available channels are reported in one XML tag.

In the second column of Table 2.2, the mean sizes of WSDB response message sizes are shown, and the third column presents the variances of the message sizes.

The essential difference between MSR and SBI, and other WSDBs, is that the response messages of MSR and SBI are only about the TV white space (TVWS). Their response messages cannot, in their current formats, carry information about other frequency bands, whereas the response messages of other WSDBs can easily specify other white spaces (WS). Unlike other WSDBs, MSR and SBI do not use JSON messaging format, and thus they do not use PAWS as the communication protocol. SBI uses sbi-paws [71] as its own implementation version of PAWS, whereas MSR uses KNOWS [72] as its communication protocol. Furthermore, MSR and SBO reply with a fixed message size (fixed template) and they control access to a particular band by the transmission power. In other words, specifying a very low

Table 2.3: Starting and ending coordinates of the straight paths specified to examine the size and the delay of the spectrum-available response messages.

Coordinates of Size distribution experiment

Land	Start (Latitude,Longitude)	End (Latitude,Longitude)
UK	(51.785840, 0.288950)	(51.785840, -2.062151)
US	(40.725952, -74.665983)	(36.115164, -95.891569)
SA	(-29.00000, 25,000000)	(-22.200000, 26.3000000)

Coordinates of Delay distribution experiment

UK	(51.506753, -0.127686)	(51.431471, -2.577637)
US	(40.506753, -100.127686)	(40.431471, -102.577637)
SA	(-24.493247, 32,450314)	(-24.568529, 30.3000363)

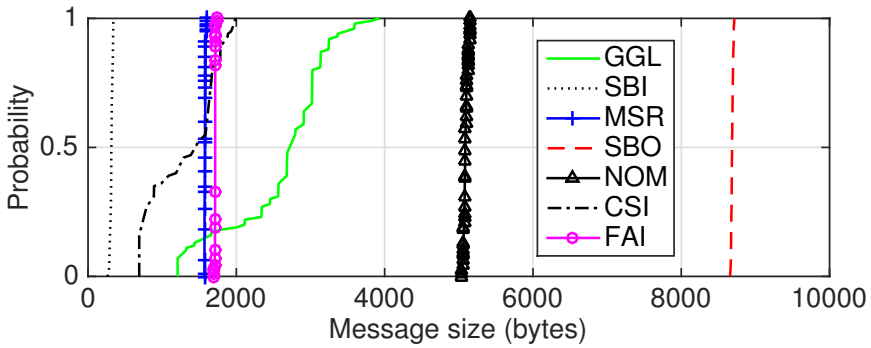


Figure 2.1: CDF of response message size of different WSDBs, see Table 2.1. The reasons for the significant differences in message sizes are the amount of information being communicated back and the way they are presented.

transmission power for a certain frequency band will prevent an SU from communicating over it. Other WSDBs respond with a variable message size that only contains the available spectrum.

2.1.2 Delay Distribution

To compare WSDB performance, we queried them at approximately the same time and along equal distances in all three countries: the US, the UK and SA, see Table 2.3. The experiment includes querying 50 locations in each country. Each location was queried 20 times and average response times were computed.

Fig. 2.2 shows the probability density function (PDF) of queries' response times, total time from sending a cURL request to a full WSDB reply. In general, the response time is WSDB-dependent. For example, GGL is the most delay-efficient, with a mean of 0.88 seconds, while SBO is the least,

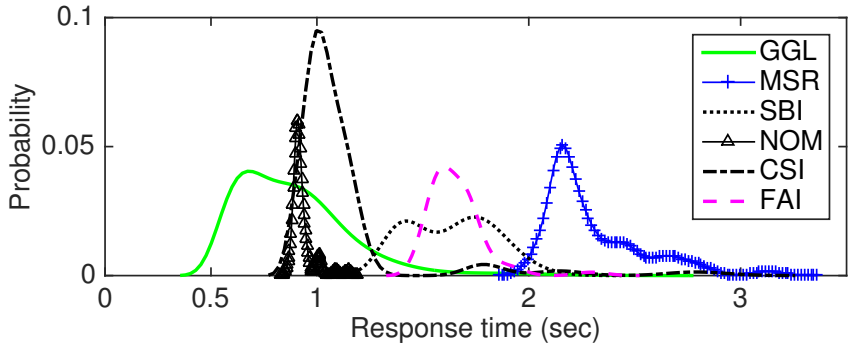


Figure 2.2: PDF of mean WSDB response times for querying along straight lines, in the US, the UK and SA, divided into 50 intervals where each location was queried 20 times.

Table 2.4: Locations used in the experiments

Symbol	Latitude	Longitude	Type
LA	34.047955	118.256013	downtown
WV	40.729655	74.002854	urban canyon
SC	41.102884	-82.957361	flatland
LE	41.575416	81.585442	lake coast
CB	34.047955	77.885639	Carolina Beach
ALp1	70.789158	-158.260175	——
ALp2	48.654895	-103.933643	——
USp1	61.091065	-156.296335	——
USp2	32.708249	-103.149980	——

with a significant mean delay of 24 seconds and a variance of 8 seconds, see Table 2.2[1] . Because of this extreme delay of SBO we excluded its results from Fig. 2.2 for clarity. NOM has a consistent response with a variance of 0.002 seconds. Furthermore, the PDF's distributions indicate that the response time is location dependent, since the response time only changes when we change the queried location.

Delay: US vs Alaska

To further investigate the dependency between a queried location and the WSDB response time, we queried GGL WSDB on straight lines across Alaska and the US. A Google Remote Procedure Call (RPC)[2] server [74] was also

[1]SBO response times varied from day to day. The fastest response we observed was about 5 seconds and the slowest was about 28 seconds.

[2]Remote Procedure Call (RPC) allows a remote user to request a server to execute a piece of code without having to know the details about the execution and obtaining the return value [73]. In other words, calling an RPC server is like calling a normal function

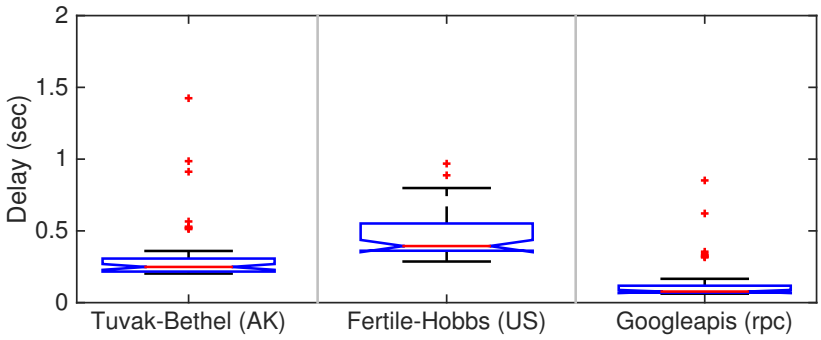


Figure 2.3: Response time of querying GGL WSDB across Alaska from ALp1 to ALp2 and US from USp1 to USp2, see Table 2.4, compared to querying Google's RPC server: https://www.googleapis.com/rpc/.

queried for comparison. The query process was as follow: Alaska and the US were divided into 49 segments. We queried the first location of Alaska and then the first location in the US and then the RPC server. This process was repeated 20 times and the averages were computed. Then the query process moved to the next location. The results, shown in Fig. 2.3, indicate that there is a dependency between a location and a WSDB response time, confirming the results of Section 2.1.2.

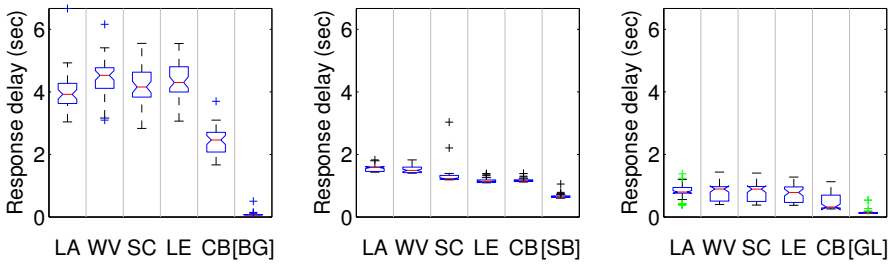
Furthermore, the response to any position in Alaska is almost always faster than the response time to any other location in the US. The reason for this difference is that, since the number of transmitters in Alaska is far less than the number of transmitters in the US, and there is a geographic separation between them, the database of Alaska is much smaller than the database of the US. As a result, the internal calculation to find a result takes less time in the Alaskan database than in the US one.

Locations and response time dependency

To further test the location and the WSDB response time dependency, we chose three WSDBs, namely GGL, MRS, and SBI, and five different locations. Each location was queried 50 times. The results presented in Fig. 2.4 clearly show that there is a dependency between a location and a WSDB response time. Moreover, the results also confirm the results shown in Fig. 2.2. Additionally, WSDBs' response times, compared to the web server response times of the respective organizations (Bing, Spectrumbridge, and Google) as in Fig. 2.4, are much longer (in the extreme case of MSR longer than 4 seconds)[3].

where a programmer can provide the input parameters and receives the output without caring about the details of the function itself.

[3]The same experiment performed using our developed code at US-headquartered Adaptrum, Inc. for MSR WSDB revealed the same delay profile as in Fig. 2.4(a).



(a) MSR (b) SBI (c) GGL

Figure 2.4: Query response delay for (a): MSR, (b): SBI, (c): GGL, for 50 queries, see Table 2.4, BG: http://www.bing.com, SB: http://www.spectrumbridge.com/home.aspx, GL: http://www.google.com. Kernel smoothing and box plot was implemented using MATLAB'second R2013a ksdensity and boxplot function, respectively.

In order to see if the number of antennas around a queried location impacts the WSDB response time. The following experiment was setup: we chose 30 locations with a different number of transmitters around each location in a circle of 60 miles [75, 76]. Each site was queried 20 times. This experiment was repeated at different day times. The average delay of all the 30 locations with a number of antennas around a position ranges from 0 to 25, with a mean of 0.82 and a variance of 0.36. In other words, the average delays of locations with 25, 5, 0 and 3 TV transmitters around them were 0.74, 0.77, 0.71, 0.74 seconds respectively, see Fig. 2.5. The results show that the number of antennas around a location has no observable effect on the WSDB response time. Therefore, we conclude that there is no correlation between a WSDB response time and the number of antennas around a queried location.

2.2 Energy Cost of Querying WSDB

Since we are targeting mobile devices, energy consumption is of premium importance. To realistically assess the energy costs associated with WSDB access we used a battery-independent, high-resolution, fully portable power meter called NEAT [77], see Fig. **??**. NEAT can extract exactly the amount of energy consumed in a particular operation, i.e. a WSDB query [78]. We choose to carry out experiments using a Samsung Galaxy S3 (GT-I9300) smartphone running CyanogenMod version 10.2.0.

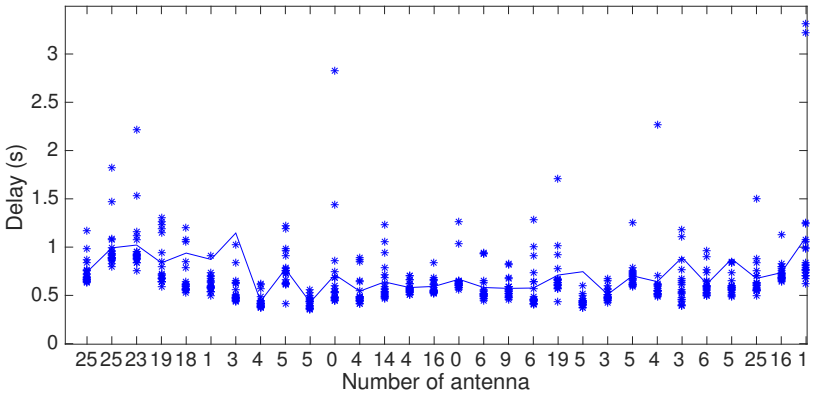


Figure 2.5: Number of antennas around a queried point in a circle of 60 miles vs. response time.

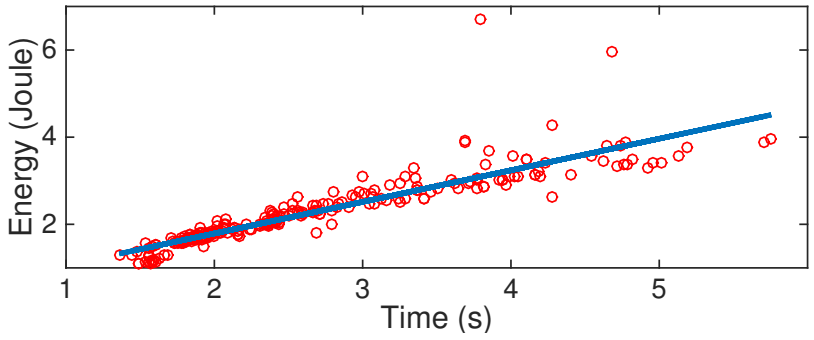


Figure 2.6: The energy consumption of querying GGL WSDB for different locations, using Samsung GalaxyS3.

2.3 Modeling Energy Cost of Querying WSDB

To measure the energy consumption we used NEAT [77]. It is fully portable and phone battery-independent power meter capable of extract the exact amount of energy consumed in a WSDB query. It is embedded into an enlarged back cover and record the data on a micro SD card. Furthermore, 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 a hardware trigger.

The authors in [78] have proposed an energy consumption model for querying a WSDB. They queried two WSDBs, namely SBI and GGL, over three different networks: Vodafone, T-mobile, and kpn. Their conclusion was that choosing a cellular network does not have a significant impact on

the energy consumption of querying a WSDB (excluding the tail energy). Although we agree with this conclusion, we think choosing only two locations may not generate a representative model. Therefore, we propose an energy consumption model based on querying GGL WSDB, see Fig 2.6, for over 250 different locations,

$$E_q(t_q) = (q_c t_q + a_q), \tag{2.1}$$

where t_q is a query response time. Based on the used MATLAB linear fitting function q_c and a_c are 0.7 and 0.3 respectively. Since the dominant factor that affects the energy consumption is the response time, namely the time from sending the query to a WSDB until receiving a full response. We will further generalize this model in the following chapter, see Section 3.2.2.

In this chapter, the performance of WSDBs was compared from an SU's perspective. We observed that there are relatively big differences in the response message sizes of WSDBs. Furthermore, there are noticeable variations in the mean response times of WSDBs and their delay distributions. We also investigated the dependency between a queried location and a WSDB's response time. Finally, we proposed an energy consumption model of querying a WSDB. In the next chapter, we present our new querying technique that offers better mobility support.

Chapter 3

WSDB Multi-Location Query

This chapter begins by introducing our new WSDB query technique denoted as *Multi-location WSDB query*. Its response time and energy consumption are analyzed, and their analytical models are tested. Furthermore, we describe three different scenarios for doing a Multi-location WSDB query, and propose a Multi-location query algorithm called *Nuna*. Its implementation on an Android-based smartphone is explained. Finally, we show that *Nuna* reduces the energy consumption and the number of queries by 50%.

3.1 WSDB Multi-Location Query

PAWS specifies a way to query a batch of locations in one request [4, Sec 4.5.4]. This feature might reduce the number of queries a white-space enabled device needs to cover a particular path. However, to the best of our knowledge, there is no single WSDB to support this feature. Inspired by batch query we were able[1] to find a new query technique that enables an SU to query a group of locations in one request, denoted as *Multi-location WSDB query*.

A spectrum-available request message is divided into a header- and a data-part [79]. The data-part is represented as a JSON object for some WSDBs. To make a Multi-location query, an SU has to concatenate the data-parts for the intended locations into one JSON array to be the data-part of the new Multi-location request message. The WSDB response message contains a list of the available frequencies for each position, and these lists

[1] We built an online php tool that enables us to easily interact with WSDBs, and to gather some statistics. During the test on changing the query parameters we discovered this model of operations, *Multi-location query*, is possible for some WSDBs. Furthermore, we tried many different parameter combinations of the request messages, including the batch query proposed by PAWS, and they were all recognised as invalid requests by the WSDBs.

are concatenated into one big list for the entire queried path. This makes the access to WSDB less frequent, and an SU less dependent on a regulator's WSDB.

An essential difference between a batch query and a multi-location query is that a batch query only allows latitude and longitude to be changed, while a multi-location query allows an SU to change any parameter in the request message. This difference can be very important if, for example, an antenna is mounted on a drone, which means that in addition to the longitude and latitude parameters, the elevation parameter needs to be changed. In other words, a multi-location query is more general than a batch query. It should be highlighted that this feature is not supported by all WSDBs. GGL and SBO WSDBs do support multi-location queries, but other WSDBs do not.

GGL WSDB allows a maximum of 3000 locations in one multi-location query. This extends the distance that a device can move in without the need to re-query WSDB from 100 m to 300 km. The maximum size[2] of an SBO multi-location query is 1000. However, it is significantly slower than GGL as will be shown in Section 3.2.

3.2 Time and Energy Analysis of Multi-Location WSDB queries

This section shows the response time of a multi-location WSDB query for a stationary station. Furthermore, it presents energy consumption and response time models for mobile devices. Finally, we discuss the response message size.

3.2.1 Response time

To examine how the GGL WSDB response time increases when the number of locations in a multi-location query increases and if a path location may affect the average response time, the following experiment, using a MacBook Air with Ubuntu 14.04 running MATLAB R2014b and cURL v7.30.0, was set up.

We chose three randomly located paths (Path 1, Path 2, and Path 3, see Fig. 3.1) in the US, that are approximately of the same length of 12 km, with different start and end coordinates. We divided each path equally such that the biggest multi-location query covered the entire path. The query process started with querying one location and then kept doubling the size of the query until it became 128 locations. Furthermore, each multi-location

[2]This is the highest number of locations that we obtained responses for. Above 1000 locations we did not receive any reply. When going above 3000 locations, GGL responds with a message indicating that the request exceeds the maximum allowed number of locations.

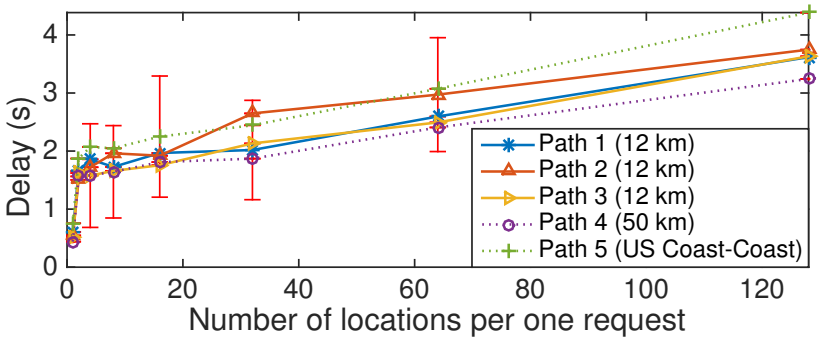


Figure 3.1: Response time of GGL multi-location WSDB queries along different paths, using MacBook Air running Ubuntu 14.04 with MATLAB and cURL

was generated 20 times and its average was computed. Fig. 3.1 shows that the response delay along the paths have almost the same response time for each multi-location WSDB query.

In order to find out the effect of increasing the step size (128 locations across 10 km vs. 128 locations across US from coast-to-coast) on GGL WSDB response time, Path 4 and Path 5 were chosen with different lengths, 50 km and 4000 km (coast-to-coast) respectively. The results presented in Fig. 3.1 clearly show that the step size does not alter the response delay. Remarkably, in all five paths there is a relatively big difference of about one second in delay between querying one and two locations in one request. Neither we nor the Google WSDB engineer [80] were able to give a clear explanation for this behavior. After that the delay increment is marginal: there is an increase of about two seconds when the number of queried locations is increased from 2 to 128 locations. The results also indicate that a multi-location query is extremely faster to cover a particular path than the current query technique: querying GGL for 128 locations sequentially takes, at least, about 1 minute, whereas a multi-location query takes about 4 seconds. MATLAB `errorbar` function was used to show the overall error of each path. Finally, we also tested SBO for the multi-location query response time, which was far slower than GGL. For a query of 32 locations the average response time was 106 s and for 64 locations 215 s. Therefore, we proceeded only with GGL.

3.2.2 Energy Model

To estimate how a multi-location WSDB query process affects a smartphone's battery life, we derived an analytical model for energy consumption of multi-location WSDB queries for an Android-OS based smartphone. The portable power meter NEAT [81] was used to extract the exact amount of energy consumed on a multi-location query. Tail energy [78] was excluded since it is

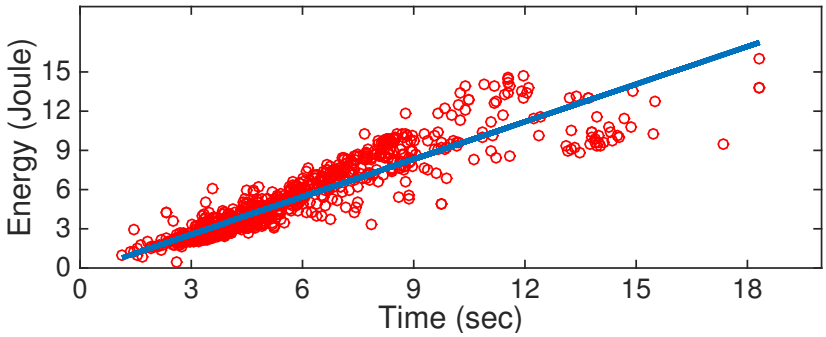


Figure 3.2: Energy consumption of GGL multi-location WSDB querying (excluding tail energy) as a function of query time.

Table 3.1: Energy consumption model 3.3 validation

Query time (s)	Measured energy (J)	Estimated energy (J)	Accuracy (%)
2.00	1.72	1.60	93
2.58	2.10	2.15	97
6.70	6.20	6.00	96
8.90	8.60	8.15	91
10.60	10.98	9.77	88
29.77	25.60	27.98	90

independent of the multi-location query size. Over several days we collected more than 1000 multi-locaion WSDB queries. The sizes of the multi-location queries range from 1 to 150 locations.

$$E_q(t) = q_c t_q + a_c, \tag{3.1}$$

where t_q is the time span of a query in seconds. Using MATLAB fitting function `polyfit`, as shown in Fig. 3.2, gives the values of $q_c = 0.95$ and of a_c = -0.3 (the energy model will not produce a minus energy value since the time of the query of a mobile device is always greater than 1). The number of queries with different time durations, test data, was chosen to estimate the accuracy of our proposed model. The results, shown in Table 3.1, indicate that the model predicts energy consumption with a high accuracy. Finally, since a multi-location query is just a single query that contains multiple locations, we can regard the single location query as a special case of the multi-location query.

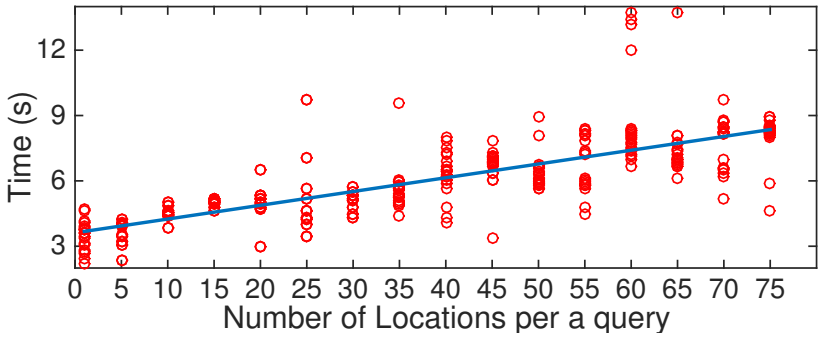


Figure 3.3: Number of locations vs. response time of multi-location WSDB query on an Android-based smartphone.

3.2.3 Time Model

Another important statistical model is the estimated response time of a particular multi-location WSDB query size for mobile stations. In Fig. 3.3 the x axis represents the multi-location query size. The number of locations was increased by a size of 5 locations and for each size 10 multi-location queries were generated, giving a total of 160 multi-location queries.

For a smartphone to do a query, the following set of events might need to happen: phone wakes up, does the query, and goes to sleep. We are only interested in the query time and not the time required for wake up or go to sleep, since they are independent of the multi-location query size. The overall time increment when the number of locations was increased from one to 75 is about 4 seconds. Using MATLAB fitting function `polyfit` the following model was generated,

$$t_q(n) = a_q n + b_q, \tag{3.2}$$

with $a_q = 0.087$ and of $b_q = 3$. n is the number of locations queried in one multi-location query. The value of the parameter a_q shows the benefit of using a multi-location query technique. The addition of one extra location to a multi-location query increases the query time by only 0.09 s, and this increment leads to a very marginal increase in energy consumption.

Combining the two proposed model gives us a model that estimate the energy consumption of a multi-location WSDB query for a particular size.

$$E_q(n) = c_q n + d_q, \tag{3.3}$$

where c_q is 0.082 and d_q is 2.55, respectively. This model also shows that increasing the multi-location query size by one location will add only 0.082 J extra.

Table 3.2: Plain text, compressed (Using gzip), and the compression ratio of GGL WSDB response message sizes

No. locations	Message size (kB)	Compressed size (kB)	Compression ration (%)
1	2.1	0.626	70
2	4.2	0.642	85
4	8.4	0.697	92
16	33.6	0.899	97
64	134.3	1.600	98
128	268.5	2.600	99

3.2.4 Response Message Size

Increasing the number of locations in a multi-location WSDB query will increase the response message size, i.e. GGL WSDB responds with an available-spectrum response message of about 2 MB when the number of positions is about 1000. This may lead to extra expenses for a user. However, a user can ask for compressed response messages, which reduces the size significantly. Table 3.2 shows the compression ratio of the response messages to different multi-location queries. We see that the compression ratio surpasses the 90% very quickly.

3.3 Multi-location WSDB queries: scenarios and techniques

Multi-location WSDB queries may serve different purposes. For instance, a master device serves other slave devices in a particular region. In such a scenario, the interest is not in a particular path but rather the entire region. It is also possible that the master device wants to communicate over white spaces. In this situation it has to know its current location or process the white space data for the entire region to find one common frequency to eliminate the localisation problem.

- *Area Multi-location (AML)*: If the starting and the farthest locations on the path are known, but the path is completely unknown, an SU can query the entire region by putting the locations within a circle. Its center is the start location and the farthest location of the path on its circumference. However, the number of locations increases quadratically with the length of the distance between the two locations. For example, if the distance is 1 km, then the number of locations in the circular area is about 314, while if the distance is 10 km then the number of locations is about 31400. Additionally, this technique

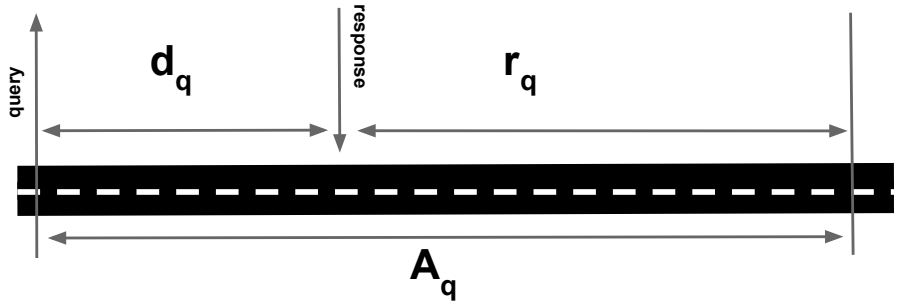


Figure 3.4: The core idea of Nuna. A_q represents the distance that the received data from a WSDB is valild in it. d_q is the query distance, obtained from multiplying the query time be the speed of the device. r_q the remaining distance.

may impose a start-up delay, since querying 3000 locations from GGL WSDB in one multi-location query takes, based on our experiments, about 20 seconds using a laptop.

- *Oracle Multi-location (OML)*: If the path and all its coordinates are known in advance, i.e. by using Google Maps, an SU can query it in one multi-location query.

  AML and OML describe techniques to obtain the frequencies from a WSDB for a particular region or path, but not where to use them. If the device needs to communicate over white space spectrum it has to know its location. However, these techniques may service different purposes, for example, providing data for analysis by a master device or a secondary server. We observed a lot of similarities in the frequencies list in the response messages. This may enable a mobile device to find one common frequency band for the entire region and communicate over it without the need for localization in this region.

- *On-Demand Multi-location query (ODML)*: If the path is not known in advance, but a device can track the user movements, this is enough for state-of-the-art WSDB querying techniques. A device can try to predict the next direction of movement and query it in a multi-location WSDB query fashion.

3.4 Nuna: Multi-Location On-demand Query Algorithm

Nuna is the first algorithm that does on-demand multi-location WSDB query, designed for location-aware mobile devices, i.e. smartphones. It predicts the

next direction of movement and queries a WSDB for the available spectrum in it.

3.4.1 Core Idea

Before introducing the core idea of *Nuna* let us highlight the main elements of the problem of querying a WSDB from a mobile device. There is a device moving with a certain *speed* (v_q) and querying a WSDB. Therefore, we have *Query Time* (t_q): the time consumed by querying the WSDB from sending a query until a full reception of a WSDB response message. Moreover, the information we receive is valid in a particular area or to a certain distance. The core idea is to convert the query time (t_q) to *Query Distance* (d_q): The distance crossed by the device while sending the query and receiving the response is given by:

$$d_q = v_q t_q, \tag{3.4}$$

This will also give us the *Remaining Distance* (r_q), which is the distance that a device can cross without the need to query the WSDB again, see Fig. 3.4. Dividing the remaining distance (r_q) by the query distance gives us the *Enlarge Factor* (e_q), i.e.

$$e_q = \left\lfloor \frac{r_q}{d_q} \right\rfloor, \tag{3.5}$$

The enlarge factor (e_q) shows how many times we can increase the size of a query successively, sending the queries back-to-back, while staying within the remaining distance (r_q). At this stage the algorithm has decided the number of locations to be queried.

The algorithm needs to predict the direction of movement to distribute the locations along the predicted path, from the history of movement (longitudes and latitudes). The algorithm computes the change factors of longitude and latitude by dividing the average change along one of the coordinates by the sum of the averages. The change factors specify the direction of movement along a straight line. The algorithm also compares the last computed change factors to the previous ones to check if there is a change in the direction or the device is moving in the same direction. If there is a change in direction, the query starts from the current location. If there is no change in the direction detected then the query starts from the last queried location.

3.4.2 Android Implementation

We have implemented *Nuna* as an application on an Android-OS. In the implementation phase *Nuna* was divided into three main stages:

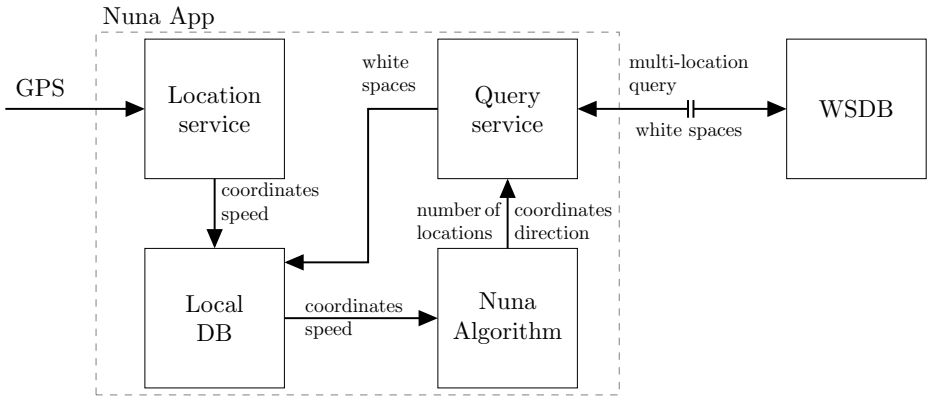


Figure 3.5: Overview of the implementation of *Nuna* on an Android-OS based smartphone. The app contains an Android service to receive the locations update from GooglePlay services and another service for the algorithm and the query service.

- The *first query* stage is used as an initialization process. *Nuna* queries only the device's current location, see Algorithm 1 line 6. The current location could be a group of surrounding locations, which is particularly useful when the device moves with a high speed.
- The *second query* stage. In this stage the algorithm predicts the direction of movement based on the history collected from the first query stage, see Algorithm 1 lines 11–12. Moreover, it calculates the size of the multi-location query, line 13, as explained in Section 3.4 and does the first multi-location WSDB query, line 15.
- In the *next query* stage the algorithm does not only decide the size of the multi-location query and predicts the direction of movement, but it also compares the current direction to the direction of movement during the previous query, see Algorithm 1 line 20–21,24. If there is a significant change then the process will start again from the *First query* stage, as a recovery mechanism. Otherwise, the algorithm does a multi-location query starting from the last queried position, unless the remaining distance (r_q) is longer than what the device can cross with its maximum speed, see Line 25. If the remaining distance is longer than that the algorithm does not generate the query and waits until the next iteration. This reduces the number of generated queries to WSDB and stops very long straight paths from being queried. As a consequence, energy consumption of the mobile device is reduced while gaining more freedom for faster movement.

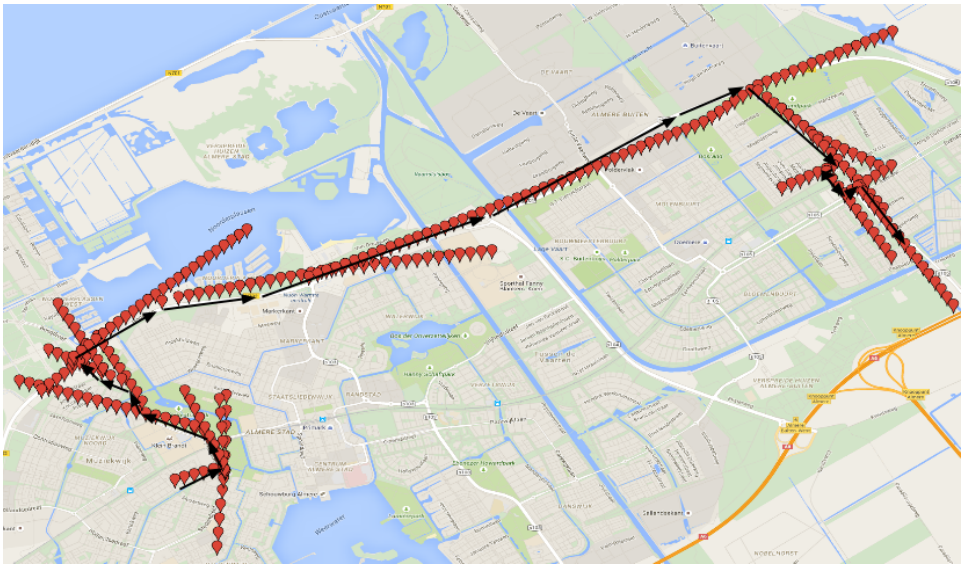
If the device was moving with a relatively low speed then the enlarge factor (e_q) becomes very large and thus the number of queried locations becomes

Algorithm 1 Nuna

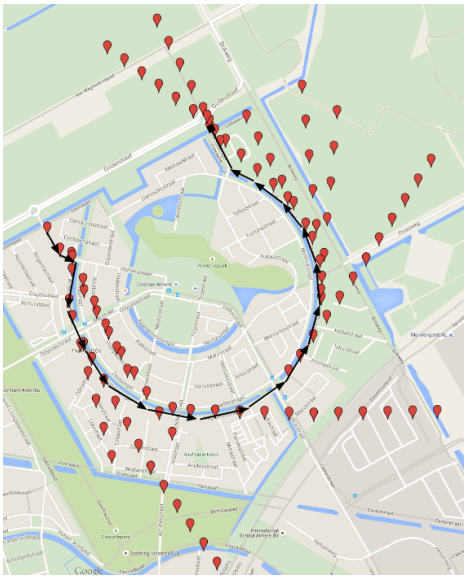
```
1: (t_b, t_a) ← (0, 0)                    ▷ Times before and after (multi) global query
2: START ← TRUE
3: while running: do
4:     if START then
5:         l ← CurrentLocation()
6:         (t_b, t_a) ← LocationQuery(l)
7:         START ← FALSE
8:         SECOND ← TRUE
9:     else
10:        if SECOND then
11:            L ← GetLocations(t_b, t_a)
12:            D ← GetDirection(L)
13:            Nol ← NumberOfLocation(S)
14:            l ← CurrentLocation()
15:            (t_b, t_a) ← MultiLocationQuery(l, Nol, D)
16:            SECOND ← FALSE
17:        else
18:            L ← GetLocations(t_b, t_a)
19:            L_a ← GetLocations(t_a, t_current)
20:            D ← GetDirection(L)
21:            D_a ← GetDirection(L_a)
22:            Nol ← NumberOfLocation(S)
23:            l ← CurrentLocation()
24:            if D − D_a < Δ then
25:                if l_last − l < D_max then
26:                    (t_b, t_a) ← MultiLocationQuery(l, Nol, D)
27:            else
28:                SECOND ← TRUE
29:                (t_b, t_a) ← LocationQuery(l)
30:    sleep(t_s)
```

very large too. Therefore, we put a limit on the size of a multi-location query between 5 and 20, depending on the speed of the device.

The error in GPS positioning estimation may make *Nuna* predict a change in the direction or make the predicted path line not perfectly aligned with the real path. To reduce the number of false direction change detections and to make use of the allowed distance of movement, 100 m for GGL WSDB, a tolerance threshold of 0.08 in the direction change estimation was introduced. This value was chosen based on a large number of tests, as a balance between GPS estimation error and small direction changes that should not generate a new query, i.e. changing from one lane to another, and to make the algorithm able to correct its direction and track the device movement.



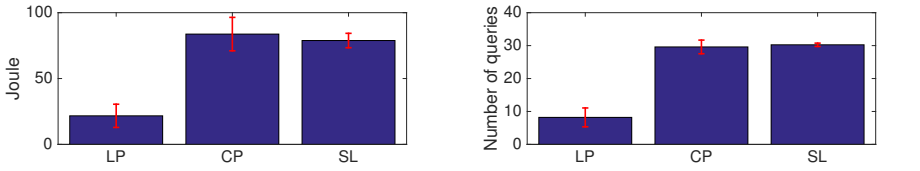
(a) Random path (RP)



(b) Circular path (CP)

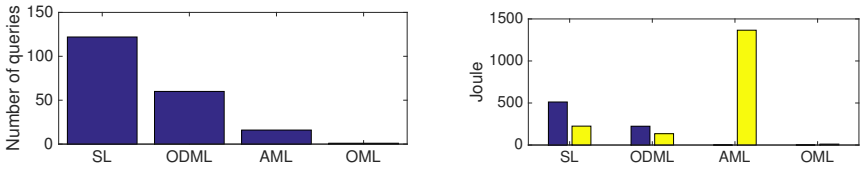
(c) Long-lines path (LP)

Figure 3.6: The red pointers show the predicted-queried paths by *Nuna* and the black arrows indicate the real taken paths



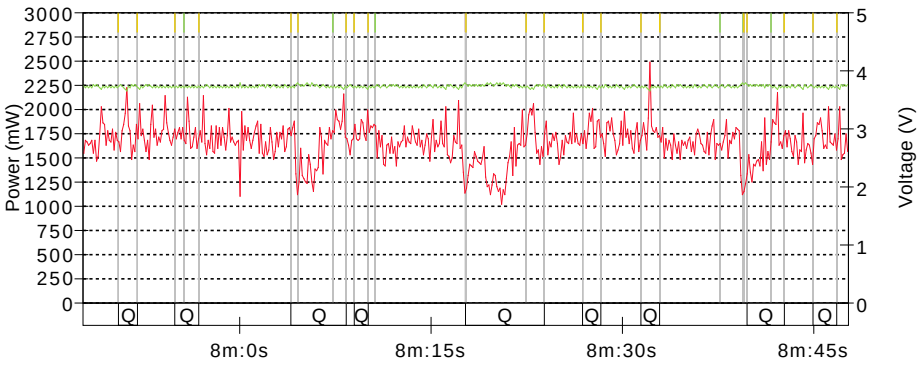
(a) Energy consumption (b) Number of generated WSDB queries

Figure 3.7: Number of generated queries and energy consumption of *Nuna* on a circular path (CP) and long lines path (LP), compared to querying the path using a single-location query technique.

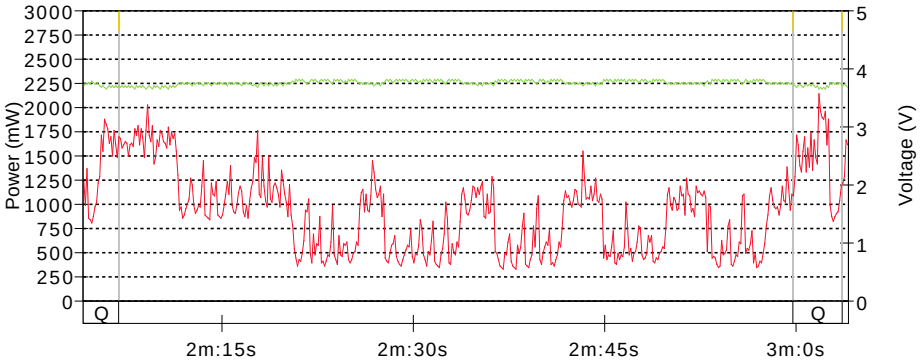


(a) Number of generated queries (b) Energy consumption

Figure 3.8: Comparing the number of generated queries and energy consumption of different multi-location WSDB query techniques and single-location queries for the RP path of 12 km. The blue bar represents the energy consumption with the GPS, while the yellow ones show the energy consumption without the GPS.



(a) Single-location query



(b) *Nuna*

Figure 3.9: A one-minute snapshot of power traces. Q denotes the query time

3.4.3 Evaluation

Nuna was tested using a Samsung Galaxy S3 running Android version 4.3.1, using a kpn cellular network. The test took place in Almere, the Netherlands[3]. We drove along one path where it approaches the best performance 3.6(c), and another where it approaches its worst performance 3.6(b) and a third random path 3.6(a) to give a representative picture.

Experiment Set-up

- We chose a long-line path (LP) 3.6(c) of 3 km. The top speed was 80 km/h and the number of iterations was five;
- We chose a circular path (CP) 3.6(b) of 3 km. The top speed was 40 km/h and the number of iterations was five;
- We chose a random path (RP) 3.6(a) of 12 km. The top speed was 80 km/h and the number of iterations were one.

Since *Nuna* always predicts the direction of movement along a straight line and then checks if there is a change in the direction, its best performance is along a straight line, and the worst is on a path of a constantly changing direction. Fig. 3.7 shows the number of generated queries and the energy consumed by these queries. Since the path length is 3 km, the required number of single-location queries is 30, which is represented in the third bar of Fig 3.7(b). The average number of generated queries along the LP is less than one third of the number generated along CP and SL.

From the energy perspective *Nuna* is much more efficient than single-location query technique on a straight-lines path, see Fig. 3.7(a). On the other hand, *Nuna* is marginally worse than a single-location query technique on the circular path. This is because multi-location queries are generated, but mostly only one location of the query is used, as shown in Fig. 3.6(b).

Fig. 3.8(b) shows the energy consumption of different WSDB querying techniques used to provide white spaces on the RP. The blue bars represent energy consumption when the GPS is on, whereas the yellow bars represent doing the same queries while the GPS is off. The way the yellow bars are generated is by extracting the size of the multi-location queries from the experiments when the GPS is on and letting the app do some queries without the need for location, speed and direction information. The distance between the farthest two points on the path was about 7 km. Therefore, applying AML required querying 15394 locations, which was done in 16 queries. The time of querying 1000 locations with a one multi-location query was over a minute. Finally, the OML technique was applied, where the entire path was

[3]Although we drove around Almere, all the coordinates were shifted in real time to the US.

queried in one multi-location query of 120 locations. The query consumed about 11 Joules.

Fig. 3.9 shows a one-minute snapshot of the power trace when *Nuna* and when single-location query technique are used to query GGL WSDB. Comparing the two traces shows the benefit of *Nuna*, and more generally of multi-location WSDB queries. Because the number of queries is reduced when *Nuna* is applied, the CPU may go into suspend mode in between the queries, meaning the power trace goes from high state to low state. The total energy consumption when the RP is queried in the single-location query technique is 1595 Joules, while applying *Nuna* reduces the energy consumption by 30%.

3.4.4 Limitations

- The minimum time *Nuna* needs to react to a change in the direction of movement is equal to the time to query WSDB and get a full response. In our implementation *Nuna* checks the direction every 8 s. This interval is an estimation for the longest query time. If a change in direction starts and ends within this interval, it will not be recognized by *Nuna*. Therefore, turning with relatively high speed might make *Nuna* exceed 100 m. To overcome this limitation, *Nuna* either has to check direction change on a shorter interval or query along three parallel lines when a device moves faster than a predefined threshold, i.e. 50 km/h.

- Because of the way *Nuna* predicts the direction of movement along a straight line and then checks if there is a change in direction, it is slightly less efficient than doing a single location query on a constantly changing path, i.e. a circular path. *Nuna* always generates a number of unused queried locations. However, the cost of the extra locations that were queried in terms of energy and time is marginal.

In this chapter we present our new WSDB query technique, *multi-locaiton WSDB query*. We show that it requires less frequent access to cover a particular path compared to the current query technique. Therefore, it saves energy and has a better mobility support. Furthermore, we introduce *Nuna*: a multi-location WSDB query algorithm, and show that it reduces the energy consumption, excluding the tail energy, and the number of queries by 50%.

Chapter 4

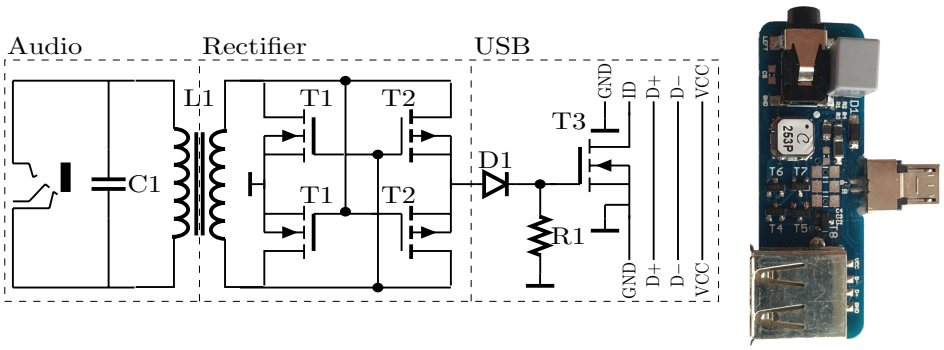
Portable Spectrum Sensing Platform

Many sensing applications use a combination of a smartphone and a USB-connected device, e.g. crowd-sourced radio spectrum sensing [5]. Unfortunately, USB devices deplete the battery of the phone. According to the USB specification, the USB device has to manage its own power consumption by going to a *suspended* state when possible or requested [82]. However, many USB devices do not support the suspend state [83]. Even if supported, a *suspended* USB device is still consuming energy [82]. Therefore, connecting a USB device to a smartphone, e.g. via a USB On-The-Go cable [84], can decrease a smartphone's battery life—both because USB device keeps the smartphone awake and because of the USB device's own consumption.

To alleviate the problem we introduce On-The-Go Switch (OTGS): a hardware/software platform that enables a smartphone to control via the audio port the *logical* connection state of a *physically* connected USB device. This approach limits both the duration at which the smartphone needs to act as USB master and the duration at which the USB device is consuming energy. Our proposed OTGS could work on any smartphone with USB OTG capabilities and a headphones connector without relying on custom kernels, rooted/jailbraked firmware's or even the operating system. Finally, we show how OTGS can reduce the overall power consumption of a radio spectrum sensing platform.

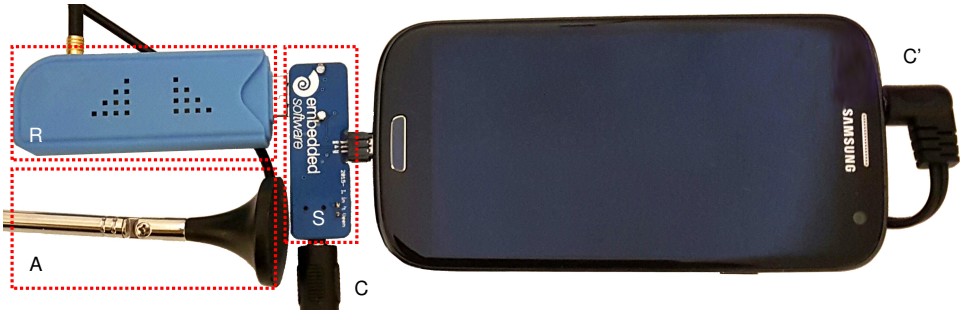
4.1 OTG Switch

Inspired by the design of [85] we propose OTG switch, which replaces the OTG cable connecting a smartphone and USB device. Following the USB standard [82] OTG cables use a fifth connection (called `ID` connection) within the OTG connector to signal their presence to a USB master. The OTGS simulates the physical connection and disconnection of the OTG cable by



(a) OTG switch, components: C1 = 1µF, **L1**: LPR6235, **T1**: ZSM61P03F, **T2**: ZSM61N03F, **D1**: DFLS120L-7, **T3**: FDV301N, R1 = 1MΩ

(b) OTG switch, PCB



(c) PoSSP hardware components: **R**: RTL-SDR dongle, **S**: OTG Switch, **A**: Antenna. Not fully shown: **C** and **C'** are connected with an audio cable.

Figure 4.1: PoSSP platform: (a) OTGS schematics; (b) OTGS; and (c) system overview.

switching this `ID` connection between pull-down and its default pull-up state, see Figure 4.1(a). The `ID`-connection, while in pull-down state, will enable the phones USB master mode to provide power and communication to the attached USB device. On a hardware level, the switching is done by a FET transistor that is controlled from the phone's headphone output. To enable the OTGS the attached smartphone plays an audio signal through its headphone connector, this waveform is rectified on the OTGS to provide a suitable signal to switch the FET. The audio signal outputted by the phone is a 15 kHz sinusoid that proved to provide the highest amplitudes (hence, most energy) on most of the tested smartphones. The OTGS needs around one second to stabilize after changing state. The combination of the large drain resistor and parasitic input capacitance mitigates the need for an extra storage capacitor.

Table 4.1: PoSSP Power Consumption per Component

	Wave	Audio	USB	Dongle	Power (mW)
$P_{\text{p,sleep}}$	—	—	—	—	390 [1]
$P_{\text{p,on}}$	—	—	—	—	789
$+P_{\text{wave}}$	on	headphones	—	—	870
$+P_{\text{otgs}}$	on	OTGS	OTGS	—	1015
$+P_{\text{usb}}$	on	OTGS	OTGS	connected	1557[2]

[1] Average power consumption for the phone while it is in a low power state in the non-sensing period.
[2] Average power consumption of four RTL-SDR dongles.

Table 4.2: Power consumption of Tuners, Chipsets and RTL-SDR dongles

Type	Tuner	Chipset	No.	P_{usb}	σ
USB HDTV Stick	E4000	RTL2832U	1	454	n/a
ezcap	FC0013	RTL2832U	1	434	n/a
NooElec	R820T2	RTL2832U	4	662	18
DVB-T+DAB+FM	R820T	RTL2832	4	615	12

4.2 Energy consumption of PoSSP

In this section, we introduce our Portable Spectrum Sensing Platform (PoSSP), analyze its power consumption and show how this can be significantly reduced by the OTGS.

Our PoSSP platform setup consists of a *Samsung Galaxy S3* running *Android 4.3.1*, our OTGS, an RTL-SDR dongle [86], and an antenna, see Figure 4.1(c). We also developed a spectrum sensing app using `rtl_power` [87], which periodically performs a sensing operation, obtains the GPS position and uploads the output to an online server.

4.2.1 Energy Consumption of Components

The power consumption of the PoSSP components is measured using a Monsoon [88] power monitor set to a voltage of 4.19 V. The device under test is an Android smartphone in *airplane mode* with the screen enabled. This phone is running a custom app, generating a 15 kHz audio signal.

Table 4.1 shows power measurements for different combinations of PoSSP components. These results show that the audio generator consumes 81 mW while the OTGS adds another 145 mW. This increase in energy consumption, while the switch is enabled, is lower than the amount of energy otherwise wasted by keeping the radio dongle continuously enabled. Furthermore, measurements show that not all RTL-SDR dongles have the same energy consumption, see Table 4.2.

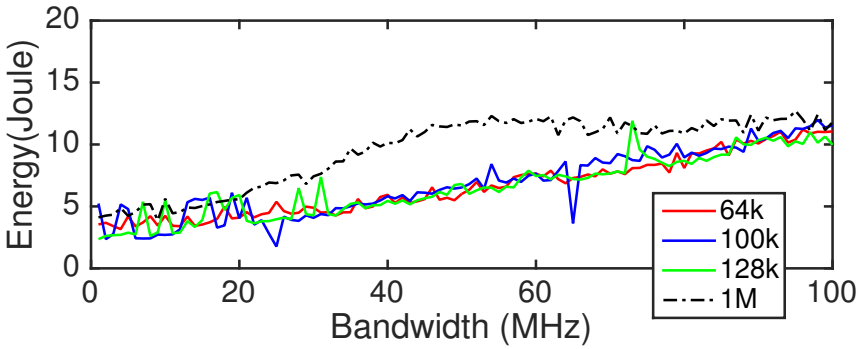


Figure 4.2: Power consumption of different frequency ranges (bandwidth). The bandwidth starts with 1 MHz and ranges up to 100 MHz with a step of 1 MHz. The experiment was repeated with different FFT-bin sizes.

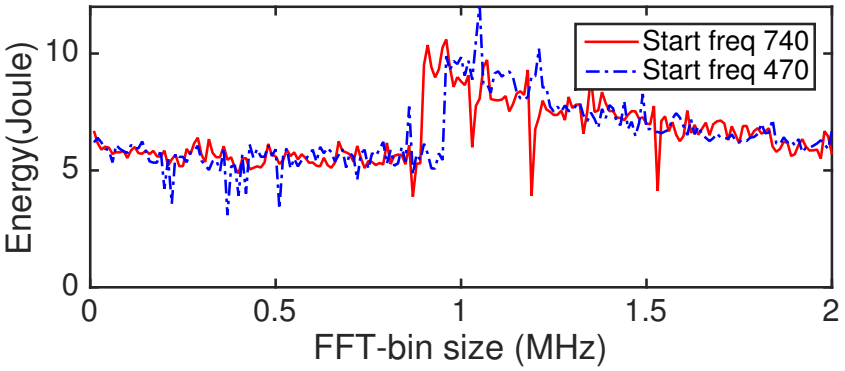


Figure 4.3: Energy consumption of different FFT sizes. FFT size ranges from 10 kHz to 210 kHz. The bandwidth is fixed to 40 MHz.

4.2.2 Energy Consumption Reduction

In this experiment, we use NEAT [81] It enables us to extract the exact amount of consumed energy during a sensing operation, by overlaying the phone events on the phone's power trace. Furthermore, we chose to use `rtl_power` [87], a command line tool, to control the RTL-SDR dongle because it is freely available on the Internet, used by other researchers [5], and can easily be integrated in an Android app. Moreover, `rtl_power` enables a user to scan the spectrum with different FFT-bin sizes and bandwidths, start and stop frequencies.

The results in Fig. 4.2 show that increasing the frequency bandwidth is indeed increasing the energy consumption. However, increasing the frequency range from 1 MHz to 100 MHz costs only 5 J more. Furthermore, we observe, see Fig 4.3, that the FFT-bin size below 1MHz does not alter the energy consumption of PoSSP significantly. in contradiction, the FFT-bin size

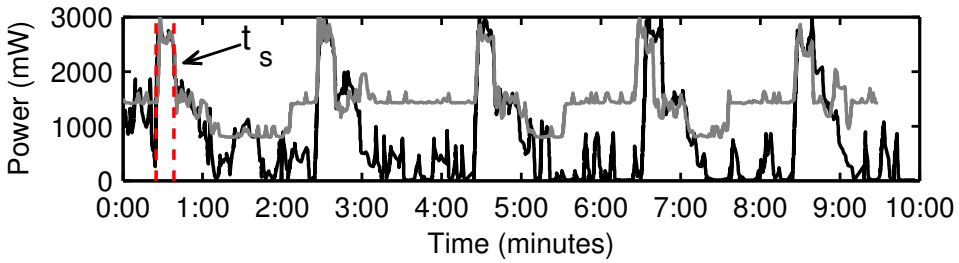


Figure 4.4: Example NEAT power traces of PoSSP, top (light) line shows consumption without, bottom (dark) line shown with OTGS. The sensing period for both traces is marked with t_s, see Section 4.2.2 for details.

of 1 MHz and above does change the energy consumption of the spectrum sensing.

In order to understand why the spectrum sensing peaks when the FFT-bin size is 1 MHz, we need to understand how the `rtl_power` does work. `rtl_power` divides the specified, by the user, bandwidth to smaller bands, then it hops from one band to another to extract (sense) the energy of the frequency bands.

When the FFT-bin size is below 1 MHz `rtl_power` always divides the specified bandwidth to an equal number of frequency bands independent of the FFT-bin size. However, when the FFT-bin size is exactly 1 MHz `rtl_power` divides the bandwidth by 1 MHz which is in turn the smallest frequency band size that `rtl_power` divides the bandwidth by. Therefore, it requires the largest number of hops. As a result, it consumes the greatest amount of energy.

Figure 4.4 shows the power traces of PoSSP while doing a periodic sensing operations every two minutes with, and without OTGS. A sensing operation takes ≈ 20 s, and it consumes ≈ 34 J. OTGS, on one hand, adds two extra joules to the cost of a sensing operation, and, on the other hand, reduces the energy consumption of a non-sensing period from 130 J to 51 J . Moreover, OTGS will save more energy if the interval between two sensing operations is increased.

4.2.3 Energy Consumption Model

OTGS allows a smartphone to go into sleep mode after a sensing operation by powering off the OTG-device, thus saving energy. To calculate the amount of saved energy we propose the following model: $E(t_{\text{total}}) = (t_{\text{total}}/t_{\text{p}})\left((P_{\text{usb}} + P_{\text{p,on}} - P_{\text{p,sleep}})\,(t_{\text{p}} - t_{\text{s}}) - (P_{\text{wave}} + P_{\text{otgs}})\,t_{\text{s}}\right)$, where E is the amount of saved energy in an experiment of duration t_{total} with t_{s} as sensing duration repeated with period t_{p}. P_{usb}, $P_{\text{p,on}}$, $P_{\text{p,sleep}}$, P_{wave} and P_{otgs} are the amount of energy for the USB connection, phone being awake, phone

being asleep, wave generator software, and the switch hardware, respectively. See Table 4.1 for typical values.

In this chapter we introduced our Potable Spectrum Sensing platform. The energy consumption of PoSSP is analyzed. We show that PoSSP energy consumption for FFT-bin size below 1MHz is independent of the FFT-bin size. However, when the FFT-bin size is 1 MHz PoSSP consumes the maximum amount of energy, assuming other parameters are fixed and the energy consumption decreases when we increase the FFT-bin size above the 1 MHz. Finally, OTGS is introduced, and we showed that it reduces the energy consumption of PoSSP substantially.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This study was focused on White Space Database (WSDB) and was done from a Secondary User (SU) perspective. In general, this study can be divided into two main parts:

- Comparative study between WSDBs. In Section 2.1, we compared the performance of seven WSDBs. Specifically we compared the response message size and the response time. The results show that Google WSDB (GGL) is the fastest responding WSDB, while SpectrumBridge WSDB (SBI) replies with the smallest message size. We observed that there is a dependency between a location and the WSDB response time. Furthermore, we concluded that the state-of-the-art WSDB query technique offers poor support for mobility, causing a mobile device to access a WSDB very frequently, which consumes a relatively high amount of energy.

- Optimization process, which can be divided into two processes:

  - WSDB mobile device access optimization: Based on our conclusion from the comparative study, we proposed a new WSDB query technique that offers better mobility support than the current query technology, called *mulit-location WSDB query*. As its name suggests, it enables a user to query a group of locations in one request. Furthermore, we showed that this technique can reduce the number of required queries significantly, and thus save a considerable amount of energy. Based on the multi-location query method, we proposed a WSDB query algorithm denoted as *Nuna* for mobile devices. We implemented *Nuna* on an Android-based

mobile device (the app contains more than 1500 lines of code, and the algorithm alone is about 400 lines of code). Furthermore, the evaluation of *Nuna* was shown and compared with other WSDB query techniques (we drove over 300 km to test, optimize and evaluate *Nuna*). We showed that *Nuna* reduced the number of queries and the energy consumption to half that of state-of-the-art query techniques.

- WSDB information optimization: As a first step towards providing a WSDB with local spectrum reading, we have developed an energy-efficient Portable Spectrum Sensing Platform (PoSSP). PoSSP consists of an RTL-SDR dongle, an Android-based smartphone, an in-house-developed On-The-Go switch (OTGS), antenna and a custom Android app (with more than 1100 lines of code) that is able to sense the spectrum, get its GPS position and upload the data to an online PHP server. OTGS enabled us to turn the RTL-SDR dongle on/off periodically, thereby saving energy by powering off the dongle when it was not in use and by allowing the smartphone to go to sleep during a non-sensing period.

5.2 Future Work

We spit the future work as follows:

- Multi-location WSDB query:
  - The next logical step for a mobile device is to communicate over the obtained white space. However, an algorithm is needed to process a WSDB response message and to extract the best parameters for communication, i.e. the frequency band with the highest transmission power.
  - Building a secondary server. Its function is to answer the following question. Given a particular path by a mobile device, what is the minimum set of white spaces (frequencies) that cover the entire path?
- Nuna:
  - Reducing the dependency of *Nuna* on the GPS. A way to achieve this goal is by using a local sensor and enforce its estimation with the GPS positioning. This will help in reducing the energy consumption and make the algorithm work reasonably well when the GPS signal is lost.
- Portable Spectrum Sensing Platform: The Portable Spectrum Sensing Platform (PoSSP) can serve different purposes:

– Replacing the RTL-SDR with, for instance, hackRf [89], to support transmission mode. Once this upgrade happens, we can study spectrum management between mobile SUs based on local sensing and WSDB. In this scenario, the WSDB is responsible for protecting the licensed services and the local sensing technique is used to manage the white space spectrum between mobile SUs.
– PoSSP can be used to make an energy map. That can help, for example, in finding the best places for energy-harvesting devices.

Bibliography

[1] M. Song, C. Xin, Y. Zhao, and X. Cheng, "Dynamic spectrum access: from cognitive radio to network radio," *Wireless Communications, IEEE*, vol. 19, pp. 23–29, February 2012.

[2] M. A. McHenry, P. A. Tenhula, D. McCloskey, D. A. Roberson, and C. S. Hood, "Chicago spectrum occupancy measurements & analysis and a long-term studies proposal," in *Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum*, TAPAS '06, (New York, NY, USA), ACM, 2006.

[3] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127 – 2159, 2006.

[4] "Protocol to access white-space (PAWS) databases." `http://tools.ietf.org/html/draft-ietf-paws-protocol-12.txt`, (2015).

[5] A. Nika, Z. Zhang, X. Zhou, B. Y. Zhao, and H. Zheng, "Towards commoditized real-time spectrum monitoring," HotWireless '14, (New York, NY, USA), ACM, 2014.

[6] "Report of the spectrum efficiency." `https://transition.fcc.gov/sptf/reports.html`.

[7] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper." `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html/`.

[8] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 201–220, 2005.

[9] J. Mitola, "Cognitive radio for flexible mobile multimedia communications," in *Mobile Multimedia Communications, 1999. (MoMuC '99) 1999 IEEE International Workshop on*, pp. 3–10, 1999.

[10] J. Mitola, "Cognitive radio—an integrated agent architecture for software defined radio," 2000.

[11] S. Geirhofer, L. Tong, and B. Sadler, "Cognitive radios for dynamic spectrum access - dynamic spectrum access in the time domain: Modeling and exploiting white space," *Communications Magazine, IEEE*, vol. 45, pp. 66–72, May 2007.

[12] Q. Zhao and A. Swami, "A survey of dynamic spectrum access: Signal processing and networking perspectives," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, pp. IV–1349–IV–1352, April 2007.

[13] T. Yücek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 1, pp. 116–130, 2009.

[14] L. Xu, R. Tonjes, T. Paila, W. Hansmann, M. Frank, and M. Albrecht, "Drive-ing to the internet: Dynamic radio for ip services in vehicular environments," in *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*, pp. 281–289, 2000.

[15] D. Hatfield and P. Weiser, "Property rights in spectrum: taking the next step," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 43–55, Nov 2005.

[16] L. Gao, X. Wang, Y. Xu, and Q. Zhang, "Spectrum trading in cognitive radio networks: A contract-theoretic modeling approach," *Selected Areas in Communications, IEEE Journal on*, vol. 29, pp. 843–855, April 2011.

[17] R. Mackenzie, K. Briggs, P. Gronsund, and P. Lehne, "Spectrum microtrading for mobile operators," *Wireless Communications, IEEE*, vol. 20, pp. 6–13, December 2013.

[18] W. Lehr and J. Crowcroft, "Managing shared access to a spectrum commons," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 420–444, Nov 2005.

[19] J. Bae, E. Beigman, R. Berry, M. Honig, and R. Vohra, "Incentives and resource sharing in spectrum commons," in *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pp. 1–10, Oct 2008.

[20] B. Kwon and J. Copeland, "A centralized spectrum sharing scheme for access points in ieee 802.11 networks," in *Consumer Communications*

and Networking Conference, 2009. CCNC 2009. 6th IEEE, pp. 1–5, Jan 2009.

[21] B. A. Witvliet, M. J. Bentum, R. Schiphorst, and C. H. Slump, "Medium usage model for the design of dynamic spectrum management in ism bands," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 4044–4048, IEEE, 2012.

[22] S. Atapattu, C. Tellambura, and H. Jiang, *Energy Detection for Spectrum Sensing in Cognitive Radio*. Springer, 2014.

[23] M. Song, C. Xin, Y. Zhao, and X. Cheng, "Dynamic spectrum access: from cognitive radio to network radio," *Wireless Communications, IEEE*, vol. 19, pp. 23–29, February 2012.

[24] Y. Yuan, P. Bahl, R. Chandra, P. Chou, J. I. Ferrell, T. Moscibroda, S. Narlanka, Y. Wu, *et al.*, "Knows: Cognitive radio networks over white spaces," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pp. 416–427, IEEE, 2007.

[25] D. Cabric, S. Mishra, and R. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 1, pp. 772–776 Vol.1, Nov 2004.

[26] A. Ghasemi and E. Sousa, "Optimization of spectrum sensing for opportunistic spectrum access in cognitive radio networks," in *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pp. 1022–1026, Jan 2007.

[27] P. Pawelczak, G. Janssen, and R. Venkatesha Prasad, "Wlc10-4: Performance measures of dynamic spectrum access networks," in *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pp. 1–6, Nov 2006.

[28] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proceedings of the IEEE*, vol. 55, pp. 523–531, April 1967.

[29] T. Yücek and H. Arslan, "Spectrum characterization for opportunistic cognitive radio systems," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–6, IEEE, 2006.

[30] D. C. J. G. Proakis, *Digital Communications*. McGrill-Hill, 2001.

[31] R. Tandra and A. Sahai, "Fundamental limits on detection in low snr under noise uncertainty," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, vol. 1, pp. 464–469, IEEE, 2005.

[32] J. Lundén, V. Koivunen, A. Huttunen, and H. V. Poor, "Spectrum sensing in cognitive radios based on multiple cyclic frequencies," in *Cognitive Radio Oriented Wireless Networks and Communications, 2007. CrownCom 2007. 2nd International Conference on*, pp. 37–43, IEEE, 2007.

[33] T. Ratnarajah, "Eigenvalue-based spectrum sensing for cognitive radio," in *Cognitive Radio Communications: European Activities and Progress, IET Seminar on*, pp. 1–25, IET, 2010.

[34] A. Sahai, R. Tandra, S. M. Mishra, and N. Hoven, "Fundamental design tradeoffs in cognitive radio systems," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*, p. 2, ACM, 2006.

[35] B. Wild and K. Ramchandran, "Detecting primary receivers for cognitive radio applications," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 124–130, IEEE, 2005.

[36] "Initial evaluation of the performace of prototpye tv-band white space devices, oet report fcc/oet 07-tr-1006, fcc oet, july 2007."

[37] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar, "Coalitional games for distributed collaborative spectrum sensing in cognitive radio networks," in *INFOCOM 2009, IEEE*, pp. 2114–2122, April 2009.

[38] "Google white Space database." `https://www.google.com/get/spectrumdatabase/`, (2015).

[39] A. B. Flores, R. E. Guerra, E. W. Knightly, P. Ecclesine, and S. Pandey, "Ieee 802.11 af: a standard for tv white space spectrum sharing," *Communications Magazine, IEEE*, vol. 51, no. 10, pp. 92–100, 2013.

[40] "Regulatory requirements for white space devices in the uhf tv band." `http://www.cept.org/Documents/se-43/6161/`, 2012.

[41] "En 301 598 white space devices (wsd); wireless access systems operating in the 470 mhz to 790 mhz frequency band," 2012.

[42] "Title 47, part 15, subpart h, television band devices." `http://www.ecfr.gov/`.

[43] "Statement on cognitive access to interleaved spectrum." `http://stakeholders.ofcom.org.uk/binaries/consultations/cognitive/statement/statement.pdf`, 2009.

[44] R. Thanki, "The economic significance of licence-exempt spectrum to the future of the internet," *White Paper*, 2012.

[45] D. Gurney, G. Buchwald, L. Ecklund, S. Kuffner, and J. Grosspietsch, "Geo-location database techniques for incumbent protection in the tv white space," in *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*, pp. 1–9, Oct 2008.

[46] "SpectrumBridge white space database." `https://spectrumbridge.com/`, (2015).

[47] "Nominet white space database." `http://www.nominet.org.uk/`, (2015).

[48] F. Paisana, J. P. Miranda, N. Marchetti, and L. A. DaSilva, "Database-aided sensing for radar bands," in *Proc IEEE DySPAN*, 2014.

[49] "3.5 ghz spectrum access system workshop." `http://www.fcc.gov/events/35-ghz-spectrum-access-system-workshop`, 2014.

[50] M. Matinmikko, H. Okkonen, M. Palola, S. Y. P. Ahokangas, and M. Mustonen, "Spectrum sharing using licensed shared access: The concept and its workflow for LTE-advanced networks," vol. 21, no. 2, pp. 72–79, 2014.

[51] P. Palka, "Future terrestrial broadcast systems," *IEEE Consumer Electron. Mag.*, vol. 2, no. 3, pp. 17–24, 2013.

[52] R. Murty, R. Chandra, T. Moscibroda, and P. Bahl, "Senseless: A database-driven white spaces network," in *Proc IEEE DySPAN*, 2011.

[53] X. Ying, J. Zhang, L. Yan, G. Zhang, M. Chen, and R. Chandra, "Exploring indoor white spaces in metropolises," in *Proc. ACM MobiCom*, 2013.

[54] M. Palola, M. Matinmikko, J. Prokkola, M. Mustonen, M. Heikkila, T. Kippola, S. Yrjola, V. Hartikainen, L. Tudose, A. Kivinen, J. Paavola, and K. Heiska, "Live field trial of licensed shared access (lsa) concept using lte network in 2.3 ghz band," in *Dynamic Spectrum Access Networks (DYSPAN), 2014 IEEE International Symposium on*, pp. 38–47, April 2014.

[55] J. Huang, Q. Xu, and B. Tiwana, "Anatomizing application performance differences on smartphones," in *Proc. ACM MobiSys*, 2010.

[56] L. Zhu, V. Chen, J. Malyar, S. Das, and P. McCann, "Protocol to access white-space (paws) databases," 2015.

[57] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. ACM IMC*, 2009.

[58] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3G networks," in *Proc. ACM IMC*, 2010.

[59] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, 2012.

[60] N. Ding, D. Wagner, X. Chen, A. Pathak, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proc. ACM SIGMETRICS*, 2013.

[61] N. Vallina-Rodriguez, A. Aucinas, M. Almeida, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft, "Rilanalyzer: a comprehensive 3G monitor on your phone," in *Proc. ACM IMC*, 2013.

[62] J. Du and M. Barth, "Next-generation automated vehicle location systems: Positioning at the lane level," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, pp. 48–57, March 2008.

[63] G. Sun, J. Chen, W. Guo, and K. Liu, "Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs," *Signal Processing Magazine, IEEE*, vol. 22, pp. 12–23, July 2005.

[64] E. Costa, "Simulation of the effects of different urban environments on gps performance using digital elevation models and building databases," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, pp. 819–829, Sept 2011.

[65] H. Abbott and D. Powell, "Land-vehicle navigation using gps," *Proceedings of the IEEE*, vol. 87, pp. 145–162, Jan 1999.

[66] Q. Vo and P. De, "A survey of fingerprint based outdoor localization," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[67] "Microsoft white space database." `http://whitespaces.msresearch.us/`, (2015).

[68] "SpectrumBridge-Ofcom white space database." `http://www.ofcom.org.uk/`, (2015).

[69] "Csir white space database." `http://whitespaces.meraka.csir.co.za/PawsService`, (2015).

[70] "Fairspectrum white space database." `https://fswsdb.com:443/wsd/index.php`, (2015).

[71] "Draft SBI-PAWS Protocol." https://tools.ietf.org/html/draft-sbi-paws-protocol-00, (2015).

[72] "Networking Over White Spaces (KNOWS)." http://research.microsoft.com/en-us/projects/KNOWS/, (2015).

[73] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 0.80 ed., May 2014.

[74] "Google remote procedure call server." https://www.googleapis.com/rpc/.

[75] "TV transmitter finder." http://www.antennapoint.com/, (2015).

[76] "Federal Communications Commission." http://www.fcc.gov/, (2015).

[77] N. Brouwers, M. Zuniga, and K. Langendoen, "Neat: A novel energy analysis toolkit for free-roaming smartphones," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, SenSys '14, (New York, NY, USA), pp. 16–30, ACM, 2014.

[78] N. B. N. Przemysław Pawełczak, Nihan Cicek and K. Langendoen, "Will dynamic spectrum access drain my battery?."

[79] https://developers.google.com/spectrum/paws/gettingstarted, 2015. Google Spectrum Database API.

[80] "Google spectrum database api discussion group." https://groups.google.com/forum/#!topic/google-spectrum-db-discuss/mv62FjK5bYc, (2015).

[81] N. Brouwers, M. Zuniga, and K. Langendoen, "Neat: A novel energy analysis toolkit for free-roaming smartphones," SenSys '14, (New York, NY, USA), ACM, 2014.

[82] "USB specification version 2.0." http://www.usb.org/developers/docs/usb20_docs.

[83] "The linux kernel archives." https://www.kernel.org/doc/Documentation/usb/power-management.txt.

[84] "Usb on-the-go and embedded host." http://www.usb.org/developers/onthego/.

[85] Y.-S. Kuo, S. Verma, T. Schmid, and P. Dutta, "Hijacking power and bandwidth from the mobile phone's audio interface," ACM, 2010.

[86] "Rtl sdr." http://sdr.osmocom.org/.

[87] "Rtl power basic discription." http://kmkeen.com/rtl-power/.

[88] "Monsoon: Power meter." https://www.msoon.com/LabEquipment/PowerMonitor/.

[89] "Software defined radio with HackRf." https://greatscottgadgets.com/sdr/, (2015).