An Evaluation of Smartphone Resources Used by Web Advertisements

by

Abdurhman Albasir

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

With the rapid advancement of mobile devices, people have become more attached to them than ever. This rapid growth combined with millions of applications (apps) make smartphones a favourite means of communication among users. In general, the available contents on smartphones, apps and the web, come into two versions: (i) free contents that are monetized via advertisements (ads), and (ii) paid ones that are monetized by user subscription fees. However, the resources (energy, bandwidth, processing power) on-board are limited, and the existence of ads in either websites or free apps can adversely impact these resources. These issues brought the need for good understanding of the mobile advertising eco-system and how such limited resources can be efficiently used.

This thesis focuses on mobile web browsing. Surfing web-pages on smatphones is one of the most commonly used task among smartphone users. However, web-page complexity is increasing, especially when designed for desktop computers. On one hand, the existence of ads in web-pages is essential for publishers' monetization strategy. On the other hand, their existence in webpages leads to even higher complexity of the webpages. This complexity in the smartphone environment, where the battery and bandwidth resources are limited, is reflected in longer loading time, more energy consumed, and more bytes transferred. With this view, quantifying the energy consumption due to web ads in smartphones is essential for publishers to optimize their webpages, and for system designers to develop an energy-aware applications (browsers) and protocols. Apart from their energy impact, ads consume network bandwidth as well. Therefore, quantifying the bandwidth consumption due to downloading web ads is crucial to creating more energy and bandwidth aware applications. This thesis first classifies web content into: (i) core information, and (ii) forced "unwanted" information, namely ads. Then, describes an approach that enables the separation of web content in a number of a websites. Having done so, the energy cost due to downloading, rendering, and displaying web ads over Wi-Fi and 3G networks is evaluated. That is, how much energy web ads contribute to the total consumed energy when a user accesses the web. Furthermore, the bandwidth consumed by web ads in a number of well-known websites is also evaluated.

Motivated by our findings about ads' impact on the energy and bandwidth, the thesis proposes and implements a novel web-browsing technique that adapts the webpages delivered to smartphones, based on a smartphone's current battery level and the network type. Webpages are adapted by controlling the amount of ads to be displayed. Validation tests confirm that the system, in some cases, can extend smartphone battery life by up to $\sim$ 30% and save wireless bandwidth up to $\sim$ 44%.

Acknowledgements

All praise is to Allah for giving me the ability and knowledge to accomplish this thesis.

Several people deserve sincere recognition. I am grateful to my advisor, Prof. Kshirasagar Naik, for his guidance, support, patience, and encouragement. I would like also to extend my appreciation and thanks to my thesis committee members, Prof. Liang-Liang Xie and Prof. Pin-Han Ho for sparing their time in reviewing my thesis.

I would like also to acknowledge the Ministry of Higher Education of Libya for financially supporting this research work.

Thanks go to my friends for their prayers and support. I am very fortunate to have so many exceptional and genuine people in my life. I thank my friends and colleagues, Tarek Alwazini, Maazen AlSabaan, Tareq Abdunabi, Abdulhakim Abogharaf, Majid Altamimi, Mahdi Alghazali, Elfetori Ibrahim and Mohamed Elmassad for all our fruitful discussions and for their friendship.

Last and by far not least, I am indebted to my parents, brothers, and sisters for their continuous support, patience, and prayers. I owe a great deal of gratitude to my wife, Khawla, and my daughter, Jori, for their continuous understanding, unlimited encouragement, and unending love during the years of my study.

Dedication

to my beloved dad, Prof. Ali Elbasir

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The past decade has witnessed a huge growth in smartphones' popularity. People all over the world now tend to have smartphones and the number of mobile phones in the market has been estimated to be almost equal to the number of people worldwide in 2013 [6]. A significant percentage of these mobile subscribers already have a smartphone or are planning to get one. A report published by ComScore in September 2012 stated that over 115 million people in the United States owned smartphones out of 234 million total subscribers [16]. This statistic shows a penetration of close to 50%. Google Android is currently the top ranked with 52.2%, and Apple comes after with 34.3%. Such growth and popularity is driven by the vast variety of applications (apps) now available. This rapid growth of smartphones combined with millions of apps make them a favourite means of communication among users, but their functionalities and development are constrained by a number of challenges (Battery life, Bandwidth, and Security and Privacy.)

The volume of wireless traffic continues to grow exponentially. Web browsing and network related applications (NRAs) rely entirely on the Internet for their operation. They are considered the main contributors to the ever-growing wireless traffic. Since the traffic generated by smartphones forms a new addition to wireless networks, the demand for studies that investigate the nature of smartphone traffic has emerged [21] [40]. These research studies are answering interesting questions, such as (i) how much traffic a device generates per day; (ii) which app contributes the most in the total smartphone-generated traffic; and (iii) which app utilises the most network resources. The results can be used by mobile operators to improve their wireless quality of services (QoS) [14] [32] [35]. Some studies have broken down the traffic generated by smartphones to the app level, to explore how much traffic is generated by each app. However, more questions remain unanswered. For example, in mobile web browsing, what is the bandwidth cost of delivering certain

information to an end user? More specifically, how many bytes are required for downloading informations that is not of user interest? Thus, more investigation is needed, and more solutions that utilize this scarce resource efficiently are of great importance.

Smartphones use Lithium-ion batteries due to their high energy densities. However, there is no tangible evidence that in the near future more energy can be packed into the small space available on today's modern hand-held devices [45]. In addition, more powerful processors, high resolution displays, huge numbers of applications, and the advanced wireless technologies (LTE, WiMAX) are employed in today's smartphones and tablets. Such technologies and components drain energy at very high rates and are considered as battery hungry. For instance, the authors of [65] found that smartphone components, namely, CPU, Wi-Fi, LCD, GPS, and 3G links, consume up to, 28.6, 24.6, 85.9, 33.6, and 36.5 Joules, respectively. Moreover, the energy cost of sending a simple SMS message and placing a call through 3G is more energy expensive than using GSM, according to [45]. Although much research has focused on their battery life and the energy consumption in smartphones, solutions that target the hardware design as well as software have yet to emerge. Since software solutions are easier to implement and flexible to customize, they are more desirable and applicable to developers.

1.1 Motivation

The subject of this thesis was firstly motivated by two facts:

1. *The high growing popularity of mobile web browsing.* Recent statistics show that users prefer to use web browsers for digital news delivery. According to a survey conducted in U.S in 2012, 61% of smartphone users (and 60% of tablets users) use mainly browsers for reading news [36]. Thus, browsers are considered to be one of the most used smartphone apps.

2. *The increasing complexity of webpages.* A study done by [63] shows that over half of the pages are not optimized for mobile browsers. Although many websites have mobile versions of their webpages, people tend to view the full/desktop versions for the richer offered content. And despite the fact that some well known website companies have already created mobile versions of their websites, their sub-pages are actually desktop versions (full versions). These webpages are getting more complex [10, 68], in other words, more computationally intensive, and the existence of ads in webpages leads to even higher complexity. This complexity in smartphone environments where

the resources (*e.g.,* battery and bandwidth) are limited is reflected in longer loading times, more energy consumed, and more bytes transferred.

Therefore with these facts in mind, we were first motivated to study the energy and bandwidth impacts of ads in webpages. Then, after studying the impact of web ads on smartphone battery life and bandwidth, we were motivated by the measurements and the findings obtained in section 3.3 to design a system that is energy and bandwidth efficient. Because ads can drain considerable energy during mobile web browsing (in some web-pages it reaches 18% of the total energy) and because bandwidth overhead caused by downloading ads in some webpages reaches 50%, we propose a novel mobile browsing technique Smart Mobile Web Browsing (SWB) that is a function of the smartphone resource and the number of ads allowed to be displayed on webpages. SWB mainly targets (i) extending smartphone battery life; and (ii) preserving the bandwidth needed to download webpages. As mentioned in the previous section, having ads displayed in webpages is essential for having free web content; therefore, our approach aims to balance the satisfaction of end users as well as the webpage owners (publishers.)

1.2 Problem Statement

Web browsing is becoming essential for everyday life, especially with the ever-increasing popularity of smartphones. The web content delivered to end users has witnessed drastic changes. Webpages used to be simple "traditional" static pages compromised of only text and some images. However, modern webpages are dynamic and media-rich. The owners of these pages (publishers), moreover, rely heavily on ads as a source of revenue. These ads are very media-rich and can consume a lot of resources, like battery and bandwidth. Entirely eliminating ads from webpages is not practical, in view of the big role they play in the web eco-system and in the process of having free web content. These factors have led to the demand for mobile browsing solutions that adapt pages in a way that is energy and bandwidth efficient.

When a webpage is loaded on a user's screen, what actually is downloaded are the core information that a user is interested in (such as news, emails, stocks, etc.), and extra enforced "unwanted" information that comes up in the form of web advertisements (ads). Figure. 1.1 shows an example of the type of content that is downloaded and displayed on a user's screen. In a smartphone environment where resources are limited, displaying ads on the screen is not an inexpensive task from the energy and bandwidth perspectives. To be displayed, an ad needs to be fetched, rendered, and finally displayed. Each of these actions requires specific resource: network access (radio interface and bandwidth), certain

Figure 1.1: Sample website showing the core information and some ads (marked in red)

CPU computations, and graphics. Figure. 1.2 illustrates the extra needed steps to display the ads. These extra requests are reflected as an increase of battery consumption.

The main problem explored in this thesis is as follows. The complexity of webpages is increasing, especially if the webpages are designed for desktop computers. The existence of advertisements (ads) in webpage leads to even higher complexity. This complexity in a smartphone environment, where resources are limited (*e.g.,* battery and bandwidth) is reflected in longer loading time, more energy consumed, and more bytes transferred. Therefore, evaluating resources used by web advertising is very important to smartphone users as well as to web designers. Thus, this thesis focuses on quantifying the energy and bandwidth cost due to web ads in smartphones and on developing a way to limit this cost. Based on the measurements and analysis performed, the question of how smartphone battery life can be extended by controlling the number of ads to be displayed has emerged, as well as how the bandwidth needed to download webpages can minimized.

None of the studies surveyed in our literature review focus specifically on measuring the resources used in mobile devices by advertisements' overhead during mobile web browsing. Most address only the issue in mobile applications, and some study the cost of ads on PCs or laptops but not on mobile devices. Therefore and to the best of our knowledge, this

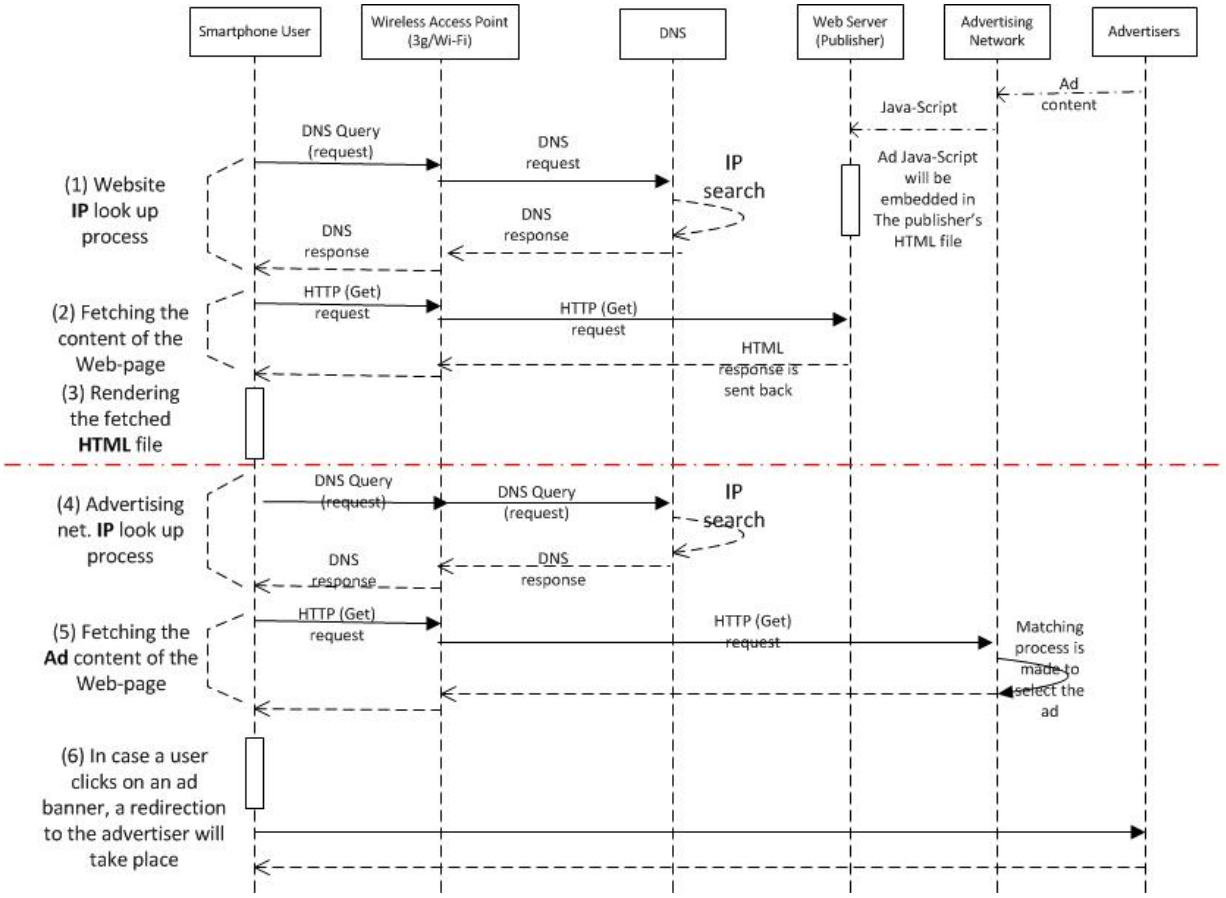


Figure 1.2: Sequence diagram of downloading webpage

study is the first that analyses mobile resources usage due to displaying ads on webpages based on real measurements. Also, although a number of attempts have tried to enhance mobile web browsing, no existing studies explore how web contents can be adapted just by controlling the number of ads allowed in webpages, so as to maximize the battery life of smartphone and minimize the required bandwidth.

1.3 Solution Strategy and Contributions

The strategies used to maintain the goals of this thesis are briefly explained in the following points. More details about these solutions strategies are provided in Chapters 3 and 4.

1. *Measuring Ads' Energy and Bandwidth Cost*: To measure exactly the energy and bandwidth cost of web ads, an approach to separate what we call core information and overhead "unwanted" information (ads) was needed. We decided to download a number of websites and have our own web server. These allowed us to enable and disable the delivery of ads as needed. Then, a number of test cases were conducted to measure the energy and bandwidth costs of web ads. The details of this strategy can be found in Chapter 3.

2. *Designing Energy and Bandwidth-aware Web Browsing*: Developing a mobile browsing technique, called Smart Mobile Web Browsing (SWB), that is a function of smartphone resources and the number of ads allowed to be displayed on webpages required an application-level modification. We modified the HTTP protocol so as to allow embedding a smartphone current resources (battery level and network type) in the HTTP header of the request made by the browser. On the server side, a certain policy is applied, resulting in an adapted webpage being sent back to the client. The adaptation of the web content comes in how many ads can be displayed on the webpage. The details of this strategy can be found in Chapter. 4.

This work makes the following contributions:

- quantifies the ad traffic generated during mobile web browsing (bytes/mo).
- estimates the additional cost of downloading ads to a user's monthly bill ($/mo).
- quantifies the energy consumed to download and display ads on webpages while mobile web browsing.
- investigates the impact of using different networks (3G and Wi-Fi) on the energy consumed to download and display ads on webpages.
- proposes a smart mobile browsing technique that aims to extend smartphone battery life and save the wireless bandwidth by controlling the amount of ads to be displayed. This solution attempt to keep the ads and at the same time alleviate its impact on smartphone limited resources.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Related research and background are presented in Chapter 2; the main challenges of handheld development are discussed. The

nature of mobile web advertising is explored and relevant existing work is reported. Chapter 3 describes the testing methodology used and the experiments carried out to achieve the first two contributions of this thesis. It also presents and discusses the measurements of energy and bandwidth used for ads. Chapter 4 contains the proposed smart mobile web browsing framework and the validation results. Finally, Chapter 5 documents some conclusions of this thesis and recommendations for future work.

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter introduces the challenges being faced in the development of modern handheld devices, smartphones and tablets, is given. An elaboration on the limited resources available on them and their implications on the user experience. The thesis framework requires a good understanding of how online advertising works and how such advertisements are embedded in both web pages and applications. Therefore, an overview of web advertising systems with a taxonomy of mobile advertising in handhelds is provided.

The ultimate goal of this thesis is to highlight the cost of web advertising in terms of bandwidth and energy. Several related studies are discussed in the literature review section.

2.2 Challenges in Handheld Development

Services offered by smartphones range from communication, entertainment, social network, to other Internet-based activities. As stated in the introduction, the growth of smartphone-use is ever increasing, primarily because today's smartphones have: powerful processors and graphics processing units, significant amounts of flash memory, high-resolution screens with multi-touch capability, and diverse applications, all thanks to the huge recent developments in both software and hardware. In spite of the fact that those diverse apps and services are available to everyone effortlessly, having such services and apps is faced by many

problems and challenges in development and design. The following subsections discuss some important issues in smartphone development.

Services offered by smartphones range from voice and Internet usage to other entertainment options. As mentioned in the introduction, the growth of smartphone popularity has sky-rocketed. A report published by ComScore in September 2012 stated that over 115 million people in the United States owned a smartphones out of 234 million total subscribers [16]. This fact shows a penetration of over 50%. The report also remarks that Google Android is currently in the top ranked with 52.2%, and Apple comes next with 34.3%. The trend worldwide is almost the same, according to *mobiThinking* [38]. The International Telecommunication Union *ITU* [4], states that there were 6 billion mobile subscriptions at the end of 2011. This figure was to reach 6.5 billion by the end of 2012, and by the end of 2016, it is expected to reach 8 billion.

The primary reasons behind the increased popularity of smartphones are the powerful processors and graphics processing units, significant amounts of flash memory, high-resolution screens with multi-touch capability and diverse applications, all driven by the rapid evolution in both software and hardware worlds. It is implied that such advanced services and technologies could not have been accomplished without advanced research. Overcoming many challenges has led to the success of smartphones and the great functionalities offered. This section mentions some of the most important problems that have been highlighted by researchers.

- *Battery life*

  In smartphones, Lithium-ion batteries are used due to their high energy densities. However, there is no tangible evidence that in the near future more energy can be packed into the small space available on today's modern hand-held devices [45]. In addition, more powerful processors, high resolution displays, huge number of applications, and advanced wireless technologies (LTE, WiMAX) are employed in today's smartphones and tablets. Such technologies and components drain energy at a very high rates and are considered to be battery hungry. For instance, the authors in [65] found that smartphone's components, namely, CPU, Wi-Fi, LCD, GPS, and 3G links, consume up to, 28.6, 24.6, 85.9, 33.6, and 36.5 Joules, respectively. Moreover, the energy cost of sending a simple SMS message and placing a call through 3G is more energy expensive than using GSM, according to [45].

  Much research efforts have focused on the battery life and the energy consumption in smartphones. Those efforts range from: testing [7],[41] and measuring certain activities, estimating and modelling the energy consumption of smartphone components (CPU, radio interfaces, GPS, and camera, etc.), measuring the energy cost of some

popular apps, to studying the impact of ads on the battery life. Solutions that could alleviate the battery life problem were proposed, as shown in section 2.4.1. Such solutions have targeted the hardware design as well as software. Since software solutions are easier to implement and flexible to customize, they are seen as more desirable and applicable to developers. Some software solutions include: using the cloud to minimize the energy cost for apps that use the Internet, developing optimization tools, and designing apps with energy efficiency in mind, as shown in section 2.4.1.

- *Bandwidth*

  A very recent addition to Internet traffic is the data traffic generated by smartphones and tablets. Along with increased number of smartphone subscribers, smartphones Internet traffic is increasing exponentially. Nokia Siemens published a report stating that cellular traffic is forecast to grow 10 times faster than fixed Internet traffic [64]. Ericsson also confirms that fact and shows that most of the cellular traffic is generated from smartphones, and network traffic is actually surpassing voice traffic [3]. However, due to the growing traffic generated by smartphones and tablets, wireless networks are edging to near capacity. As a result, serious pricing and performance issues may come to picture [9]. Thus, more research is needed to know better the characteristics of the traffic generated from hand-held devices [14][59].

  In fact, the bandwidth issue in the wireless environment can be broken down into two main categories: (i) from a network operator's perspective; with the smart devices available today and the vast number of apps that essentially function via Internet connection, and the fact that radio resources are scarce, many performance issues have emerged. Such an issue is considered to be expensive to tackle. For instance, Vasona Networks, in 2013, has put up a budget of $12 million to help operators optimize their network and solve the problems of mobile data congestion [34]. (ii) From an end user's perspective, given the facts mentioned in the first point, mobile operators tend to offer metered pricing plans that are more affordable to customers than the unlimited ones. Consequently, another issue arises and impacts user's experience: bandwidth hungry apps. According to [30] and [66], such apps generate extra traffic that is not needed for the actual operation of these apps. The extra traffic "overhead" refers to the advertisements and analytics traffic that was found to be a significant contributor in monthly data usage and in a user's monthly bills.

- *Security*

  The development of both devices and services have been market-driven. Most of them focus on new features but neglect security [19]. A recent study conducted by security software provider McAfee found that the amount of malicious software,

also known as malware, targeting Android, had jumped by 76% since the previous quarter. At the same time, Android had surpassed Symbian as the most-attacked mobile platform. Security issues due to ads are beyond the scope of this thesis.

Smartphone platforms, in some cases, need to give access to various resources, such as GPS, Network interfaces, and system tools, thus preventing the phone from sleeping. Applications that require many privileged-to-smartphone resources can create a high chance for security and privacy hazards. Such apps can perform some unauthorised actions by users or send some private information to remote servers without the awareness of users.

In a summary, the three above mentioned issues are very serious challenges to smartphone development. They require more research efforts to: (i) find out the main contributors of such issues (ii) come up with techniques and solutions that alleviate their severity. It is important to mention that the involvement of third-party advertisements, as a monetization technique, in apps and in other services offered on smartphones increases the seriousness of these issues. A lot of studies have found that ads contribute significantly to the severity of such issues as will be shown in detail in sections 2.4.1 and 2.4.2.

2.3 Mobile advertising

The field of advertising has evolved rapidly with today's advanced technologies. The variety of telecommunication network technologies and the increasing penetration rate of end-user devices createan ideal environment to reach billions of people effortlessly. Newspapers, magazines, TVs, radios, and street posters were the only means of advertising about 10 years back. With the advances in technology today, consumers are not only receiving ads wherever they are, but rather they can interact with it immediately. Moreover, ads are no longer distributed in a random way, they are selected and delivered in a smart way, using a user's location for example, so that consumers are exposed only to ads that are relevant to their interests. Therefore, mobile advertising with such potentials is one of the largest sources of revenue. According to [27], the mobile advertising market is forecast to grow from $3.4 billion in 2010 to $22 billion in 2016, with a 37% annual growth rate.

To further clarify the concept of mobile advertising, the Mobile Marketing Association in [8] defines it as "any paid form of marketing, advertising or sales promotion activity aimed at consumers and conducted over a mobile channel." The definition includes all forms of mobile advertising, in other words, any advertising mechanism that delivers ads to consumers over a mobile channel [22]. To narrow the field of mobile advertising, this

thesis focuses on advertising using online resources on smartphones and tablets, which is one of the most effective ways of advertising. Guha *et al.* in [24] consider it as a key economic force in the Internet economy. Mobile online advertising can be defined as any form of delivering ad content over the Internet to a mobile hand-set, e.g. smartphones, tablets.

It is important to mention that we are not particularly interested in the philosophy of the mobile advertising market. Rather, we want to give a general background of how those systems work and what entities contribute to the make-up of the system from a technical point of view. This chapter provides a brief background on the architecture of mobile advertising; however [23][33][62][53][13] provide more details on the architecture of mobile advertising eco-systems and the related measures of effectiveness.

2.3.1 Mobile online advertising ecosystem

The mobile advertising ecosystem is composed of four main entities; Users, Publishers, Ad networks, and Advertisers [53][13][17][31]. These actors are directly related to today's modern technologies. Therefore, the ecosystem of mobile advertising has grown exponentially, as illustrated next. This section begins with explaining each entity and the role that each one takes, and ends with answering the question of how the system actually works.

- *Advertisers* are companies that want to persuade people (users) to buy their products. They advertise their products by placing their ads on applications or websites through an advertising network or broker. Advertisers pay both publishers and ad networks based on specific pricing models [52], [13]. Gucci, and Toyota are example of advertisers.

- *Ad networks*, sometimes called ad brokers, are organizations who link advertisers with publishers as shown in Fig. 2.1. Their main function is to select ads that are then put on publisher's application or web-page. Based on either certain data gathered about the end user or the content that a publisher provides, appropriate ads are selected and delivered to the end user. Examples of advertising networks are AdMob, AdSense, and iAds.

- *Publishers* can be either the owner of a web-page or an application. In the case of web-pages, the owners of such websites allow ads to be displayed on their web-pages in order to make profit and keep providing end users with free content. Application's developers who offer free versions of their apps allow displaying ads on them so that they can keep those apps available to end users free of charge. Both cases are

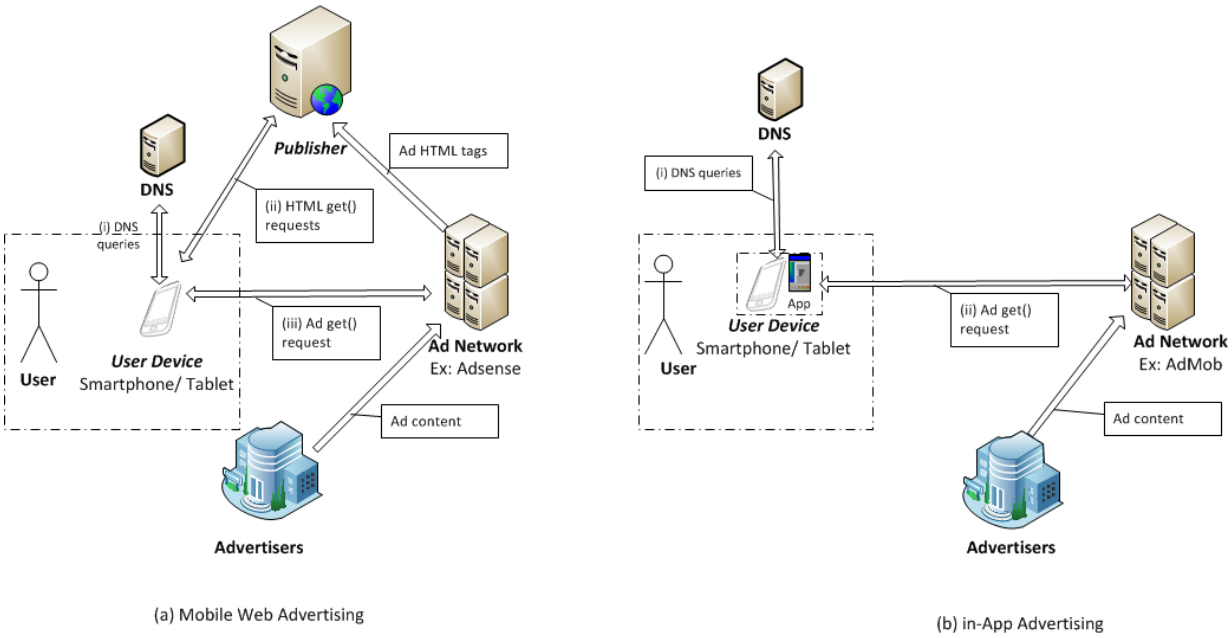


Figure 2.1: Mobile online advertising ecosystem.

basically interested in monetizing their contents and increasing user loyalty. For large publishers, a common practice is that they themselves take the role of ad networks. They usually have their own departments to take care of the work done by ad networks, *i.e.*, making the negotiation with the advertisers regarding the allowed ad contents and the pricing model they offer as well as performing the ad matching process based on the end user interest (ad selection).

- *Users* the last players in this system are the owners of the smart devices. They are interested in either using certain free apps or the contents of certain websites. Users interact with ads that are displayed on their screens. Consequently, they complete the circle that keeps the mobile ad ecosystem going by clicking on the ads they have an interest in.

The mobile advertising systems works as follows: all the actors above contribute to the generation of the revenues that keep the system going and all the players satisfied. As illustrated in Fig. 2.1, there are two scenarios to implement mobile online advertising:

- *Mobile Web Advertising*

In this scenario, the needed requests and messages to complete the task of downloading and displaying ads are depicted in the sequence diagram in Fig. 2.2. It starts with an end user fetching a certain website (publisher, *e.g.* www.mydictionary.com); the publisher's server then sends an HTML file that contains the pages contents plus other elements (Cascade Style Sheets CSS, Images, and JavaScribt codes/files.) The browser at the end user side starts rendering the HTML file. It will find tags for the other elements that require fetching. It sends a GET request to retrieve these elements. Ads come to the picture in the form of HTML or JavaScribt code, once such code is triggered while rendering the HTML file, another GET response is generated and sent to a specific ad server (Ad network, *e.g.* AdSense) to download ad content. The ad sever figures out what sort of ads should be sent to the users based on a matching technique that is done with the publisher's webpage content (Contextual targeting.) Once a user finds an interesting ad and clicks on it, he/she will be redirected to the advertiser's web-page. Behind the scene; first, advertisers make special sort of agreements with the ad network that specifies the pricing method and provide the ad network with the ads of products they want to promote. Publishers as well make other agreements with the ad networks that specify the financial conditions and the some regulations on what kind of ads are allowed to be published on their pages. The ad network, in turn, provides the publisher (website) with a piece of JavaScribt code or HTML tags that the publisher embeds in its web-page.

- *Mobile in-Application Advertising*

  In this scenario, things are different. Developers of mobile applications embed some ad libraries, as illustrated in Fig. 2.3, that are supplied to them by ad networks, within their applications' code. These ad libraries are pieces of code that send requests to ad networks in order to download ads and display them on the apps while they are run by users. The same role is played by the ad network (*e.g.,* AdMob,) which connects both advertisers and publisher (apps developers) together. In this scenario, though, ad libraries are set to send certain information regarding the users so that ad networks deliver ads that best match the user interest, which is called targeted advertising. Moreover, ad requests are periodically sent by the embedded ad libraries to ad networks, in order to show a variety of products and maximize the chances that a user clicks on them. As a result, developers maximize their profit by having more clicks on the ads they display on their apps. In some particular cases, namely games, another form of in-app ads can be found, where some brands are displayed in the background graphics of the game itself.

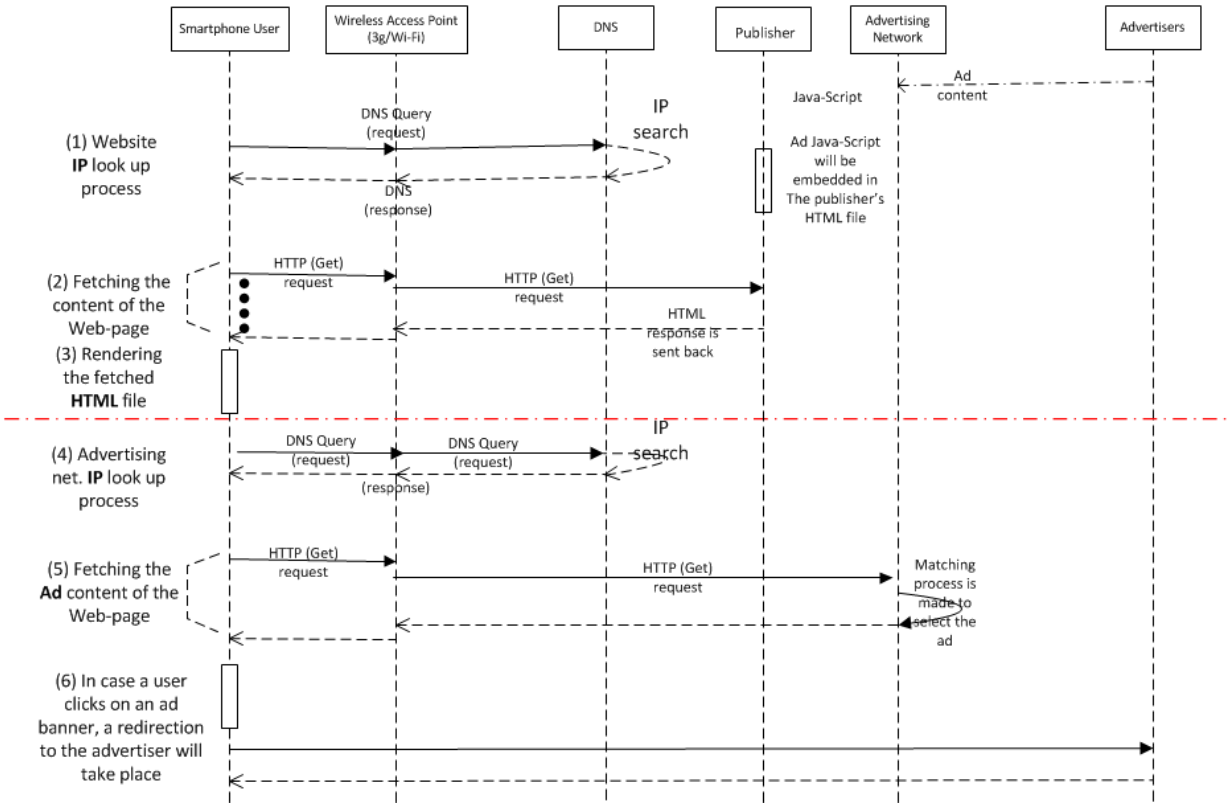


Figure 2.2: Sequence diagram of Mobile web browsing (DNS: Domain Name System.)

```
<manifest ... ...
    package="com.rovio.angrybirdsrio" >
    <application                                                    Admob Publisher IDs/Settings
        <activity android:name="com.rovio.ka3d.App">
            <intent-filter>  <action android:name="android.intent.action.MAIN"> </action>
                <category android:name="android.intent.category.LAUNCHER">    </category>
            </intent-filter>
        </activity>
        ... ...
        <meta-data android:name="ADMOB_PUBLISHER_ID" android:value="a14d6f9cc06f96b">
        <meta-data android:name="ADMOB_INTERSTITIAL_PUBLISHER_ID" android:value="a14d6fa2b901034">
        <meta-data android:name="ADMOB_ALLOW_LOCATION_FOR_ADS" android:value="true"> </meta-data>
        <activity android:name="com.admob.android.ads.AdMobActivity"> </activity>
        <receiver android:name="com.admob.android.ads.analytics.InstallReceiver" >
            <intent-filter >
                <action android:name="com.android.vending.INSTALL_REFERRER"></action>
            </intent-filter>
        </receiver>
        ... ...
    </application>
    <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"> </uses-permission>
</manifest>                                                         Admob Components
```

Figure 2.3: Example of in-App Ad library as it appears in the source code file [23]

2.3.2 Advertisement format

Different ad formats can support different ad types. The most common formats used in mobile advertising can be categorized as follows:

- *Text ads*

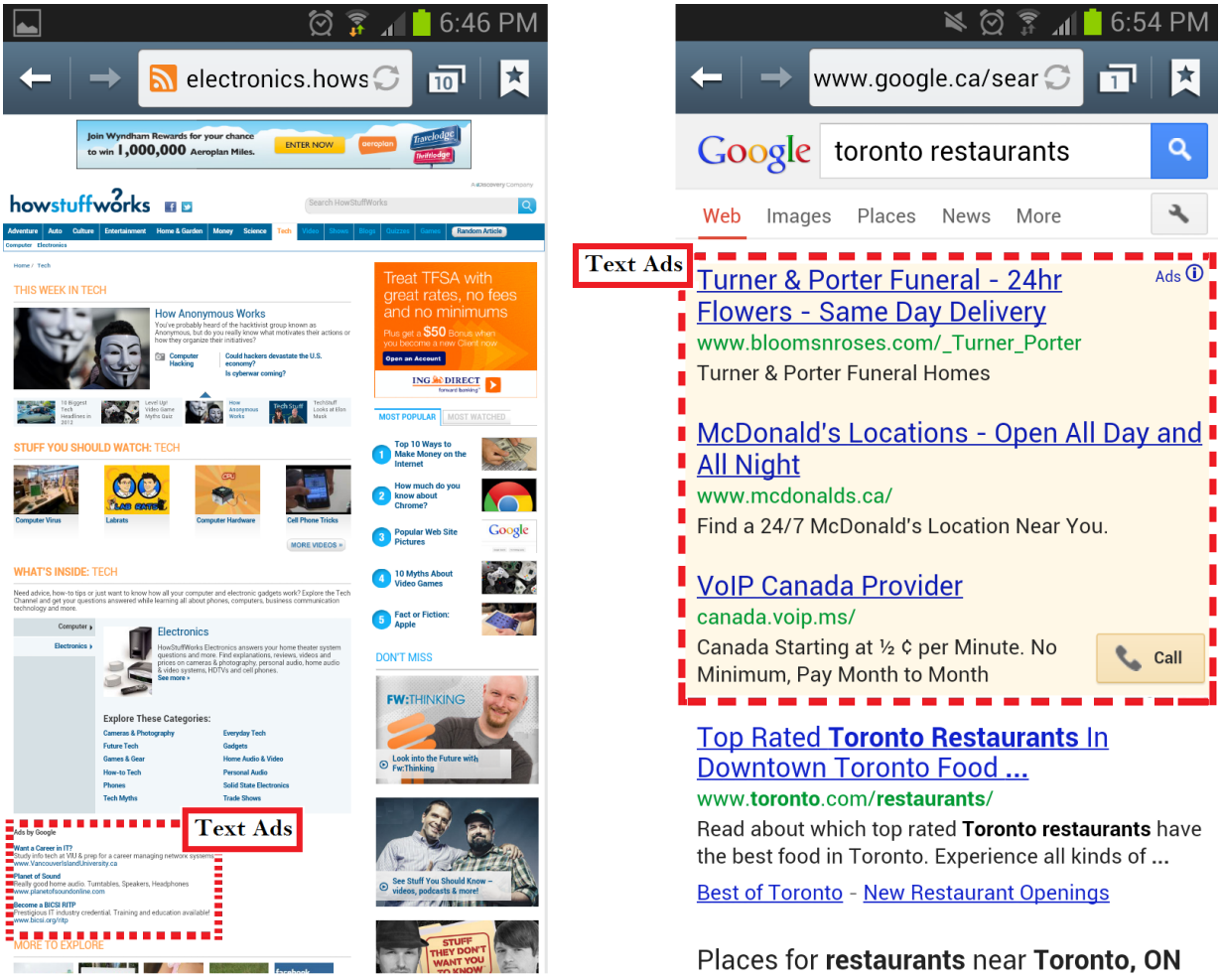


Figure 2.4: Examples of text ad formats

Text ads on mobile devices contain mainly three lines of text: (i) a title; (ii) a short description; and (iii) a URL that links to the advertiser's website. Fig. 2.4 shows an

example of such ad formats. This type of ad is very common in search advertising and considered to be the most dominant one as well as the less annoying in terms of user experience. For example, when a user searches for certain information using a search engine, the first few results in the top of the list are usually ads. The ads in such advertising are usually selected based on: (i) the keywords that the user used, (*Keyword-targeted advertising*), or (ii) based on the content of a webpage, (*Contextual targeting*).

- *Display ads*

  Display ads are one of the oldest forms of advertising. They are basically images designed with high graphics content and with a link to the advertiser's website, and occupy a fixed area of a webpage or an application. Fig. 2.5 shows an example of this kind of ads. The most familiar type of display ads are banners which are normally located on the top or/and the bottom of web-pages or apps. In some cases these ads include rich media within the banner ad itself. Those ads come in a form that consist of multimedia elements such as sound and animation. In addition, those ads require browsers capable of executing JavaScript codes.

2.3.3 Ad selection mechanisms

There are several methods used to best select ads to be displayed on the user screen. Those methods are based on either: (i) the information gathered about the end user Behavioural Targeting, or (ii) the web-pages' content that a user is surfing (Contextual Targeting).

- *Behavioural Targeting*: In this method, ad networks make use of the information gathered about the end user. Then they apply complex algorithms to accurately match the user's interests and needs with appropriate ads. Such information can be found in the form of the history of a user's browsing, the user's current location, and the apps that a user has on his/her mobile device. Based on this information, ad networks profile users and hence are able to provide them with the most relevant ads. For example, ad networks would send ads that are about sports tools to a user who is known to have searched or visited specific sports tools section of a general sport-store website).

- *Contextual Targeting*: This method involves no user information in the process. Instead, the ad network selects ads based on scanning the keywords of the page that a user is surfing. Such a process is very complex and requires tools that uses data mining algorithms to map the web-page content with the relevant ad content. For

Figure 2.5: Examples of Display ads format.

example, if a user is surfing a website that provides information about learning the English language, the ad network would send ads for some English Language Schools or electronic dictionaries.

2.3.4 Ad pricing models

As mentioned in subsection 2.3.1, an ad network makes a number of agreements with advertisers and publishers to get the system functioning appropriately; one of them is the pricing agreement. Based on the following pricing models, ad networks and publishers get their revenue.

- *CPC (Cost Per Clicks)*: In this model, the cost of advertising is based on the number of clicks received. Advertisers pay the publisher and the ad network that delivered the ad to the end user when the user clicks on the ad. However, a user does not necessarily buy something. From the advertisers side, once a set amount of clicks has been reached, the advertising campaign is made inactive.

- *CPM (Cost Per Mille)*: In this model, the cost per mille is media term describing the cost of 1,000 impressions- views. The advertiser pays for the exposure of their ads. In such a case, advertisers buy a package with a certain number of views; this can be any positive number. Once that fixed number is reached, the campaign remains in a finished status, *e.g.,* a $70 CPM means the advertiser pays $70 per every 1,000 times his/her ad appears on any website.

- *CPA (Cost Per Action)*: Advertising impressions are free in this model, and advertisers pay only for the number of users who complete some action that advertisers feel will lead to a sale. Publishers run the ad at their own risk and do not get paid until users click the ad and make a purchase, sign up for a service, or register for a newsletter.

2.3.5 Advertisements in Smart Devices

A fundamental question that can be asked is: Why are smart devices a suitable environment for advertising? Basically, the features that mobile environment offer make them a suitable advertising platform. In addition, the almost 6.8 billion mobile-cellular subscriptions worldwide [6] is an indication that there is a huge opportunity that advertisers can reach their audience almost effortlessly. Compared to web-based advertising, apps and other mobile

services create an attractive and unique channel to the advertisers to present their ads to the potential consumers. Moreover, advertisers have the capability to instantly target the consumers individually by leveraging the Global Positioning System (GPS) location and the search history of web-browsers and map apps.

2.4 Literature Review of Existing Research

2.4.1 Energy Cost of Advertising in hand-held devices

The energy consumption and the battery life in smartphones have been the topic of numerous publications [42][39][47][59]. Energy management solutions in a smartphone environment were discussed in details in [58]. However, fewer publications have reported on the issue of energy consumed by the advertisements embedded in both apps and web pages. In this section, the current existing work is categorized into applications and web pages, as described below.

2.4.1.1 Applications

Prior studies have deeply analyzed the usage of mobile apps from an energy perspective, since battery life is a very important aspect of smartphones. In particular, they focused on the free apps, because free apps rely extensively on ads as a source of revenue. Such ads are actually battery hungry and adversely affect battery life. The energy studies on smartphone apps were either done by measuring the energy using hardware tools [47][59] or estimating it using software tools [42][39], such tools are based on modeling energy-consumption behavior. Table 2.1 shows a summary of the work done on the impact of in-apps advertising on the battery life of smartphones.

In [42] for instance, a huge effort has been made to profile the energy consumed by hardware components of smartphones (GPS, Camera, CPU, Wi-Fi, 3G, etc.) and link the amount of energy drain to the originating apps. The authors implemented a software tool called *eprof* that is used to estimate the energy consumed by apps. The tool is a fine-grained energy profiler that can model the power consumption online on Android and Window smartphones. The modelling of the consumed power is done based on Finite State Machine, FSM [43]. Such a tool is able to track program calls as well as track the power

Table 2.1: Summary of the work done on in-apps advertising energy cost

Ref. #	Platform	Kind of Apps	Network Access Mode	Evaluation Method/ Measurement tools	Findings	Proposed Solution	Improvements/ Observations
[42]	Android, Windows	Games, news, Social networking apps, Web Browser	Wi-Fi, 3G	*Eprof*: an energy accounting tool; Energy Modelling Software based on FSM.	65%-75% of the energy consumed by apps is due to ads.	Certain bundles strategies	20% to 65% reduction of the total energy consumption of 4 selected apps
[47]	Android	Free games apps	Wi-Fi	Experimental work using Monsoon Power monitor tool [56]	in 5 popular games, ads consume up to 4%-15% of apps total energy	N/A	The power consumed by ads depends on how frequently ads are fetched
[59]	Android	A purpose-built app	Wi-Fi, 3G	Experimental work using Monsoon Power monitor tool [56]	Parameters such as; refresh interval and retrieval mechanisms can significantly impact the power cost of ads. However, ads power overhead due to 3G connectivity is more than Wi-Fi.	*Adcache*, a caching mechanism that reduces the impact of ad traffic on battery life	The power consumption due ads is reduced by 50% using *Adcache*
[39]	Widows	Games, news, weather, productivity, social networking apps	3G	*WattsOn* [37], an energy estimating tool for Windows mobile phones.	Ads contribute by ∼65% of the total app's communication energy	Ad prefetching with app usage prediction	a reduction of more than 50% in energy consumption due to serving ads
[39]	Windows (on a PC)	Browsers	Wired connection	Experimental work	The measurements show that the additional energy consumption due web advertising is 2.5 W.	N/A	Based on their results and some national statistics, they found that the total energy used by ads equalize the annual electricity usage of 1891 Dutch households.

consumed by hardware components of the smartphone, and finally map the power consumption to the apps that initiated such calls. This solution is meant to help developers design energy-aware apps. The findings of Pathak *et al.* [42] revealed a number of interesting points, most importantly the amount of energy that is spent by third-party advertising libraries.The authors claim that 65% - 75% of the energy consumed by free apps is spent on fetching advertisements and uploading users' data to remote servers. Such data is used by advertising networks to serve up ads that are most relevant to users.

Another study that addresses the cost of ads in smartphones was published by Prockova *et al.* [47]. The authors investigated the influence of ads on mobile games apps running on Android platform. They found that the existence of ads in such apps increases the energy consumed by the device while using a Wi-Fi confection. For instance, in five mobile games, 4-15% of the total energy consumption was due to requesting ads. The reason behind this increase is that each time an app publishes an ad, it first fetches it and then displays it. In other words, the radio interface has to be turned on to transfer the ad content from the ad server to the smartphone each time a new request is made by the embedded ad library. Hence, more energy would be consumed during such tasks. The process takes place periodically, depending on how frequently a new ad is fetched. The experimental work in [47] focuses on measuring power consumption due to network usage, *i.e.,* the cost of fetching advertisements. The authors relied on comparing free apps that include ads and the paid ones that are ad free. The difference in the measured consumed energy between the two types is the pure energy cost of fetching and displaying ads on free games on smartphones.

Two months later, the energy consumption of ad networks was addressed in [59]. Vallina-Rodrigues *et al.* started by characterizing mobile ad traffic and then discussing the benefits of some techniques that are used to mitigate the energy and network signalling overhead caused by the current ad delivery systems. They proposed a system called *Adcache* that minimizes the cost of ad delivery mechanism by friendly caching ad content and serving it locally when needed. *Adcache* allows the retrieval of ad contents under certain optimal network conditions. The advantage is to cache the ad content whenever the the optimal network condition is satisfied and displayed it on used apps later. Such a technique enables publishing of ads with less energy wastage as well as less network signalling overhead. Their results shows saving of up to 50% of the energy consumed by the process of delivering ads using *Adcache*. The evaluation was done on a purpose-built app where measurements were taken over both radio interfaces, namely, Wi-Fi and 3G, with different ad retrieval mechanisms and different refresh intervals. They, moreover, compared the

energy consumption of requesting ads from most popular ad networks: AdMob, Millennial Media, and InMobi.

Mohan *et al.* in [39] have addressed the in-app mobile advertising from two important aspects. First, they have studied how much ads in Widows phone apps contribute to the total energy consumption. They found that ads, on average, account for 65% of the total app's communication energy. Then they proposed a solution based on ads prefetching and apps usage behaviour predictability. The evaluation of the proposed approach shows a reduction of 50% in energy consumption due to serving ads.

2.4.1.2 Webpages

In the context of studying the energy cost of web ads while mobile browsing, there are not much studies that talked specifically about this issue [55]. However, there are several research articles that investigated mobile browsing from energy perspective [67, 57, 42, 68] as well as other performance aspects, namely, Speed, simplicity, and usage and usability [63, 26, 50].

Studying the energy cost of web advertising on smartphones or hand-helds in general has not been performed so far, to the best of our knowledge. However, there is a study done by Simons and Pras [55] at targeted the hidden energy cost of web advertising on desktops (PCs). They investigated the amount of CPU and display energy consumption due to web advertising and tracking. They found that displaying ads that are rich in animations and graphics is an energy-expensive task. It puts the CPU under computational stress that causes higher power consumption and leads to harming the environment as well as costing more money. For instance, their results reveal that the cost to render and display web ads is 2.5 W. Using some national statistics, and based on their results, they found that the total energy used by ads equalize the annual electricity usage of 1891 Dutch households.

The browser's choice impact on battery life of laptops was investigated by Walton in [61]. He made an effort to answer the question of: Which web browsers is the most energy efficient on laptops? Webpages energy performance modelling was the issue in several studies. In [42], Pathak *et al.* profiled the Android browser for 30 seconds over a 3G link using the proposed *eprof*. The scenario was: a user opens the browser, does a Google search and then closes the browser. they found where exactly the energy spent in the browser. HTTP, GUI, user tracking and TCP *conditioning* consume 38%, 5%, 16% and 25% of the total energy consumed by the browser, respectively. In [67], a similar tool called PowerTutor was

described. PowerTutor is a tool, available on the Android Application Market, that estimates and profiles the power drained from apps and informs developers about the resultant energy implications of their software designs on the battery life. Understanding webpage characteristics from energy perspective was considered in [68][69]. The authors exploited such understanding and applied it to achieve more energy-efficient mobile web browsing. Their methodology focuses on effectively making full use of processor frequency scaling to balance its performance with the resultant energy consumption of loading webpages.

In [57], Thiagarajan *et al.* analysed the energy consumed by mobile browsers. They measured the total energy cost of rendering web page elements as well as the 3G radio energy needed to fetch such a web page. Then, they went further by breaking down such measurements and answering the question of how much energy each and every web component consumes: cascade style sheets CSS, Javascript, images, and plug-ins. Based on their measurements and analysis, a number of recommendations were provided to develop and build more energy efficient site. However, the ads were not in the focus in [57]. Some other studies looked at web browser and web pages from other perspectives. For instance, in [63], Wang *et al.* critically analysed the slowness of web browsers on smartphones. They found that the main reason the slow web browsing is the process of fetching different web contents. They concluded that the key point of improving mobile browsers is to speed up such process. In [26], Hoehl *et al.* investigated the possibility of benefiting from presenting mobile version websites on desktop computers so that simpler web contents can be provided to people with cognitive disabilities.

2.4.2 Bandwidth Cost of Advertising in hand-held devices

Due to the scarcity of resources on mobile devices, the topic of characterizing the traffic for mobile devices has been addressed in several studies in the recent years. Such studies range from understanding the nature of data traffic to analysing the cost of particular contents (*i.e.* ads.)

Table 2.2: Summary of the work done on in-apps ad bandwidth cost

Ref. #	Platform	Kind of Apps	Network Access Mode	Evaluation Method/ Measurement tools	Findings	Proposed Solution	Improvements/ Observations
[66]	Android, iOS	Games, Entertainment, Education, Life style, free app tools.	Wi-Fi	Experimental work using *tcpdump* [28]. *tcpdump* a network monitoring tool that captures all the 802.11 packets	The cost of 4 free apps per month range from $2 (∼40 MBytes) to $13 (∼210 MBytes), depending on the platform, data plan price, and usage time. (Note: The cost analysis is based on empirical calculations)	N/A	Given a limited monthly data plan, the cost of some of the free apps is 48%-1299% higher than the same paid apps
[30, 29]	Android	Free apps	Wi-Fi, 3G	Experimental work using *Shark for Root* App [1]	Some free apps that are played for 30 minutes a day, can generate up to 40 MBytes a month	*CAMEO*, a middleware frame work that can mitigate the cost of ad delivery. It's based on caching ads locally and predictively download them whenever un-metered or inexpensive wireless connectivity is available. And then sever them locally when a user runs an app	*CAMEO* can mitigate the cost (Energy and Bandwidth) of ad delivery mechanism significantly
[59]	Android, iOS	N/A	Cellular network	Off-line analysis of a data set comprises of 22 TBytes of traffic, that belongs to one day of European mobile operator with 3 million users.	Ad traffic contribute with 1% of the total 22 TBytes (∼220 GByte).	N/A	N/A
[51]	Any devices	Any apps	Wi-Fi, 3G	N/A	N/A	*MASTAds*, an ad delivery architecture that can be integrated to the existence ad ecosystems with minimal changes.	*MASTAds* provide means to deliver ads to mobile phone apps with minimum resource usage (Energy & Bandwidth) without compromising user's privacy
[60]	Windows (PCs)	Browser	Wired connection	Experimental work using Wireshark [15] and Proximodo [46]. The first tool was used to monitor the traffic, and the second to enable and disable the ads	Ads take up 7-9% of the total data traffic	N/A	The additional cost would be $4.5-7, $50-80 while browsing domestically and internationally, respectively

Since the generated mobile data traffic is a new addition to the traditional traffic, the first study that noticed the emergence of the added mobile traffic was by Maier *et al.* [35]. They have found in their data set that comprise more than 20,000 residential DSL lines, Digital Subscriber Lines, that the contribution of the traffic due to mobile hand-held devices to the overall traffic volume is still small. However it is growing very rapidly compared to the traditional traffic. The work in [21] is similar to the one done by Maier. However, Falaki *et al.* in [21] broke it down to the application level and analysed a small data set that consisted of 43 smartphone users to study the nature of smartphone traffic. They found that the largest contributor of the traffic is web browsing. Based on their analysis, moreover, they proposed several mechanisms that can improve the power consumption of smartphones as well as its overall performance. In [32, 14], attempts have been made to analyse mobile traffic. Such researches were conducted to provide network operators with a clearer picture of the nature of mobile traffic and the originating applications, so the network operators would optimize their quality of service accordingly. However, Chung *et al.* [14] were the first to show the contribution of the traffic due to advertisements. They found that ads contribute with 0.007% of the total mobile traffic that was gathered from a university with 4000 users. These studies have led to deeper studies that focus on the traffic generated only from mobile advertising as we will see next.

2.4.2.1 Applications

Several previous studies have focused on analysing the overhead traffic generated by network related apps. Specifically, they focused on the free apps that embed ad library to generate revenue to the developers. In [66], for instance, Zhang *et al.* answered the question of how expensive *Free Apps* are. In their study, they consider the traffic that is not necessary to the app's operation as overhead, specifically, ads and analytic data traffic. Their measurements were done on a Wi-Fi access point, and they found that some free apps generate up to 1 MB of overhead traffic while being in use for 10 minutes. Moreover, they estimate the cost of such traffic in terms of the monthly payment a user makes for cellular data plan. A comparison between a number of free and paid apps were made. Interestingly enough, their detailed measurements show that, given a limited monthly data plan, the cost of some of the free apps is 48%-1299% higher than the same paid apps.

In [30, 29], an attempt was made to mitigate the real cost of free mobile apps. Khan *et al.* [30, 29] first analysed the network overhead due to the delivery of mobile ads. They found that, for 30 minutes of active use of free apps, such overhead can contribute with a good fraction of the total data usage. They also found a free app namely Fruit Ninja Game,

played for 30 minutes a day, can generate 40 MB of ad traffic in a month. Such app usage can cost ∼56 cents a month which is significantly expensive compared to the paid version of the same app (without ads) that cost only 99 cents. Then, based on the observations gotten from the above analysis, they proposed a new advertisement delivery mechanism called *CAMEO* that can reduce the bandwidth cost of in-application advertising. Their method aims basically to: (i) predict the kinds of ads that should be displayed to the user based on utilizing his/her context; (ii) cache the ad content (prefetching) to a local cache when the user is connected to un-metered connections (such as Wi-Fi); and (iii) display it when the app is being used.

In another major study, Vallina-Rodriguez *et al.* [59] used data sets, from European mobile operator corresponding to one day of traffic to characterize mobile ad traffic. They developed a methodology that classifies the gathered traffic into three main components: ad network traffic, analytic traffic, and mediation services traffic. Their analysis shows that ad traffic contributes with 1% of all mobile traffic in the data set, with 31 kB per day for 50% of the devices. They found that ad traffic generated by 50% of Android devices accounts for more than 5% of the total daily traffic. The case for Apple devices is almost the same, the volume of ad traffic generated by 20% of iPad and iPhone devices account for more than 4.7% and 7.6% of their traffic, respectively.

In a very recent study, Seneviratne *et al.* [51] focus on in-app advertising. They studied the existing in-app mobile advertising eco-system, which enabled them to precisely address the issue of maximizing system efficiency, i.e. minimizing the resources used to deliver ads to smarphones without compromising users' privacy. Moreover, they proposed a new mobile advertising system, called *MASTAds* (Mobile Anonymous but Still Targeted Ads), that delivers targeted ads efficiently without having the traditional conflicts, which are saving energy and bandwidth and preserving users' privacy.

2.4.2.2 Webpages

There have been some studies in literature that quantified the traffic generated while web browsing. An attempt was done by Erman *et al.* [20] to classify the HTTP traffic in homes. Their study covered 17,000 broadband DSL subscribers' traffic for a month in Texas. They answered some questions that characterize the nature of HTTP traffic in homes. For instance, they found that not all the HTTP requests are made by PCs or laptops. Also, they found that 11% of all HTTP requests are being made to communicate to advertising servers, which is equivalent to 0.2% (that is ∼312 GB) of the total bytes downloaded. However, their study did not particularly target the amount of ad traffic generated only from mobile web browsing. Their results show the ad traffic generated by PCs, laptops,

mobile application, and mobile web browsing.

Brande *et al.* [60] considered ads as an unwanted traffic while mobile browsing. They investigated ads traffic contribution in the total web traffic. They built a testing system in a way that allows them to block the ads when needed, and automated the browsing process to mimic the browsing behavior. Their results reveal that web advertisement take up 7-9% of the total data traffic. Based on the percentage of the generated ad traffic and the average domestic and roaming charges, the authors estimated the cost of web advertisement for mobile users. They found that the additional cost would be $4.5-7 and $50-80 while browsing domestically and internationally, respectively.

Chapter 3

Measurements and Analysis

This chapter describes a number of experiments carried out to achieve the first two contributions of this thesis. Section 3.1 describes the performance metrics used throughout this thesis. Section 3.2 details our testing methodology and test benches; finally, the results and measurements of the energy and bandwidth are presented and analysed in Section 3.3.

3.1 Performance Metrics

The measurements and analysis of smartphone resources are conducted based on specific chosen performance metrics. The main metrics used to evaluate the impact of web advertising during mobile web browsing are the *Bandwidth (B.W)* and *Energy Consumption*.

- *Bandwidth* is the bandwidth needed to download web-pages, and has been chosen to be an evaluation metric for the following reasons: (i) downloading web ads consume B.W, and in the wireless environment, the available data plans tend to be limited and costly, especially if the monthly cap is exceeded, and (ii) not all of the downloaded information is of interest to end users. Accordingly, this "scarce" resource is evaluated to study the impact of web advertising during mobile web browsing. The term B.W cost in this thesis means how many bytes are transferred over the network to download a web-page.
- *Energy Consumption*: as mentioned in section 2.2, the battery life is known to be the bottleneck in the advancement of smartphones; therefore, and since ads are considered overhead data that consumes extra energy, the energy consumed while web

browsing is chosen as a performance metric. The term "energy cost" in this thesis means how is much energy drained from the battery when a web-page is accessed.

3.2 Testing Methodology

To perform the testing, the first challenge was how to separate the web content. A method was sought to block or insulate the ads from the web-pages. The available blocking techniques are as follows.

3.2.1 The Available Ad-blocking Techniques

- *Ad-blocking Applications*: A number of apps available online can block the ads, namely Ad-Vanish Lite[1], NoRoot Ad-Remover Lite[2], and AdFree[3]. These apps work by turning off the Internet connection completely. They are designed to block ads from *Android* games and works is as follows: a user is requested to prepare an Ad-block list, so if he/she wants to stop displaying ads on a game, all he/she needs to do is put this game on the Ad-block list. These apps in turn make sure that the games will have no access to the Internet any more, and hence the ads are blocked. These Ad-blocking applications work fine for the apps that do not require Internet connection. In our case, web browsing, establishing an Internet connection is essential. Therefore, this Ad blocking option does not satisfy our testing methodology.

- *Browser Plug-in Ad-blockers*: mobile web browser such as Mozilla Firefox and Android can use plug-in ad-blockers to block ads. Some plug-in apps are available online (Adblock Plus [4],) their functionality is based on filtering out the ad URLs. These plug-ins have a black list that contains advertising companies' URLs; thus, whenever an ad request made by the browser is found on the black-list, that request is aborted or cancelled. Our comments on these ad-blocking methods are as follows: (i) installing these plug-ins will change the way browsers work, and hence the measurements will be distorted. In other words, having such plug-ins working in the background of the browser requires extra activities and computational power. As a result, extra energy is consumed or wasted. For our case, where we are looking to measure the real energy and bandwidth required to download and display ads, using

[1]https://play.google.com/store/apps/details?id=com.atejapps.advanishlite
[2]https://play.google.com/store/apps/details?id=com.atejapps.litenorootadremover
[3]https://play.google.com/store/apps/details?id=com.moshedavidson.browser
[4]https://adblockplus.org/en/android-install

this option will give us distorted measurements that are not accurate. (ii) Not all of the ads will be blocked since the operation of these plug-ins relies entirely on the specified list, which may not contain all the advertising companies (Ad networks); see section 2.3.1 for further details. As a result, some ads will still be displayed even with using this option. (iii) The ad links (ad tags) that a web-page has are already delivered and parsed by the browser; as a result, some bandwidth and energy would have been consumed. For these reasons, we had eliminated this option as well.

- *Proxies*: Proxies, namely, Privoxy [46], can be installed at a middle point (*e.g., a Wi-Fi access point or a network router*) to block ads. The mechanism used to block the ads is similar to the one explained in *Ad-blocking Applications*. Therefore, this option has also been eliminated.

The last two options suffer from the same problems; moreover, they require human effort and time to keep their ad-blocking lists up-to-date. Therefore, another approach was need to achieve the goal of blocking ads without engaging the smartphone in this process.

3.2.2 Our Ad-blocking Strategy

The approach used to tackle the ad-blocking problem and take full control of the web content crystallizes in having our own web hosting server. By doing so, we can either enable the ads or remove them completely from the web-pages before they are even requested by the end user. The process is as follows; (1) choose certain web-sites, (2) download them and made two copies of every one, (3) modify and remove ad codes found in one of them and keep the other one as is, and (4) upload one at a time to perform the testing. We modify the the web-pages manually by removing all the ad-related content from the requested HTML document. Figure 3.1 illustrates the sequence of the ad-blocking process and the final view that is displayed on the end user screen after the ad-blocking technique is applied.

3.2.3 Energy Test Set-up

The test bench used to measure the energy consumption in smartphones is described in this section. Our experimental setup is shown in Fig. 3.2. We use a Monsoon power monitor [56] to power the smartphone and accurately measure the real-time consumed current. We partially disconnect the phone's battery to allow the communication work between the phone and the battery. We do so to prevent the phone from going into sleep mode or power saving modes. As shown in Fig. 3.3, the battery has four pins, namely, the positive

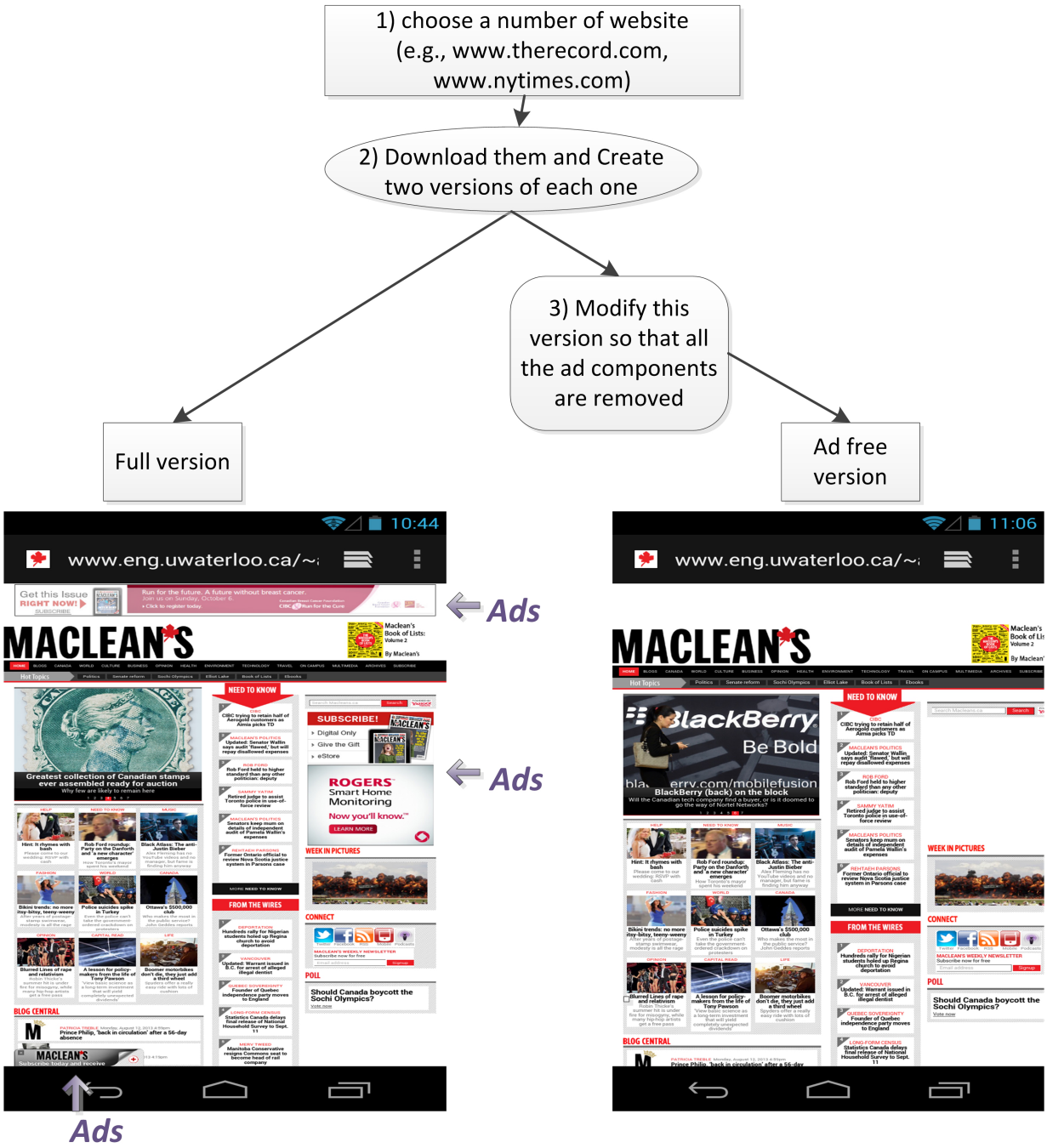


Figure 3.1: Ad blocking strategy

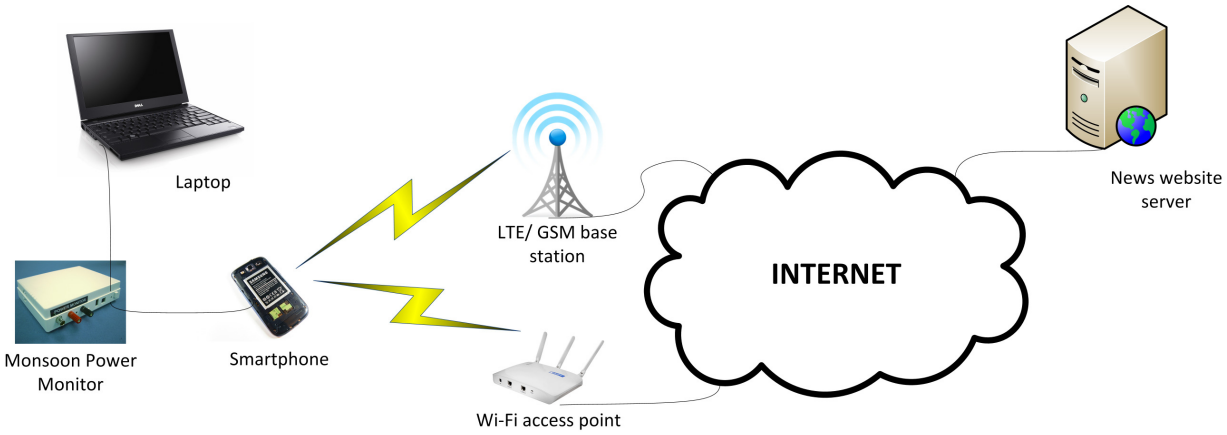


Figure 3.2: The energy measurement set-up.

(pin # 1) and ground (pin # 3) pins, communication (pin # 2) and temp line (pin # 4) pins. We insulate the battery's pin # 1 and connect the one in the phone directly to the power meter. We keep the other three pins connected; however, we also connect the power monitor ground to the phone and battery ground as illustrated in Fig. 3.3. By doing so, we can measure the actual current drained by the smartphone without any hidden cost.

The power monitor is connected to a desktop where the measurements are received and stored for off-line manipulation and analysis. The PC runs the Monsoon Power Tool software. This power monitor provides very accurate measurements; at a sampling rate of 5000 sample per second, and also has adjustable DC voltage output (from 2.1 to 4.5 V.) Moreover, Matlab is installed in the PC, and is used to process that measurement and generate the result graphics. We performed our experiments on a *Galaxy Nexus* smartphone running *Android V4.2.1.* The phone settings during the testing were kept constant to provide consistence measurements.

To conduct the experiments, we chose *Android Browser* that is the built-in web browser in *Galaxy Nexus* smartphone and another browsing application "semi web browser" that we developed. The energy cost experiments were done over a *3G* cellular link and a *Wi-Fi* private access point (AP) installed in our lab. The AP is *Cisco Linksys*, and it supports the *IEEE802.11g* interface. For the *3G* cellular connection, an *HSPA+ Bell* prepaid SIM card was used. This connection provides a typical speed up to 7 to 14 Mbit/s [2]. The experiment was done under good network conditions. For the phone configurations, we followed the methodology proposed by [7], in order to have consistence measurements.

To ensure that the measured energy is the real cost of requesting web-pages, we make

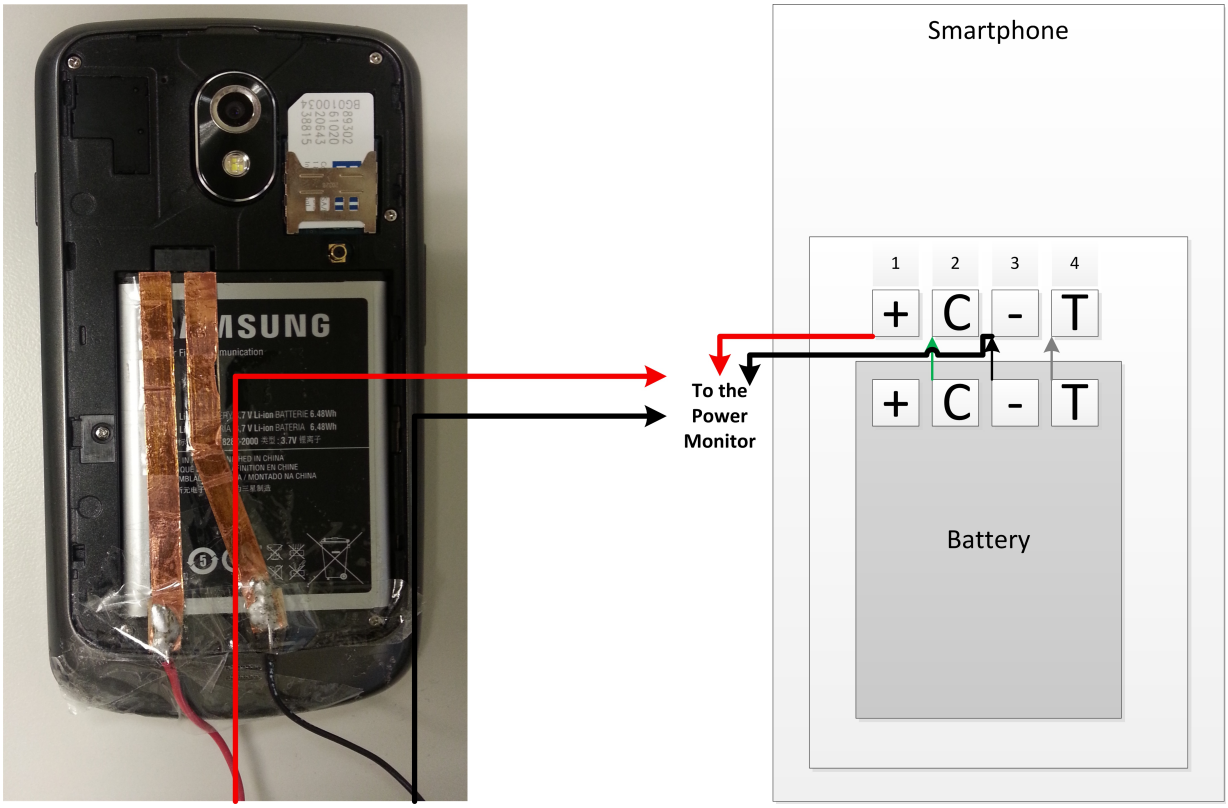


Figure 3.3: The battery bypass connection

sure no data is cached and all the web elements will be requested entirely fresh from the web server in each web request. Therefore, we clear the browser's cache before every experiment and also disable all the other network-related applications (*e.g.*, Android update libraries.) The web server used for testing, is an HTTP server implemented in Java, located in our lab. The server is deployed on a Dell desktop machine running *Windows 7*.

3.2.4 Bandwidth and Network Test Set-up

The test bench used to monitor the traffic going from and to the smartphone is shown in Fig. 3.4. We installed Wireshark on a Mac-book laptop to leverage the *Monitor Mode* feature that is available built-in on Mac-book laptops. This feature allows the packet sniffing task to performed passively; that is, we can capture all the traffic exchanged between the AP and the smartphone without involving any activities on the phone. Wireshark is a

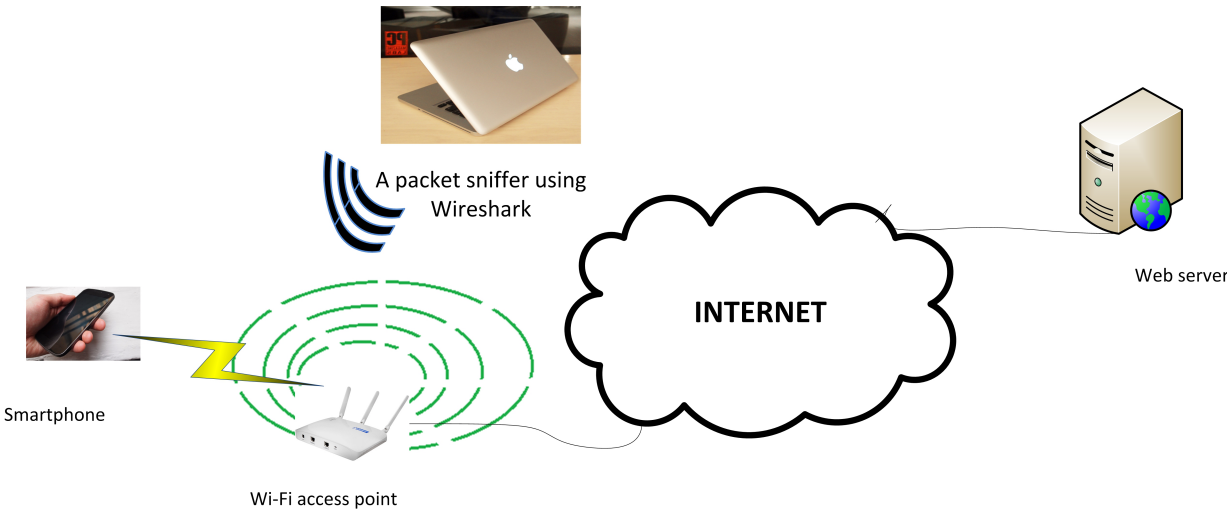


Figure 3.4: The bandwidth measurement set-up

powerful monitoring and analysis tool, and enables us to monitor packet information from the radio level up to the application level packet information. The collected packet traces are analysed to classify how much bandwidth web ads consume and how much bandwidth the non-ad related web contents consume.

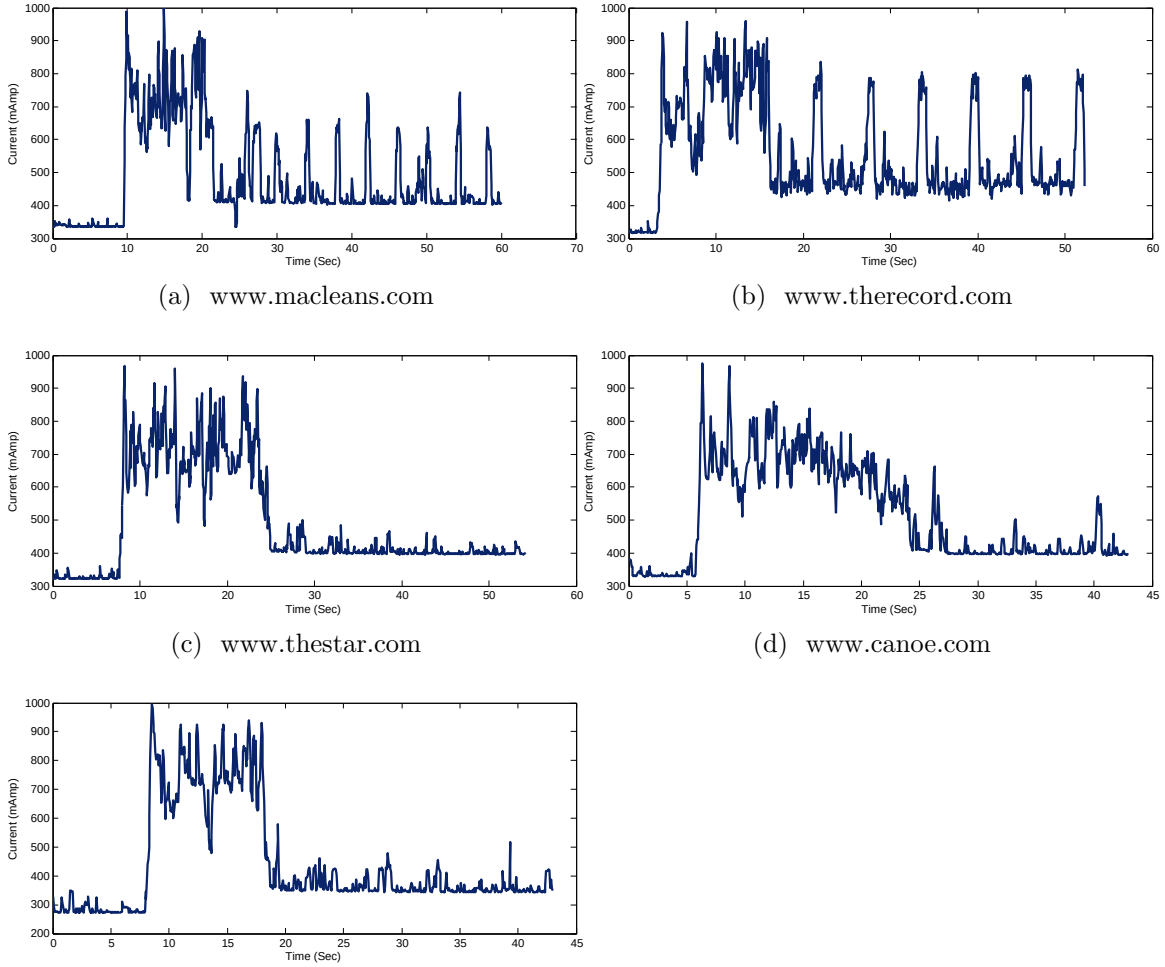
3.3 Energy and Bandwidth Measurements of Web Advertisements in Smartphones

3.3.1 Energy Measurements Test Cases

Using the described testing infrastructure for measuring the energy consumption we were able to measure the instantaneous drained current when a user is accessing the web. We started our energy experiments by testing how much energy is consumed when browsing a number of websites. The terms "drained current" and "consumed energy" are used interchangeably throughout this thesis due to the direct relationship that connects both of them, as shown by the equation $E = I * V * t$, where E is the energy, I is the current, V is the voltage, and t is the duration time.

We chose a number of websites: (i) the most-accessed local news websites [5], (ii) a highly

[5] www.thereocrd.com



(a) www.macleans.com

(b) www.therecord.com

(c) www.thestar.com

(d) www.canoe.com

(e) www.nytimes.com

Figure 3.5: The real-time energy footprint of a number of websites over a Wi-Fi connection

ranked international news website[6], and (iii) three samples of highly ranked Canadian magazine websites [7], according to Alexa [5]. We surfed the full version of each website, using the built-in *Android* web browser over *Wi-Fi* and *3G* connections. Figure. 3.5 illustrates the energy behaviour of a number of news websites.

First of all, we noticed a common energy-consumption footprint for mobile web browsing. As can be seen in Fig. 3.5, the energy behaviour starts with the phone's idling energy, which is the state before a user launches the web browser. This state is marked on the figures and found to be approximately 300 *mAmp*, and is followed by high-energy activities that last for some period of time. The high-energy-activities period represents the energy needed to; (i) launch the browser app, (ii) download the web content (the network/radio interface energy cost), (iii) render the received HTML file (the computational energy cost), and (iv) display what is being rendered and is ready to be displayed on the user screen. The next energy state represents the energy cost only of displaying the entire webpage content. In general, we found that browsing a full version of the website over Wi-Fi for 30 seconds consumes ∼60 to 73.9 Joules.

To correlate the energy activities with the displayed web content, we started from an end-user point of view and closely examined Fig. 3.5. We noticed, for instance, periodic energy spikes that occur approximately every 5 seconds in Fig. 3.5(b). To investigate the real cause of such an energy activity, we looked into the contents displayed on the phone's screen and found that these spikes are due to displaying an ad content. This ad content is basically a dynamic ad that displays different ads every 5 seconds. We then went further, to the application layer, and investigated the *HTML* document that contains this ad content. We found out that this ad content is made of *JavaScript* piece of code, such a content requires high computational power to render and display on the screen. Thus, such activities cause extra energy consumption.

3.3.1.1 Energy-Bandwidth Mapping in Webpages

In an attempt to understand the energy behaviour of web browsing, we correlated the energy consumption footprint to the network activities needed to download the web content. Doing so, we can identify how long fetching a web component takes, and consequently, how much energy doing so consumes. Using the test bench described in section 2.4.2, we conducted an offline analysis of the traffic being transferred from and to the smartphone while a user is accessing the web. Figure. 3.6 maps the network activities (that is, the loading time for each TCP connection) onto the consumed energy. As was expected, we

[6] www.nytimes.com

[7] www.macleans.com, www,thestar.com, and www.canoe.com

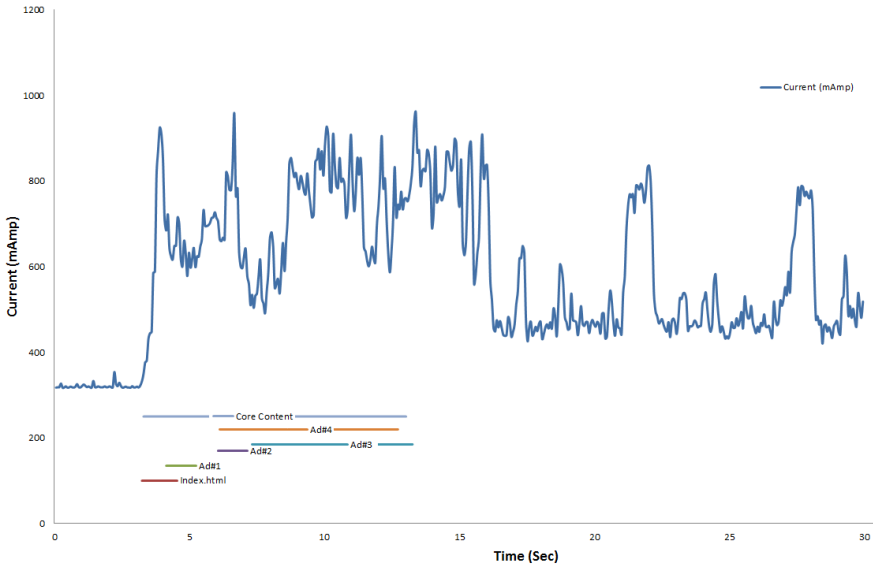


Figure 3.6: A sample of Energy-Bandwidth Mapping in a Webpage

notice that there are multiple TCP connections are established in parallel. Multiple parallel TCP connections are typically needed to speed up the retrieval of web content. The first content that the browser downloads is typically the index HTML documents, as shown in Fig. 3.6. Once this document is downloaded, the browser starts to parse and script its content. By doing so, the browser finds other web resources (images, CSS, JavaScript files, ads, *etc*) to be fetched. Consequently, more TCP connections are established. Now, since the core web content and the ad content come from different servers, in some cases 3 to 13 TCP connections were found to be dedicated to ad servers and to analytic servers that gather statistic and other information about users. Table. 3.1 lists the number of TCP connections (# of servers) needed for advertising purposes.

As explained in Section. 2.3, for the browser to display webpages, all web content referenced/tagged in the index HTML document, certain processes are required. The browser has to (i) resolve the DNS of the URL that is referencing that web object, (ii) handle the HTTP request that is to be made, (iii) establish the TCP connection with the server containing that object, and (iv) render that object based on the rules specified in the CSS. These time-consuming processes are shown in our figure to be under the time period of a high energy state, which is not very surprising considering the high computations required by the browser to accomplish its task, *i.e.* displaying the requested webpages. We noticed also that the overall time needed to download all the webpage content is around

10 seconds, in some cases. Approximately 2 seconds beyond those 10 seconds show high energy activity; we believe that is due to the computation needed to render the remaining fetched objects (objects that are downloaded with no network activities are involved). We found it very difficult to separate the energy consumed by the rendering process and the radio/network interface activity. Moreover, it is important to mention here that the objective of this thesis is to quantify the web advertisement energy and bandwidth impact, not to focus on analysing the energy consumption of web browsers.

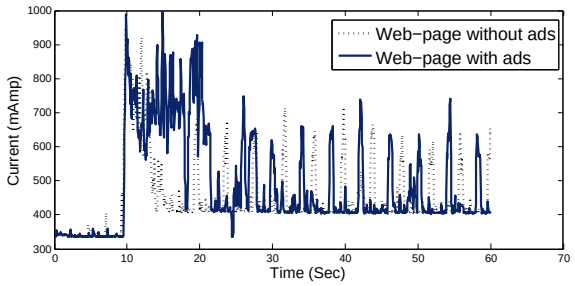
Table 3.1: Summary the traffic statistics while web browsing with and without ads

Website	# of servers with Ads	# of servers without Ads	Time download the page with ads(Sec)	Time download the page without ads	Energy difference (J)
www.therecord.com	17	9	10.5	8	15
www.nytimes.com	23	19	10	6	6.2
www.macleans.com	35	22	7	11	6.1
www.thestar.com	29	17	7.5	5	10.6
www.canoe.com	19	16	15.5	15	4.5

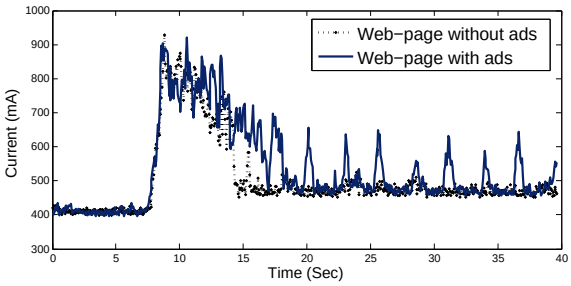
To identify the relationship between the number of TCP connections and the corresponding energy consumption, we modified some webpages using our Ad-blocking strategy. Applying this strategy allowed us to change the number of TCP connections that take place. Then, we compared the energy consumption in both cases for each website, and found, as expected, that the more TCP connections we have, the more is energy consumed, as shown in Table 3.1. Moreover, as the number of TCP connections increases, the fetching loading time increases as well, and so does the energy consumption. Downloading the ad content prolongs the active period of the radio interface, and hence increases the energy consumption.

3.3.1.2 Ads Energy Impact over Wi-Fi

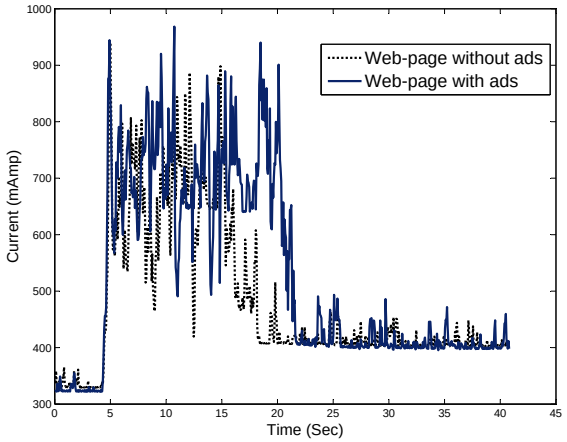
To study the impact of web advertising on energy, we started by browsing the full version of a number of websites over a Wi-Fi network, using the phone's built-in *Android Browser*. Then, we applied our Ad-blocking strategy, described in section 3.2.2, and repeated the same experiments that were conducted for the full versions of the websites. Next, we compared the energy needed to download web-pages with and without ads. Figure 3.7 shows the energy consumption for a number of websites. The dashed line refers to the web-pages that are ad-free and the solid line refers to the full version of web-pages (web-pages



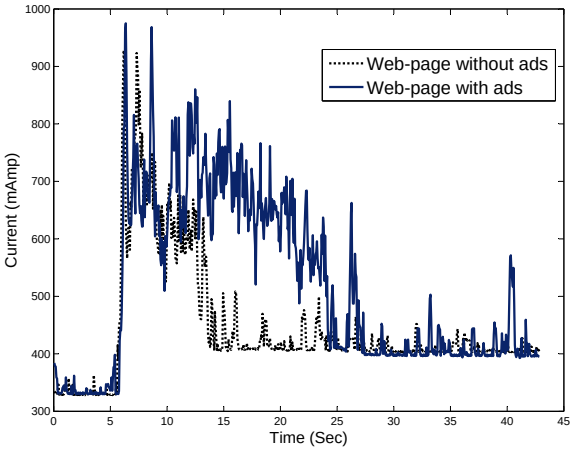
(a) www.macleans.com



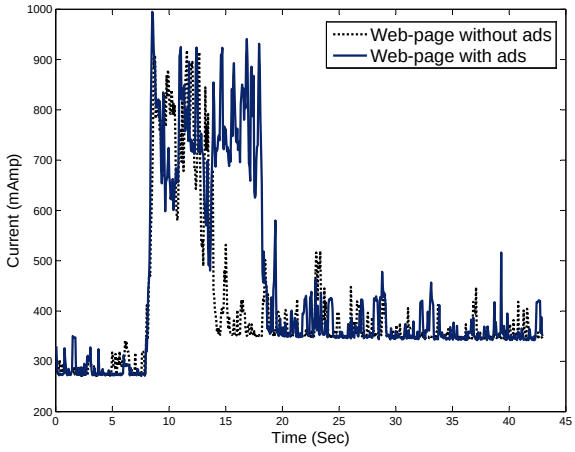
(b) www.therecord.com



(c) www.thestar.com



(d) www.canoe.com



(e) www.nytimes.com

Figure 3.7: The real-time energy footprint of a number of websites over a Wi-Fi connection with and without ads

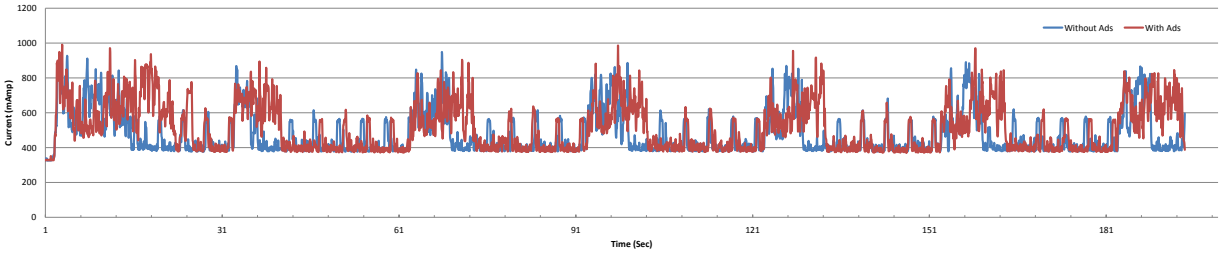


Figure 3.8: Energy measurement of our browser

with ads.) It is worth mentioning here that during our energy measurements we performed passive network sniffing over the Wi-Fi connection as well. Our offline analysis of the gathered network traces confirmed our previous argument that the more TCP connections there are, the longer the loading time is, and consequently, the more energy is consumed. As shown in Table 3.1, the energy overhead due to ads ranges from 4.5 to 15 Joules.

Reliability of Measurements

To ensure the reliability of our measurements, we developed a semi-web browser on *Android*. The architecture of the browser is explained in detail in section 4.2.1. To ensure more reliable measurements, the experiments were repeated multiple of times. We wanted to keep the same settings throughout all of the experiments, and most importantly, minimise end-user interaction with a phone. Therefore, we developed our own web browser. Doing so (1) enabled us to repeat the same experiments a number (7 times) of times at consistent time intervals (30 seconds); that is, every 30 seconds a request was made. (2) meant that no information was allowed to be cached. (3) minimized end-user interaction with the phone; that is, a single tap on the app's icon was enough to start the browsing. Figure 3.8 shows the energy footprint while using our browser app. Moreover, we conducted a number of tests just to ensure that our app energy consumption did not differ a lot from the built-in *Android Browser*. We found in many cases that both measurements were very close, and within some cases a 100 % match was noticed, as the measurements show in Table 3.2 and Table. 3.1.

Table 3.2 summarizes the energy consumed by a number of websites, with and without ads. The difference in energy consumption ranges from 6 to 17.5 %. Again, these figures are due what we call ad-overhead, that is, the net energy cost of the processes of downloading, rendering, and displaying web ads over a Wi-Fi network. These numbers are average values. We noticed that the ad contents vary each time a request is made, and observed

Table 3.2: The cost of ads over a Wi-Fi interface

Website	Content Type	Energy (J)	Energy Overhead due to Ads	Difference in the Estimated Battery Life (min)
www.therecord.com	browsing **with ads**	68.5	16%	27
	browsing **without ads**	59		
www.nytimes.com	browsing **with ads**	60	9%	23
	browsing **without ads**	55		
www.macleans.com	browsing **with ads**	65	12%	17
	browsing **without ads**	58		
www.thestar.com	browsing **with ads**	67	17.5%	35
	browsing **without ads**	57		
www.canoe.com	browsing **with ads**	61.5	6 %	8
	browsing **without ads**	58		

that webpages with ads take longer to load. As we shown in Fig. 3.7, around 4 to 10 seconds extra are needed to download and display ads. These times are reflected in the form of extra energy, as the measurements in Table 3.2 show. We compared the extra times introduced by the existence of ads just for downloading, (Table 3.1), and the overall time added due to downloading and rendering ads (Fig. 3.7.) In some cases, most ads energy consumption was due to the process of rendering. As the test case of browsing *www.canoe.com* shows, downloading the ads requires only half a second extra time, while the energy footprint in Fig. 3.7-(d) shows an extra 10 seconds of high activity energy state. Moreover, Table 3.2 shows an estimation of battery life, *i.e.*, how long the battery would last if we were to browse each version of the websites repeatedly until the battery died. These numbers illustrate how much energy is wasted by the existence of ads in web-pages.

3.3.1.3 Ads Energy Cost over 3G

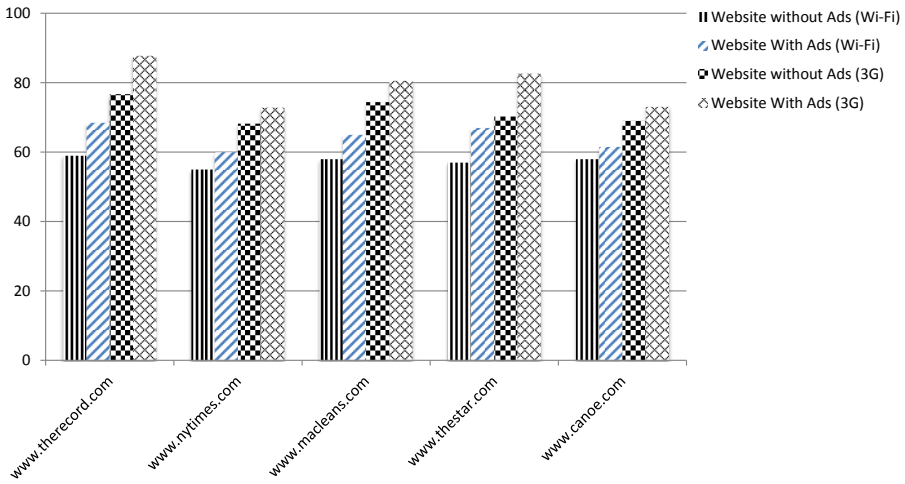


Figure 3.9: The energy consumption of different websites over 3G and Wi-Fi networks

We repeated the same experiments over a 3G connection. Table. 3.3 summarizes the energy consumed by a number of websites, with and without ads. We noticed that browsing over a 3G connection consumes more energy in general. The energy overhead due to web ads ranges from 6 to 17.5%. However, the energy difference due to ads is less than the difference noticed over Wi-Fi. Moreover, we observed that browsing a full version of a website, with ads, over Wi-Fi, is less expensive than browsing an ad-free version of the same website over 3G connection, as shown in Fig. 3.9. With 3G usage, batteries die faster

than with Wi-Fi, which was certainly expected taking into consideration the observations mentioned in this paragraph. The higher energy consumption over the 3G network is due to the energy tails and the capacity dissimilarity, according to [48]. "Energy tail" refers to the high energy state that the 3G radio interface has after the network activity is terminated, while the capacity dissimilarity refers to the 3G bandwidth capacity limit. Compared to Wi-Fi, 3G is slower, and consequently, downloading webpages over 3G will take a longer, resulting in higher energy consumption.

Table 3.3: The cost of ads over a 3G interface

Website	Content Type	Energy (J)	Energy Overhead due to Ads	Difference in the Estimated Battery Life (min)
www.therecord.com	browsing **with ads**	87.75	14%	22
	browsing **without ads**	76.74		
www.nytimes.com	browsing **with ads**	72.8	7%	15
	browsing **without ads**	68.25		
www.macleans.com	browsing **with ads**	80.55	8.23%	14
	browsing **without ads**	74.44		
www.thestar.com	browsing **with ads**	82.65	17.5%	28
	browsing **without ads**	70.29		
www.canoe.com	browsing **with ads**	73.05	6%	5
	browsing **without ads**	69.06		

When we considered browsing over the two network interfaces, 3G and Wi-Fi, we found a difference in battery life of, in some cases, one hour of operation. In other words, we could browse the web for an hour longer when using a Wi-Fi connection instead of 3G. Based on this observation, we came up with the idea of "Smart Mobile Web Browsing" that we explain in Chapter. 4.

3.3.2 Bandwidth Measurements Test Cases

Using the previously described testing infrastructure for monitoring network activities while mobile web browsing, we were able to capture the network traces. We browsed a number of news websites and conducted an offline analysis to measure how much bandwidth ad downloading requires.

3.3.2.1 Ads Bandwidth Cost

Table 3.4: Traffic breakdown per web-page

Web site	Total page traffic (**with ads**) in Bytes	Total page traffic (**without ads**)in Bytes	Total ads traffic in Bytes
www.therecord.com	3,025,082	1,449,882	1,575,200 (52%)
www.nytimes.com	975,646	634,018	341,628 (35%)
www.thestar.com	714,961	351,732	363,229 (50%)
www.macleans.com	987,682	617,857	369,825 (37%)
www.canoe.com	6,005,729	5,861,272	144,457 (2.5%)

Using Wireshark traces, we investigated the traffic needed to download ad components in webpages, and found that in one case, surprisingly enough, ad traffic comprises almost 50 % of the traffic needed to download some news webpages. More detailed traffic statistics are shown in Fig. 3.10, where we see that when ads are enabled, 8 extra servers are contacted. We can also see that the number of HTTP requests made by the client (smartphone) is almost three times more when ads are enabled, and the total number of packets exchanged between the client and the access point is doubled. Table 3.4 shows the overhead bytes needed to download ads. These numbers are average values since the ad contents vary each time a request is made.

In summary, we found that ads consume a considerable amount of energy and bandwidth. To have a better sense of the effective cost of ads, we, next, give an empirical estimation of (i) how much energy a smartphone would consume to download and display

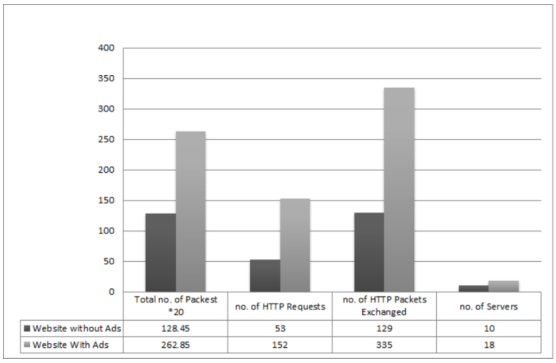


Figure 3.10: Traffic statistics while web browsing with and without ads.

ads, and (ii) how much bandwidth it would cost a user to download these ads for a certain period of time. We assume that a user spends around one hour on web browsing a day, not necessary continuously. In this hour, he/she would spend 2 minutes per website; therefore, on average, he/she would access at least 30 websites. Based on Tables 3.3 and 3.4, we get the following:

- the energy that a smartphone would consume, to download and display ads in one hour of browsing is 360 Jules/day; and
- the amount of bandwidth needed to download ads would be equal to 2.6 Mbyte/day, that is 78 Mbyte/month. Based on some US metered data plans (usage-based pricing plans) [60], $ 12.48 would be spent just on downloading ads.

Chapter 4

Smart Mobile Browsing Strategy

Web browsing is becoming very essential for every day life, especially with the ever increasing popularity of smartphones. The web content delivered to end users has witnessed drastic change. Webpages used to be simple "traditional" static pages compromised of only text and some images. However, modern webpages are dynamic and media-rich. The owners of these pages (publishers), moreover, rely largely on ads as a source of revenue. These ads are very media-rich and consume a lot of resources, like battery and bandwidth as we show in Chapter 3. Eliminating the existence of ads entirely from webpages is not practical knowing the big role they play in the web eco-system and in contributing to the availability of free web content. These factors have led to the demand for mobile browsing solutions that adapt webpages in a way that is energy and bandwidth efficient. In this chapter, we propose and implement a novel web browsing technique that adapts the webpages delivered to smartphone based on the smartphone's current battery level and network type. Webpages are adapted by controlling the number of ads to be displayed.

4.1 Motivation

The Smart Mobile Web Browsing (SWB) design is motivated by the measurements and the findings obtained in section 3.3. As mentioned there, the more ads a webpage contains and the more the complex ads get, the more resources are consumed. As ads can consume considerable energy while a user is mobile web browsing, in some webpages 18% of the total energy, and that the bandwidth overhead caused by downloading ads in some webpages reaches 50%, we propose the above mobile browsing technique, whose pages delivery

mechanism is function of a smartphone's resources and the amount of ads allowed to be displayed in the webpages.

Our approach mainly targets (i) extending smartphone battery life; and (ii) preserving the bandwidth needed to download the webpages. As mentioned in the previous section, having ads displayed in web-pages is essential for having free web content; therefore, our approach aims to balance the satisfaction of end users as well as that of webpage publishers.

4.2 Design requirements and objectives of SWB

This section presents the high-level concept and features of our Smart Mobile Web Browsing (*SWB*) strategy. The key design decisions are as follows.

- *Minimize the number of ads when needed.* As shown in section 3.3, ads can be very costly in terms of energy and bandwidth. Therefore, so that publishers do not exhaust their audience's smartphone's resources, the amount of ads to be delivered to the end user should be based on a device's available resources. *SWB* adapts the web-pages that are sent back to the smartphone based on a smartphone's current battery level and network type. The adaptation of the web content comes in the form of how many ads are allowed to be displayed on the webpage.

- *Minimum adjustment to the existing browsing systems.* The modifications required to be made on existing browsers and web servers are not complex. *SWB* implementation maintain this condition. For the server side, some plug-in modules can do the necessary job. And for the client side, some components can simply be added to existing browsers, as we explain in detail in section 4.2.1. The main challenge was in what layer should we embed the required information in the client side. We decided to choose the application due to the simplicity of adding optional fields in the HTTP protocol. This approach facilitates the need to make minimum changes to existing web browsers.

To achieve the above design goals, our solution strategy crystallizes in making the server intelligent enough to sense a smartphone's operational environment and provide it with customized webpages. Having done that, *SWB* can extend a smartphone's battery life while users still enjoy browsing the full versions of their favourite websites (full pages). This system also takes into account the type of network that the mobile client is connected to and accordingly provides it with full pages that require less bandwidth to download.

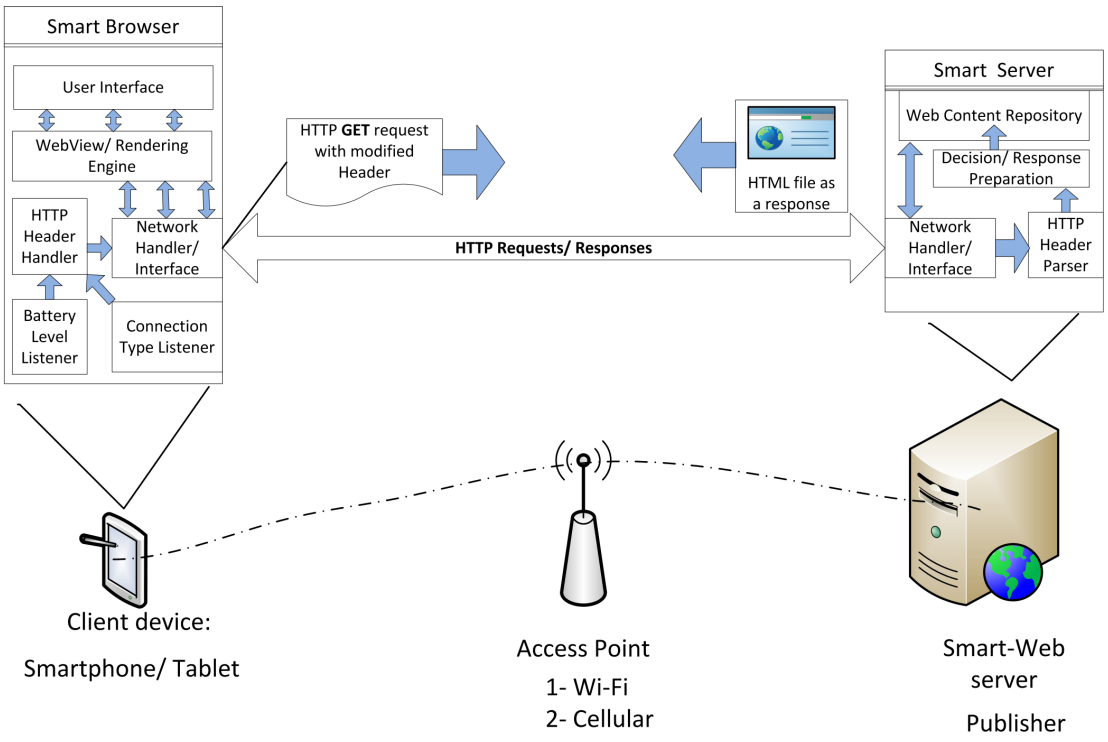


Figure 4.1: System Architecture

In this system, unlike other web adaptation solutions, *e.g.* [12], no user involvement is required, the client application (browser) sends the current available resources (Battery Level and Network Type) to the server, along with the HTTP request. Based on that information, the server sends web content back to the client.

4.2.1 System Architecture

In this subsection, we introduce the overall system architecture, and the decision policy that the server applies to prepare the response. Our approach is a client-server based system, as depicted in Fig. 4.1, similar to other conventional web-browsing techniques. The details of the system as follows:

- *Smart Browser*

As shown in Fig. 4.1, the client application *browser* consists of six components, namely User Interface, *WebView* engine, HTTP Header Handler, Network Interface Handler, Battery Level Listener, and Connection Type Listener. The first four components exist in all browsers; we present them separately to show exactly the internally basic components of conventional browsers, and how the our system can be implemented in general. The function of each component is described in Table 4.1.

Table 4.1: The functions of client's components

Component	Description and Function
User Interface	This displays what user sees and interacts with when running the browser.
WebView engine	This is an *Android* Java class that provides browser-like functionalities.
HTTP Header Handler	This handles the modification that we do to the HTTP request's header.
Network Interface Handler	This handles the HTTP requests that are made by the browser.
Battery Level Listener	This obtains smartphone battery level every time a user makes a web request, and passes it to HTTP Header Handler.
Connection Type Listener	This gets the type of network the smartphone is connected to and passes it to HTTP Header Handler.

- *Smart Server*

  The Server in our system can be any HTTP server. We added a HTTP Header Parser to extract the information that the browser includes with the HTTP request. Then the information is passed to the Decision and Response Preparation unit, where the policies are applied and decisions are made based on the current available resources on the smartphone. A proper webpage is next prepared and sent back to the client.

Note that when a user wants to access certain website, the browser gets the *Battery_Level* and *Connection_Type*, modifies the HTTP header, and sends the HTTP "GET"

Algorithm 1: Smart Web Browsing

Input:

Modified HTTP Request

Output:

Customised Web Content

Description:

1: *Server* $\Leftarrow$ *(i) receives the HTTP request, (ii) extracts* ***Battery_Level*** *and* ***Connection_Type*** *from the header*
2: **if** $Battery\_Level \geq 80 \wedge Connection\_Type = Wi - Fi$ **then**
3: return **full version of the website**
4: **else if** $Battery\_Level \geq 60 \wedge Connection\_Type = Wi - Fi$ **then**
5: return **Web-pages with only textual ads**
6: **else if** $Battery\_Level \leq 40 \wedge Connection\_Type = Wi - Fi$ **then**
7: return **ads free web-page**
8: **else if** $Battery\_Level \geq 80 \wedge Connection\_Type = Cellular$ **then**
9: return **Web-pages with only textual ads**
10: **else**
11: return **ads free web-page**
12: **end if**

request. On the server side, The pseudo-code shown in Algorithm 1 explains how the server works. The server receives the request, extracts the information needed to make the decision, applies certain policies, prepares the proper web-page, and then sends the web-page to the smartphone. Lines 2-12 of Algorithm 1 shows how the decision policy parameters can be set, .

4.3 System Implementation

We implemented a prototype system, in which the client side was implemented on *Android* and the server side was implemented in Java. Figure. 4.2 shows the GUI interfaces of the client side and the server side.

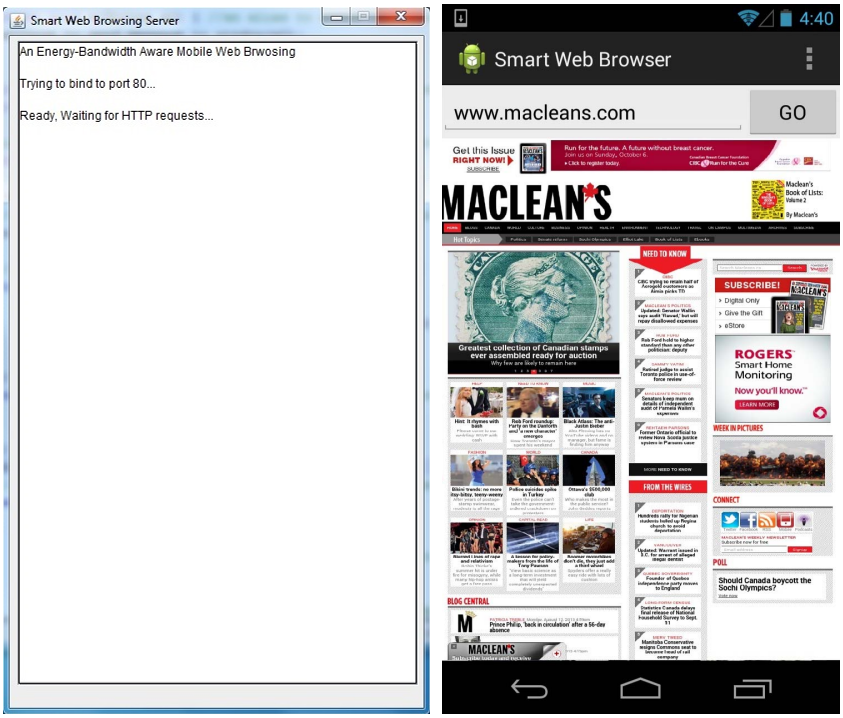
4.3.1 Client Side Configuration

The configurations that must be made on the client side (Browser) are described next:.

- *Battery Level Listener.* This listener's main duty is to obtain the smartphone battery level every time a user makes a web request. Code 4.3.1 shows the simplicity of the code to be added. The obtained battery level is then passed to the HTTP Header Handler.

```
public float getBatteryLevel() {
      Intent batteryIntent = registerReceiver(null, new
         IntentFilter(Intent.ACTION_BATTERY_CHANGED));
      int level = batteryIntent.getIntExtra(BatteryManager.EXTRA_LEVEL,
         -1);
      int scale = batteryIntent.getIntExtra(BatteryManager.EXTRA_SCALE,
         -1);

      if(level == -1 || scale == -1) {
         return 50.0f;
      }
      return ((float)level / (float)scale) * 100.0f;
   }
```



(a) Server GUI (b) Client GUI

Figure 4.2: The Graphical User Interface of the Smart Server and Smart Browser

- *Connection Type Listener.* This listener's main duty is to identify the type of network the smartphone is connected to and pass it to the HTTP Header Handler. Code 4.3.1 shows how we implemented this component. The obtained network type is also passed to the HTTP Header Handler to be included in the HTTP header.

```
ConnectivityManager conMan = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
                //mobile
               State mobile = conMan.getNetworkInfo(0).getState();
                //wifi
                State wifi = conMan.getNetworkInfo(1).getState();
                                String connection = new String();
                if (mobile == NetworkInfo.State.CONNECTED || mobile ==
                    NetworkInfo.State.CONNECTING) {
                      //mobile
                     connection = "mobile";
                   } else if (wifi == NetworkInfo.State.CONNECTED ||
                       wifi == NetworkInfo.State.CONNECTING) {
                       //wifi
                           connection = "wifi";
                   }
```

- *HTTP Header Handler.* This handler modifies the HTTP request's header. It inserts the battery level and the type of network into the optional fields of HTTP protocol. Figure. 4.3 (a) shows how the header looks after the new fields are added to the conventional HTTP header (Fig. 4.3 (b).) These snapshots are for our system, taken from Wireshark to illustrate the modifications the system makes into the HTTP packets. Figures 4.3 (a) and 4.3 (b) show the headers appearance before and after the modifications are made, respectively.

4.3.2 Server Side Configuration

We implemented an HTTP server in Java. The server, located in our lab, is deployed on a Dell desktop machine running *Windows 7*. For our configurations we added the (i) HTTP Header Parser that extracts the information that the browser includes with the HTTP request; (ii) Decision and Response Preparation unit, where the server applies the

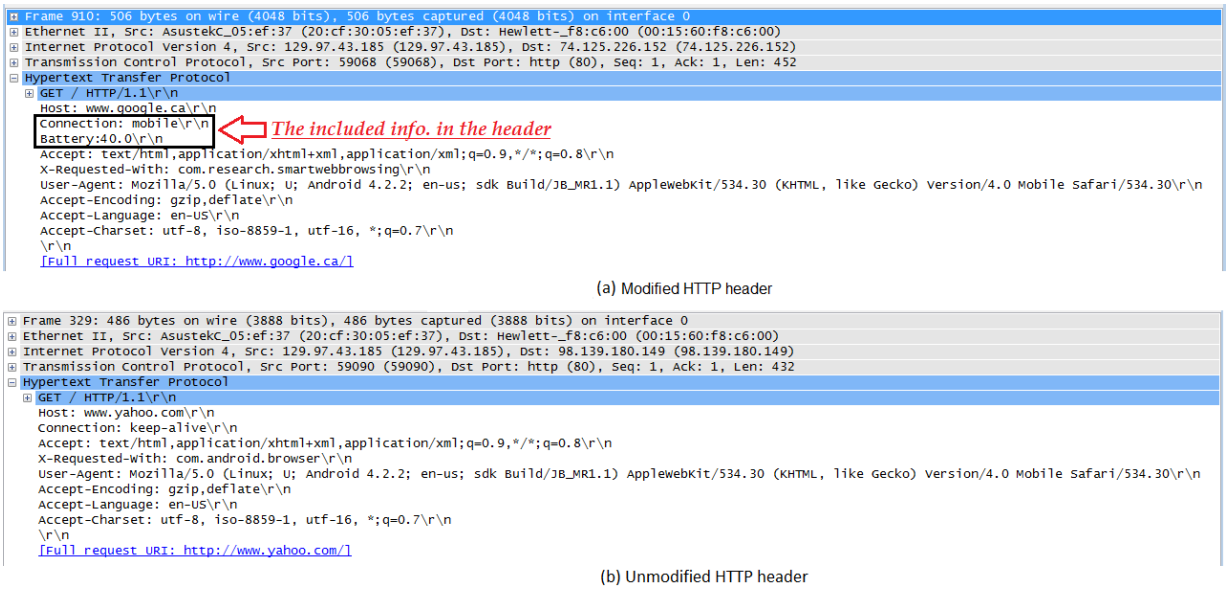


Figure 4.3: HTTP header modifications

specified policies. Accordingly, a proper web-page is prepared and sent back to the client based on its current available resources.

4.3.3 System Parameters

Certain parameters are associated with this mobile browsing technique, and are either end-user controlled ones or the smartphone's:

1. *Battery_Level* This parameter changes according to the amount of energy drained from the battery.

2. *Connection_Type* This parameter is user controlled. The user connects to a cellular or Wi-Fi network based on availability. The assumptions in this case are that (1) the cellular network is a metered plan, based on usage; and that (2) Wi-Fi network is completely free and requires no charge whatsoever.

These two parameters are used to evaluate how effective our system is, as shown in section.4.4.

4.4 Validation and Results

In this section, we validate our system by means of measurements. We compare its performance with the conventional web-browsing techniques and measure the energy and bandwidth consumption under different scenarios.

All the energy and bandwidth measurements were conducted using the test benches described in Chapter 3. Our Smart Browser was installed on a *Galaxy Nexus* smartphone running *Android V4.2.1.*. The results are presented and discussed in the following subsections.

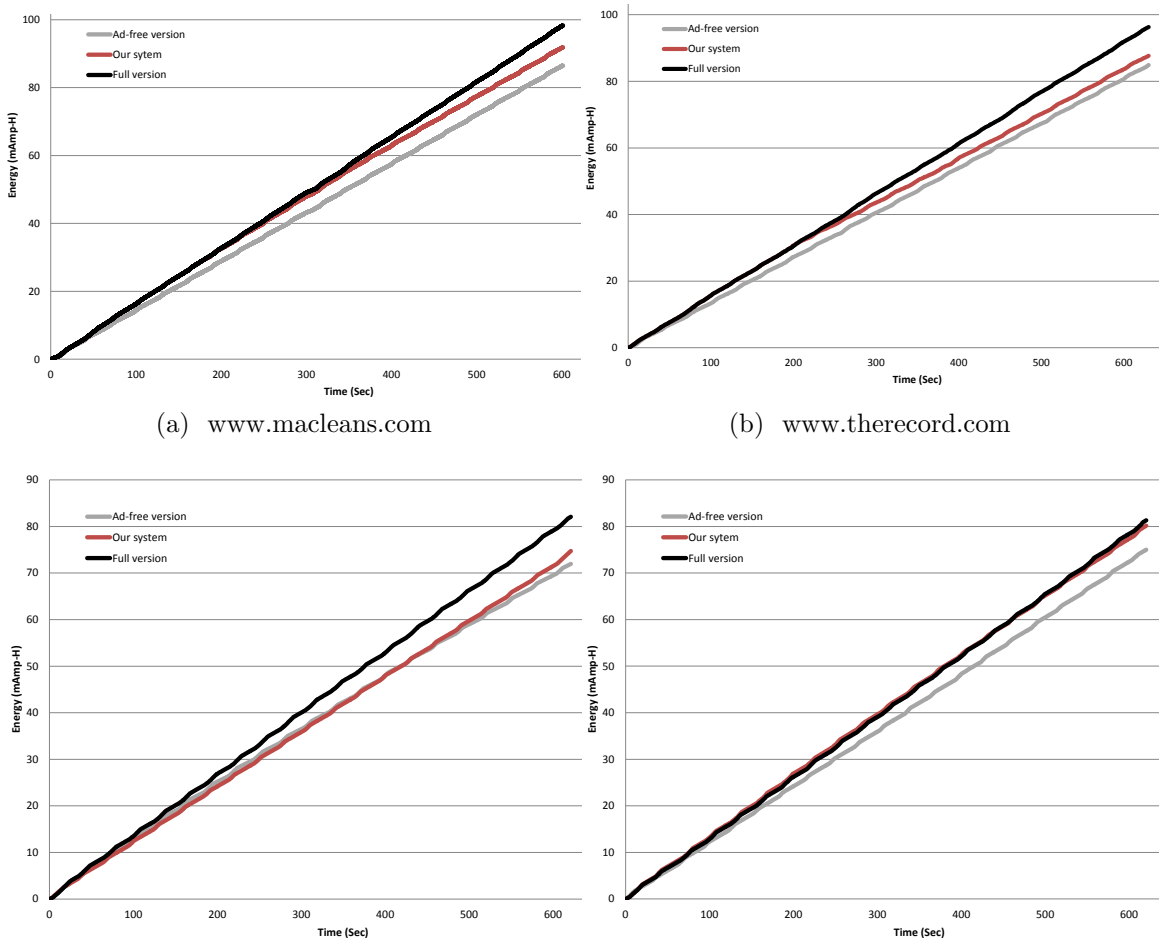
4.4.1 Energy and Bandwidth Results

The first evaluation scenario is as follows. We ran the Smart Browser for 10 minutes over a Wi-Fi network. During this period, a page request was being made every 30 seconds, for a total of 20 page requests. We set our server to deliver three kinds of web content: (1) the full version of the web site, (2) a webpage with only textual ads, (3) a webpage with no ads at all. In this scenario, the server sends back the full version for the first three minutes, then switches to send back the webpage with only textual ads for the next 3 minutes, and finally, it sends back the ad-free webpages for the last 4 minutes.

To compare our system with current browsing techniques, we disabled our system's features and browsed the same website again. We started by browsing the website's full version and redid the same browsing experience; that is, we requested the webpage repeatedly for 10 minutes. Using conventional browsing, however, the only type of content to be sent back was the full version of a website. Then, we applied the other extreme case, which is ad-free webpages, and redid the same browsing experience. Figure 4.4. shows the energy drained from the battery as a function of time for the four test cases (websites); by considering only the available *Battery_Level*, 2 to 12 % of the consumed energy can be saved using SWB.

Our second scenario was repeating the first experiment over a 3G network. Hence, the webpage adaptation decisions were made based only on the current *Battery_Level*. Figure. 4.5 shows the results obtained for the test cases.

The last scenario considers both parameters, *Battery_Level* and *Connection_Type*. This scenario shows the overall effectiveness of our system. Figure. 4.6 shows the overall performance of the system. In this scenario, the first case used only normal browsing (*i.e.*, browsing only the full version of a website, with no adaptation made) for 20 minutes in total, (10 minutes over Wi-Fi and the other 10 minutes over a 3G network.) The second



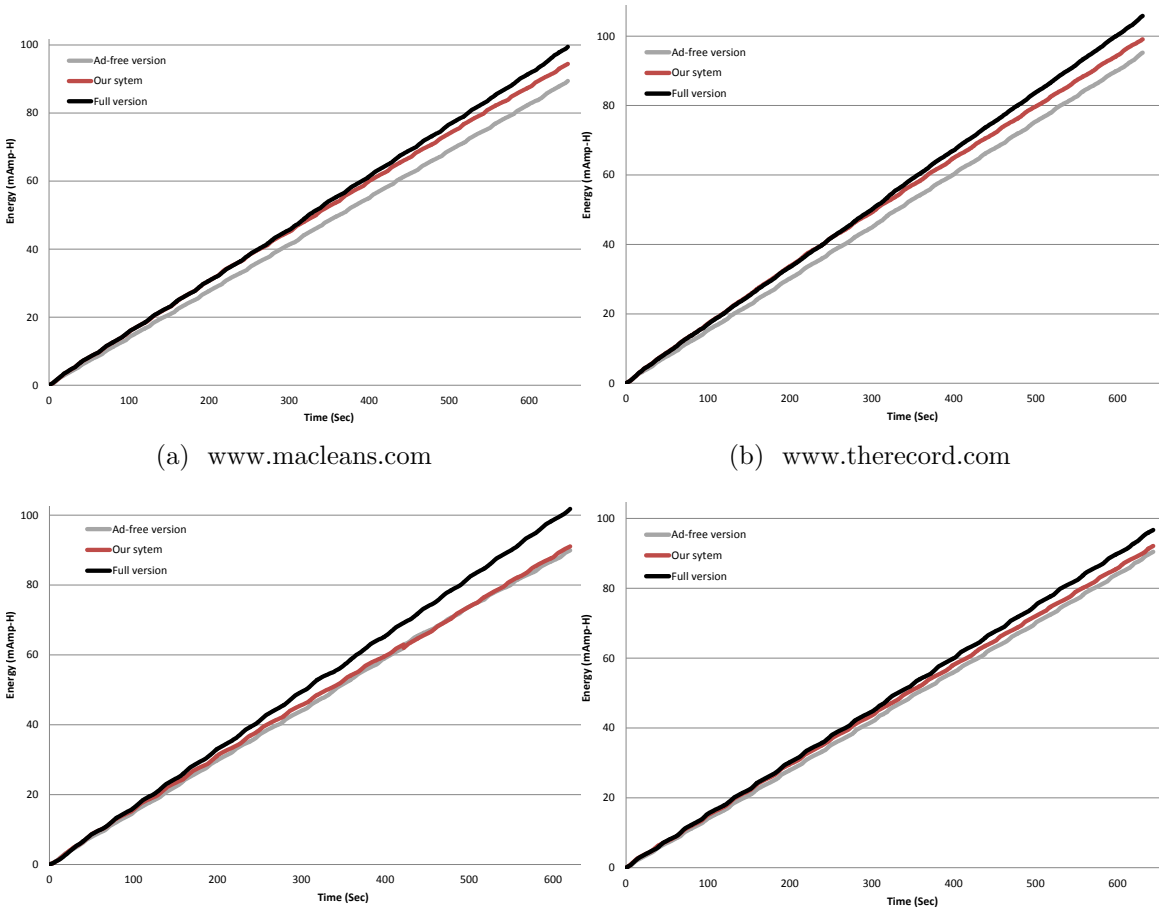
(a) www.macleans.com

(b) www.therecord.com

(c) www.thestar.com

(d) www.nytimes.com

Figure 4.4: Comparison of the energy consumption over a Wi-Fi connection



(a) www.macleans.com

(b) www.therecord.com

(c) www.thestar.com

(d) www.nytimes.com

Figure 4.5: Comparison of the energy consumption over a 3G connection

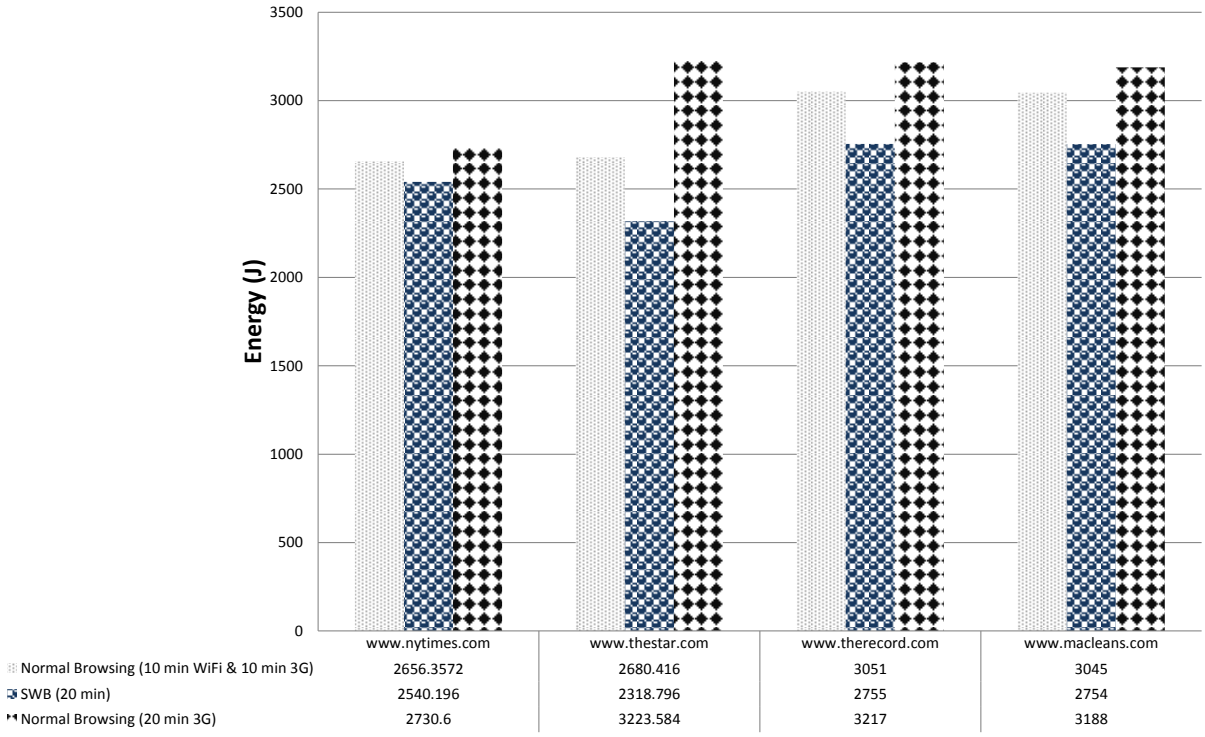


Figure 4.6: Energy consumption comparison between SWB and normal browsing

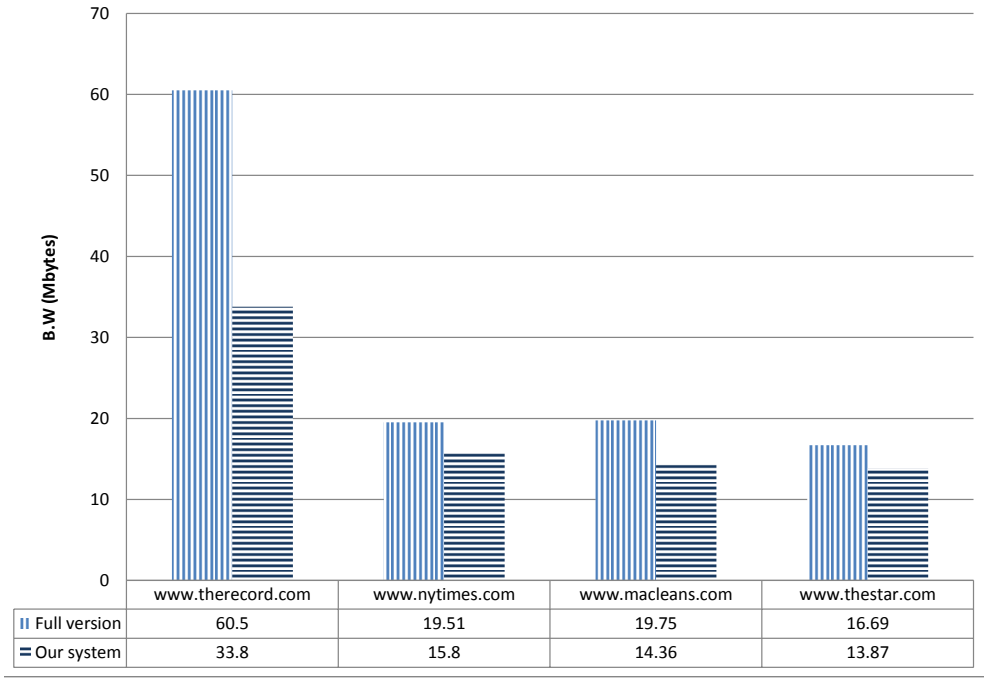
case was browsing using SWB, also 20 minutes in total. We found that SWB outperforms in this test case. The energy results in Fig. 4.6 can be used to estimate a complete discharging cycle of the battery; that is, consecutive webpage requests are being made until the battery dies. The battery of the *Galaxy Nexus* smartphone can supply energy up to 1,750 mAmp-H, or approximately 28,830 Joules. Based on this number and the obtained results in Fig. 4.6, Table. 4.2 presents a summary of our results under the three browsing cases: Case # 1, normal browsing, that is, browsing is done half of the time over a 3G network and the other half it is done over a Wi-Fi network; Case # 2, browsing using SWB for the whole time; Case # 3, normal browsing over 3G network only. Applying such scenarios, our results show that SWB can extend battery life up to an hour, compared to Case # 3.

Table 4.2: Summary of the estimated battery life under three different cases

	Battery Life Estimation in Hours				
Test Cases	Case # 1	SWB	Case # 3	Δ (SWB - Case # 3)	Δ (SWB - Case # 1)
nytimes	2.91	3.05	2.83	0.21	0.13
thestar	2.89	3.34	2.40	0.93	0.45
therecord	2.53	2.81	2.40	0.40	0.27
macleans	2.54	2.81	2.43	0.38	0.26

The bandwidth required to download webpages was measured in the test cases that required a Wi-Fi connection. We could not measure the bandwidth when a 3G network was used because Wireshark cannot be used for this purpose without extra tools. Also, we did not want to use the apps that capture the traffic of the phone, due to the fact that their operation will cause extra energy consumption, and thus, distort the energy measurements. Therefore, we only measured the consumed bandwidth over a Wi-Fi network. Figure. 4.7 shows the results obtained form running our system for ten minutes. In this period of time, a total of 20 webpage requests were made. The results confirms that SWB can save wireless bandwidth up to $\sim$ 44% compared to normal web browsing. Regarding the bandwidth consumed when a 3G network is used, we believe it will be the same as the case of Wi-Fi network since the content to be downloaded is the same.

We made empirical calculations to estimate how much money a user saves using SWB. According to [18], users spend approximately 53 minutes a day on mobile browsing. If we assume that he/ she spends a minute on each webpage, the total number of requested webpages would be 53. Based on our results in Fig. 4.7, $\sim$ 7 to 70 MBytes per day can be saved using SWB under this scenario. That number accumulates to $\sim$ 210 to 2100 Mbyte per month. Based on some metered data plans in US (usage-based pricing plans) reported



	www.therecord.com	www.nytimes.com	www.macleans.com	www.thestar.com
Full version	60.5	19.51	19.75	16.69
Our system	33.8	15.8	14.36	13.87

Figure 4.7: Bandwidth measurements

in [60], $\sim$ \$ 11.2 to 336 would be saved using our system.

Chapter 5

Conclusion

This thesis has reported work done on the impact of the existence of ads in mobile applications and in webpage from energy and bandwidth "monthly data usage" prospectives. Considerable amount of literature for the researchers and developers regarding the above issues are provided in Chapter. 2. It is concluded that different measurement studies have proven the adverse effect of ads on smartphone battery life and monthly data usage. However, having free contents or free apps is no way possible without the deployment of ads. Therefore, we surveyed the current solutions that attempt to keep the ads and at the same time alleviate their impact, either by considering only the energy issue or only the bandwidth issue, or both issues together.

In Chapter. 3, we focused on the energy and bandwidth impact of the existence of ads in webpages and noticed many interesting observations. We investigated the energy consumption due to ads in five well-known websites, and found that ads can consume $\sim$ 3.5 to 12 Joules over Wi-Fi and 3G networks, that is $\sim$ 6 to 18 % of the total energy of web browsing.

Then, we proposed a framework for mobile browsing that adapts the web-pages delivered to a smartphone, based on the smartphone's current battery level and the network type. The adaptation of the web content comes in the form of controlling the amount of ads to be displayed on the web-page. Moreover, our system targets: (i) extending smartphone battery life; and (ii) preserving the bandwidth needed to download the webpages, while balancing the satisfaction of the publishers of webpages as well as the end users. The architecture and implementation of the system is explained in detail, in Chapter. 4. By means of experimental validation, we showed that our system can extend smartphone battery life by up to $\sim$ 25% and save wireless bandwidth up to $\sim$ 45%.

In the future, the test bench of the evaluated websites can be further extended. Instead of evaluating only five websites, more can be covered. We will also consider fully automating the operation of the server side, in particular, the webpage adaptation process. Currently, the amount of ads embedded in webpages are manually controlled. For this thesis, we manually prepared a number of versions of the same webpage, in each webpage allowing certain amount of ads. In future, we will work on automating this process, which will require artificial intelligence. Machine learning can be applied to detect the ads on the full-version webpages, and accordingly, the amount of ads can be controlled based on the received current resources of the smartphone.

References

[1] Shark for root at google play. https://play.google.com/store/apps/details?id=lv.n3o.shark&hl=en. [Online; accessed 30-Mar-2013].

[2] Bell hspa/umts. http://mobilesyrup.com/2010/12/12/review-bell-novatel-wireless-u547-turbo-stick-42-mbps/, 2009.

[3] Mobile data traffic surpasses voice. http://www.ericsson.com/thecompany/press/releases/2010/03/1396928, 2010.

[4] http://www.itu.org/, 2012.

[5] Alexa, the web information company. http://www.alexa.com/, 2013.

[6] The world in 2013: Ict facts and figures. http://www.itu.int/ITU-D/ict/facts/material/ICTFactsFigures2013.pdf, 2013.

[7] A. Abogharaf, R. Palit, K. Naik, and A. Singh. A methodology for energy performance testing of smartphone applications. In *Automation of Software Test (AST), 2012 7th International Workshop on*, pages 110–116, 2012.

[8] Mobile Marketing Association et al. Mma code for responsible mobile marketing. http://www.barnesgraham.com/resource-centre/industry-news-and-articles/article-89/mma-code-for-responsible-moble-marketing, 2003.

[9] Howard Baldwin. Wireless bandwidth: Are we running out of room? http://www.computerworld.com/s/article/9223670/Wireless_bandwidth_Are_we_running_out_of_room_, 2012.

[10] Michael Butkiewicz, Harsha V. Madhyastha, and Vyas Sekar. Understanding website complexity: measurements, metrics, and implications. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011.

[11] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, page 21, Berkeley, CA, USA, 2010. USENIX Association.

[12] Sindhura Chava, Rachid Ennaji, Jay Chen, and Lakshminarayanan Subramanian. Cost-aware mobile web browsing. *Pervasive Computing, IEEE*, 11(3):34–42, 2012.

[13] N Chowdhury. A survey of search advertising, 2007.

[14] J.Y. Chung, Y. Choi, B. Park, and J.W.K. Hong. Measurement analysis of mobile traffic in enterprise networks. In *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pages 1–4. IEEE, 2011.

[15] Gerald Combs. Tshark-the wireshark network analyser. *URL http://www. wireshark. org*.

[16] Comscore. comScore Reports September 2012 U.S. Mobile Subscriber Market Share. http://www.comscore.com/Insights/Press_Releases/2012/11/comScore_Reports_September_2012_U.S._Mobile_Subscriber_Market_Share/, 2012. [Online; accessed 23-Seb-2012].

[17] Marco Cristo, Berthier Ribeiro-Neto, Paulo B Golgher, and Edleno Silva de Moura. Search advertising. In *Soft Computing in Web Information Retrieval*, pages 259–285. Springer, 2006.

[18] Mian Dong and Lin Zhong. Chameleon: a color-adaptive web browser for mobile oled displays. *Mobile Computing, IEEE Transactions on*, 11(5):724–738, 2012.

[19] Marwa Elfattah, Aliaa AA Youssif, and Ebada Sarhan. Handsets malware threats and facing techniques. *CoRR*, 2012.

[20] J. Erman, A. Gerber, and S. Sen. Http in the home: It is not just about pcs. *ACM SIGCOMM Computer Communication Review*, 41(1):90–95, 2011.

[21] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet measurement*, pages 281–287. ACM, 2010.

[22] Z. Fang, Y. Yang, F.M. Deng, and J. Cai. Research advances in mobile advertising areas. *Applied Mechanics and Materials*, 248:555–558, 2013.

[23] Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 101–112. ACM, 2012.

[24] S. Guha, A. Reznichenko, K. Tang, H. Haddadi, and P. Francis. Serving ads from localhost for performance, privacy, and profit. In *Proceedings of the 8th Workshop on Hot Topics in Networks (HotNets 09), New York, NY*, 2009.

[25] Kelly Hodgkins. Angry birds android to top $1 million per month in ad revenue. http://www.intomobile.com/2010/12/03/angry-birds-android-1-million-ad-revenue/, 2010.

[26] J. Hoehl and C. Lewis. Mobile web on the desktop: simpler web browsing. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, pages 263–264. ACM, 2011.

[27] Breg Insight. Mobile advertising and marketing. http://www.berginsight.com/News.aspx, 2011.

[28] V Jacobsen, Craig Leres, and Steven McCanne. Tcpdump/libpcap, 2005.

[29] Azeem J. Khan, V. Subbaraju, Archan Misra, and Srinivasan Seshan. Demo: context driven advertisement optimizer. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 487–488, New York, NY, USA, 2012. ACM.

[30] Azeem J. Khan, V. Subbaraju, Archan Misra, and Srinivasan Seshan. Mitigating the true cost of advertisement-supported "free" mobile applications. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 1:1–1:6, New York, NY, USA, 2012. ACM.

[31] Marc Langheinrich, Atsuyoshi Nakamura, Naoki Abe, Tomonari Kamba, and Yoshiyuki Koseki. Unintrusive customization techniques for web advertising. *Computer Networks*, pages 1259–1272, 1999.

[32] S.W. Lee, J.S. Park, H.S. Lee, and M.S. Kim. A study on smart-phone traffic analysis. In *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pages 1–7. IEEE, 2011.

[33] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. Don't kill my ads!: balancing privacy in an ad-supported mobile application market. In *Proceedings*

of the Twelfth Workshop on Mobile Computing Systems & Applications, page 2. ACM, 2012.

[34] Sean Ludwig. Vasona networks raises 12m to fix our mobile data congestion issues. http://venturebeat.com/2013/03/18/vasona-networks-funding/, 2013.

[35] G. Maier, F. Schneider, and A. Feldmann. A first look at mobile hand-held device traffic. In *Passive and Active Measurement*, pages 161–170. Springer, 2010.

[36] Amy Mitchell. THE EXPLOSION IN MOBILE AUDIENCES AND A CLOSE LOOK AT WHAT IT MEANS FOR NEWS. http://www.journalism.org/analysis_report/future_mobile_news, 2012. [Online; accessed 23-Mar-2012].

[37] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 317–328. ACM, 2012.

[38] mobiThinking. Global mobile statistics 2012 Part A: Mobile subscribers; handset market share; mobile operators. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#mobilephonepenetration/, 2012. [Online; accessed 23-Seb-2012].

[39] Prashanth Mohan, Suman Nath, and Oriana Riva. Prefetching mobile ads: Can advertising systems afford it? In *Proceedings of the 8th European Conference on Computer Systems (EuroSys' 13)*, 2013.

[40] Veelasha Moonsamy, Moutaz Alazab, and Lynn Batten. Towards an understanding of the impact of advertising on data leaks. *Int. J. Secur. Netw.*, pages 181–193, 2012.

[41] Rajesh Palit, Renuka Arya, Kshirasagar Naik, and Ajit Singh. Selection and execution of user level test cases for energy cost evaluation of smartphones. In *Proceedings of the 6th International Workshop on Automation of Software Test*, pages 84–90, 2011.

[42] A. Pathak, Y.C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[43] A. Pathak, Y.C. Hu, M. Zhang, P. Bahl, and Y.M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[44] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner. Addroid: privilege separation for applications and advertisers in android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ACM, 2012.

[45] G.P. Perrucci, F.H.P. Fitzek, G. Sasso, W. Kellerer, and J. Widmer. On the impact of 2g and 3g network usage for mobile phones' battery life. In *Wireless Conference, 2009. EW 2009. European*, pages 255 –259, 2009.

[46] Privoxy. Privoxy. *URL http://www.privoxy.org/*.

[47] I. Prochkova, V. Singh, and J.K. Nurminen. Energy cost of advertisements in mobile games on the android platform. In *Next Generation Mobile Applications, Services and Technologies (NGMAST), 2012 6th International Conference on*, pages 147 –152, Sep. 2012.

[48] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Characterizing radio resource allocation for 3g networks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 137–150. ACM, 2010.

[49] Ashwin Rao, Justine Sherry, Arnaud Legout, Arvind Krishnamurthy, Walid Dabbous, and David Choffnes. Meddle: middleboxes for increased transparency and control of mobile traffic. In *Proceedings of the 2012 ACM conference on CoNEXT student workshop*, pages 65–66. ACM, 2012.

[50] Grischa Schmiedl, Markus Seidl, and Klaus Temper. Mobile phone web browsing: a study on usage and usability of the mobile web. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, page 70. ACM, 2009.

[51] Aruna Seneviratne, Kanchana Thilakarathna, Suranga Seneviratne, Mohamed Ali Kaafar, and Prasant Mohapatra. Reconciling bitter rivals: Towards privacy-aware and bandwidth efficient mobile ads delivery networks. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–10. IEEE, 2013.

[52] Venkatesh Shankar and Marie Hollinger. Online and mobile advertising: current scenario, emerging trends, and future directions. *Marketing Science Institute*, pages 07–206, 2007.

[53] Maad Shatnawi and Nader Mohamed. Statistical techniques for online personalized advertising: a survey. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 680–687. ACM, 2012.

[54] Shashi Shekhar, Michael Dietz, and Dan S Wallach. Adsplit: Separating smartphone advertising from applications. *CoRR, abs/1202.4030*, 2012.

[55] RJG Simons and A. Pras. The hidden energy cost of web advertising. 2010.

[56] Monsoon Solutions. Power monitor. http://www.msoon.com/LabEquipment/PowerMonitor/.

[57] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J.P. Singh. Who killed my battery: analyzing mobile browser energy consumption. In *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012.

[58] N. Vallina-Rodriguez and J. Crowcroft. Energy management techniques in modern mobile handsets. *Communications Surveys Tutorials, IEEE*, 15(1):179–198, 2013.

[59] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, and J. Crowcroft. Breaking for commercials: Characterizing mobile advertising. pages 343–356, 2012.

[60] J. van den Brande and A. Pras. The costs of web advertisements while mobile browsing. In *Information and Communication Technologies*, pages 412–422. Springer, 2012.

[61] Jarred Walton. Browser face-off: Battery life explored. http://www.anandtech.com/show/2834/, 2009. [Online; accessed 19-Seb-2012].

[62] Yong Wang, Daniel Burgener, Aleksandar Kuzmanovic, and Gabriel Maciá-Fernández. Understanding the network and user-targeting properties of web advertising networks. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 613–622. IEEE, 2011.

[63] Z. Wang, F.X. Lin, L. Zhong, and M. Chishtie. Why are web browsers slow on smartphones? In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 91–96. ACM, 2011.

[64] Nick Wood. Mobile data traffic growth 10 times faster than fixed over next five years. http://www.totaltele.com/view.aspx?ID=448681, 2009. [Online; accessed 19-Seb-2012].

[65] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. Appscope: Application energy metering framework for android smartphone using kernel activity monitoring. 2012.

[66] L. Zhang, D. Gupta, and P. Mohapatra. How expensive are free smartphone apps? *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(3):21–32, 2012.

[67] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R.P. Dick, Z.M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.

[68] Y. Zhu, A. Srikanth, J. Leng, and V. Reddi. Exploiting webpage characteristics for energy-efficient mobile web browsing. *Computer Architecture Letters*, 2012.

[69] Yuhao Zhu and Vijay Janapa Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. *Network*, 2012.