Received September 5, 2014, accepted September 30, 2014, date of current version December 12, 2014.

Digital Object Identifier 10.1109/ACCESS.2014.2365091

SMoW: An Energy-Bandwidth Aware Web Browsing Technique for Smartphones

ABDURHMAN ALI ALBASIR AND KSHIRASAGAR NAIK
Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada
Corresponding author: K. Naik (snaik@uwaterloo.ca)

This work was supported by the Natural Sciences and Research Council of Canada.

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 favorite means of communication among users. In general, the available contents on smartphones, apps, and web, come in two versions: 1) free content that is monetized via advertisements (ads) and 2) paid content that is monetized by user subscription fees. However, the resources, namely, energy, bandwidth, and processing power, on-board are limited, and the existence of ads in websites and free apps can significantly increase the usage of these resources. These issues necessitate a good understanding of the mobile advertising eco-system and how such limited resources can be efficiently used. In this paper, we present the results of a novel web browsing technique that adapts the webpages delivered to smartphone, based on the 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 can extend smartphone battery life by up to ∼30% and save wireless bandwidth up to ∼44%.

INDEX TERMS Smarphones, energy efficiency, efficient web browsing, web adaptation.

I. 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 [1]. A significant percentage of these mobile subscribers already have a smartphone. 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 [2]. This statistics shows a penetration of close to 50%. Such growth and popularity are driven by the vast variety of applications (apps) now available. The rapid growth of smartphones combined with millions of apps makes them a favourite means of communication among users, but their functionalities are constrained by a number of challenges concerning battery life, bandwidth, and security.

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, there is increased interest in the investigation of the nature of smartphone traffic [3], [4]. These research studies focus on interesting questions: (i) how much traffic a device generates per day; (ii) which app contributes the most to 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) [5]–[7]. 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 information 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 hand-held devices [8]. In addition, more powerful processors, high resolution displays, huge numbers of applications, and advanced wireless technologies (LTE, WiMAX) are employed in today's smartphones and tablets. Such technologies and components drain energy at

high rates, and are considered as battery hungry. For instance, the authors of [9] 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.

The subject of this research was motivated by two observations: (i) *the high growing popularity of mobile web browsing*; and (ii) *the increasing complexity of webpages*. Recent statistics show that users prefer to use web browsers for digital news delivery. According to a survey conducted in the United States in 2012, 61% of smartphone users and 60% of tablet users use mainly browsers for reading news [10]. Thus, browsers are considered to be among the most popular apps. Regarding the increasing complexity of webpages, a study reported in [11] shows that over half of the webpages are not optimized for mobile browsers. Although many websites have mobile versions of their webpages, people tend to view the full (that is, desktop) versions to experience the richness of the contents. 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 [12], [13], that is, more computation intensive; and the existence of ads in webpages further increases the complexity. This complexity in smartphone environment, where resources (*e.g.,* battery and bandwidth) are limited, is reflected in longer loading times, more energy consumed, and more bytes transferred.

Therefore, with those observations 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 III-B to design a system that is energy and bandwidth efficient. In Section III-B we show that: (i) for some webpages, the energy cost of ads is about 18% of the energy cost of browsing the same pages without ads; and (ii) bandwidth overhead due to downloading ads is almost 50% of the total required bandwidth to download those pages without ads. Therefore, we propose a novel web browsing technique, called Smart Mobile Web (SMoW) browsing that takes into account two parameters: (i) the state of charge of the battery; (ii) the current network type (i.e., Wi-Fi or 3G).

SMoW mainly targets: (i) extending smartphone battery life; and (ii) preserving the bandwidth needed to download webpages. As mentioned previously, having ads displayed in webpages is essential to having free web contents; therefore, our approach aims to balance the satisfaction of end users and the interest of the webpage owners (publishers) to push ads.

With the aforementioned backdrop, this paper makes the following contributions:

- quantify the ad traffic generated during mobile web browsing;
- quantify the energy consumed to download and display ads on webpages while mobile web browsing;
- investigate the impact of using different networks, say 3G and Wi-Fi, on the energy consumed to download and display ads on webpages;
- propose a smart mobile browsing technique that aims to extend smartphone battery life and save wireless bandwidth by controlling the amount of ads to be displayed. This solution attempts to retain the ads, and, at the same time, alleviates its impact on the smartphone's limited resources.

The rest of the paper are organized as follows. The related work and background are presented in Section II. Section III describes the testing methodology used and the experiments carried out to achieve the first two contributions of this research; it also presents and discusses the measurements of energy and bandwidth expanded on ads. Section IV describes the proposed smart mobile web browsing framework and the validation of results. Finally, Section V lists some conclusions of this paper and recommendations for future work.

II. BACKGROUND AND RELATED WORK

In this section we provide the background information related to web browsing, the impact of web advertising on smartphone resources, and the existing related work.

A. THE STRUCTURE OF WEBPAGES

Webpages basically are HTML files that are delivered to the end user's browser, which in turn, renders the components that the HTML file contains and displays them on the user's screen in a readable way. Tags are used in HTML files to reference these components, distinguish them from each other, and let the browser understand what to do with each and every component. Zooming in into the HTML files to see what kind of components it contains, we find: (i) textual information; (ii) images; (iii) Cascading Style Sheets (CSS); and (iv) JavaScript code. From an end user's view, what she sees are some information that she is interested in, and some other ''unwanted'' information (ads), as shown in Fig. 1. The big arrows in the figure highlight those ''unwanted'' information (ads.) From an end user's prospective, we call the ads ''unwanted'' forced information because users mostly see them as a source of annoyance. Besides, ads are downloaded and displayed whether or not a user wants to see them.

A simplified view of how things work while web browsing is depicted in Fig. 2. The figure shows only the main players. It starts with an end user fetching content from a certain website (publisher, *e.g.* www.mydictionary.com). The browser makes the necessary DNS (Domain Name System) look up to get the IP (Internet Protocol) address of the website, establishes a TCP connection with the server, and then sends an HTTP *GET* request to that server (website). The publisher's server sends back an HTML file that contains the page contents and other components, namely, CSS files, images, and JavaScript code. The browser at the end user side starts rendering the HTML file, and it finds tags for other components that need to be fetched. It sends more HTTP *GET* requests to retrieve these elements. Ads come into the picture in the form of HTML or JavaScript code. Once such code is triggered, while the browser renders the HTML file,

FIGURE 1. Example of webpage contents.

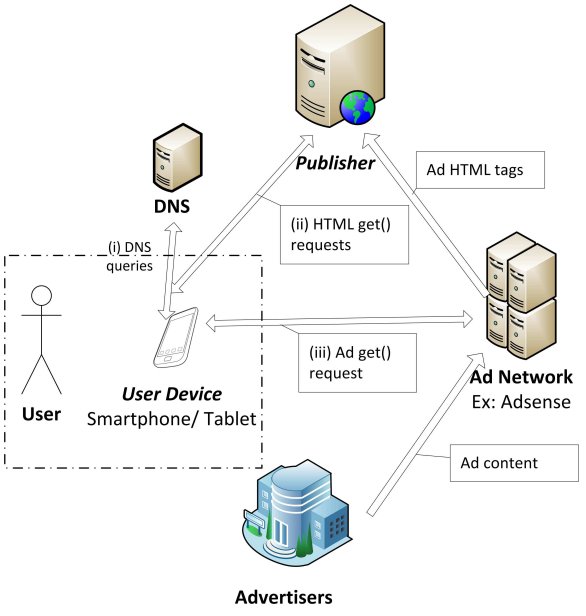

FIGURE 2. A high level view of how web browsing works.

another TCP connection is established with a specific ad network server (Ad network, *e.g.* AdSense) followed by an HTTP *GET* request to download ad contents. The ad sever figures out what sort of ads should be sent to the user based on a matching technique that is done with the publisher's webpage content (Contextual targeting.) If a user finds an interesting ad and clicks on it, she will be redirected to the advertiser's webpage. Behind the scenes, advertisers make agreements with the ad network that specify 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 some regulations on what kind of ads are allowed to be published on their pages. Ad networks, in turn, provide the publisher (website) with a piece of JavaScript code or HTML tags that the publisher embeds in its webpages.

B. THE IMPACT OF WEB ADVERTISING ON SMARTPHONES

Appearance of ads on webpages and mobile apps is essential and necessary for having free web contents and free apps. The features that smartphones offer make them a suitable advertising platform. In addition, the ever increasing popularity of smartphones is an indication that there is a huge opportunity that advertisers can reach the audience almost effortlessly. Advertisers have the capability to instantly target the consumers individually by leveraging the GPS (Global Positioning System) location and the search history of web-browsers and map apps.

Ads come in different formats in the webpages: some can be just simple textual ads and others can be very complex dynamic ads. An example of complex dynamic ads is having an ad banner that shows ads that a user can interact with; such ads are rich in animation and computation intensive. As mentioned in Subsection II-A, ads are embedded in the main HTML files in the form of JavaScript code or an HTML tag code. Such codes are computationally intensive, and, consequently, consume much of energy and require high bandwidth to download (*e.g.* JavaScript components, on average, may occupy ∼200 Kbytes [14]). The more ads a webpage contains and the more the ads get complex, the more resources are wasted. Therefore, the involvement of third-party advertisements, as a monetization technique, in webpages as well as in mobile apps increases the seriousness of the energy and bandwidth issues on the available limited resources of smartphones.

C. RELATED WORK

Recently, there have been several studies that focused on better mobile web browsing experience. Some proposals are content adaptation-based solutions [14], [15], and others manipulate the way browsers download the web contents and the resulting latency [11], [16].

Chava *et al.* [14] have addressed the adaptation of webpages based on the user's data plan and usage levels. Their solution aims to reduce the cost of web browsing by monitoring the user's web usage and the user's data plan quota to ensure that the user does not exceed the specified data limit (data quota.) Their system requires a network

middle-box (proxy) to: (i) manage the required adaptation each time a web request is made; and (ii) track the user data usage to ensure that the data limit is not exceeded. Dubey *et al.* [15] proposed a framework based on pre-defining the loading order of the webpage components. Their content creator approach works in a way that lets the user be part of the web content adaptation process. Therefore, based on user's preferences, the content creator can prioritize the importance of webpage items. The loading of these items is done according to the highest priority to the lowest. By doing so, user's experience can be enhanced.

Zhao *et al.* [16] proposed an energy-aware web browsing solution for smartphones while connecting to 3G networks. They leverage two main properties of web browsing on 3G networks: (i) the computation sequence of loading web contents by the browsers; and (ii) the different power states that a smartphone has while connecting to a 3G network. The idea is to download all the contents in one attempt instead of downloading them in multiple attempts. In that case, the 3G radio interface is in a high power state till all the web components are fetched. Once all the components are downloaded, the radio interface goes into a low power state. The browser, in turn, runs the computations needed to render the downloaded web contents for the user. As a result, the required power is lowered and the latency (time needed to load webpages,) as well. In reference [11], Wang *et al.* critically analyse the slowness of web browsers on smartphones. They found that the main reason for the slow web browsing is the process of fetching different web contents.

Some other solutions related to energy efficiency in smartphones were proposed in [17] and [18]. Huang *et al.* [17] aim to reduce the number of HTTP requests and responses made by mobile mesh applications, and, consequently, lower the energy consumption as well as speed up the web resource retrieval process.

III. MOTIVATIONAL STUDY

This section contains the main study motivating the design of Smart Mobile Web Browsing (SMoW) system. This section describes a number of experiments carried out to achieve the first two contributions of this paper (see Section I.) Subsection III-A details our testing methodology and test benches; finally, the results and measurements of energy and bandwidth are presented and analysed in Section III-B.

A. TESTING METHODOLOGY

To perform testing, the first challenge was to separate the web contents. In other words, we wanted a way to separate the ad contents from the original webpage, so that we can test the smartphone once with the existing of ads and a second time without ads. As a result, we can measure exactly the energy and bandwidth cost of ads by calculating the difference between the two cases. A method was sought to block or insulate the ads from the webpages. The available blocking techniques are as follows.

1) THE AVAILABLE AD-BLOCKING TECHNIQUES

- *Ad-blocking Applications:* A number of apps available online can block ads, namely, Ad-Vanish Lite, NoRoot Ad-Remover Lite, and AdFree. These apps work by turning off the Internet connection completely. They are designed to block ads from *Android* games and they work as follows: the user is requested to prepare an Ad-block list, so if she wants to stop displaying ads on a game all she needs to do is to 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 an Internet connection. In the case of web browsing, establishing an Internet connection is essential. Therefore, this Ad blocking option does not satisfy our testing needs.
- *Browser Plug-in Ad-blockers:* Mobile web browsers such as Mozilla Firefox and Android can use plug-in ad-blockers to block ads. Some plug-in apps are available online (Adblock Plus) and they filter out the ad URLs. These plug-ins maintain 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. 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 requires extra activities and computational power. As a result, extra energy is consumed. For our case, where we are looking to measure the actual energy and bandwidth required to download and display ads, using this option will give us distorted measurements. (ii) In addition, 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.) As a result, some ads will still be displayed while adopting this option. (iii) The ad links (ad tags) in a webpage are already delivered to the smartphone and parsed by the browser; as a result, some bandwidth and energy would have been consumed. For these reasons, we eliminated this option as well.
- *Proxies:* Proxies, namely, Privoxy [19], 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 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 needed to achieve the goal of blocking ads without engaging the smartphone in this process.

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 webpages before they are even requested by the end user. The process works as follows: (1) choose certain web-sites; (2) download them and make two copies of every one; (3) modify and remove ad codes found in one copy and keep the other one as is; and (4) upload one copy to the server at a time to perform testing. We modify the the webpages manually by removing all the ad-related contents from the requested HTML documents. Figure 3 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.

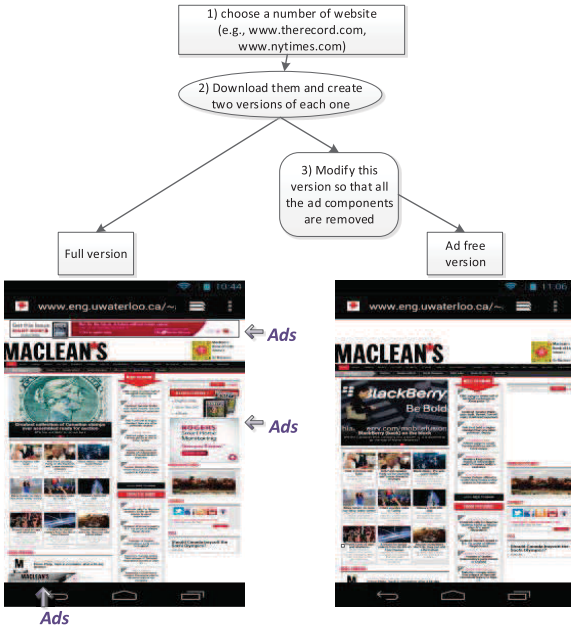

FIGURE 3. Ad blocking strategy.

3) 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. 4. We use a Monsoon power monitor [20] to power the smartphone and accurately measure the current drawn by the smartphone in real-time. The power monitor is connected to a personal computer (PC) where the measurements are received and stored for off-line analysis. 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 consistent measurements.

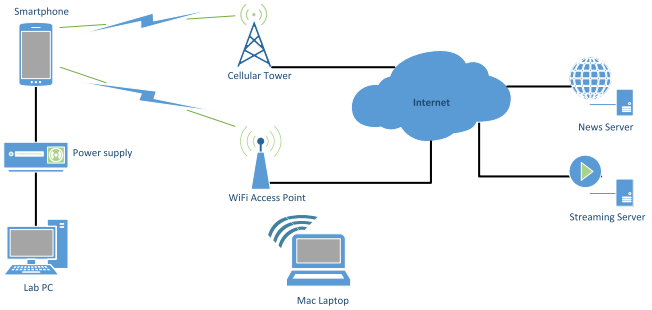

FIGURE 4. Measurement Set-up.

To conduct the experiments, we chose the built-in *Android Browser* on *Galaxy Nexus* smartphone and another browsing application caslled "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 a *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 of 7–14 Mbit/s [21]. For the phone configurations, we followed the methodology proposed in reference [22], in order to have consistent measurements.

To ensure that the measured energy is the real cost of requesting webpages, we make sure that 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.)

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. 4. We installed Wireshark on a Mac-book laptop to leverage the *Monitor Mode* feature available on Mac-book laptops. This feature allows the packet sniffing task to be 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 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.

B. MEASUREMENTS OF COSTS OF WEB ADVERTISEMENTS ON SMARTPHONES

1) TEST CASES FOR ENERGY MEASUREMENTS

Using the described testing infrastructure for measuring the energy consumption we were able to measure the instantaneous current drawn from the battery 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 paper 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 constant voltage, and t is the time duration.

We chose a number of websites: (i) the most-accessed local news websites; (ii) a highly ranked international news

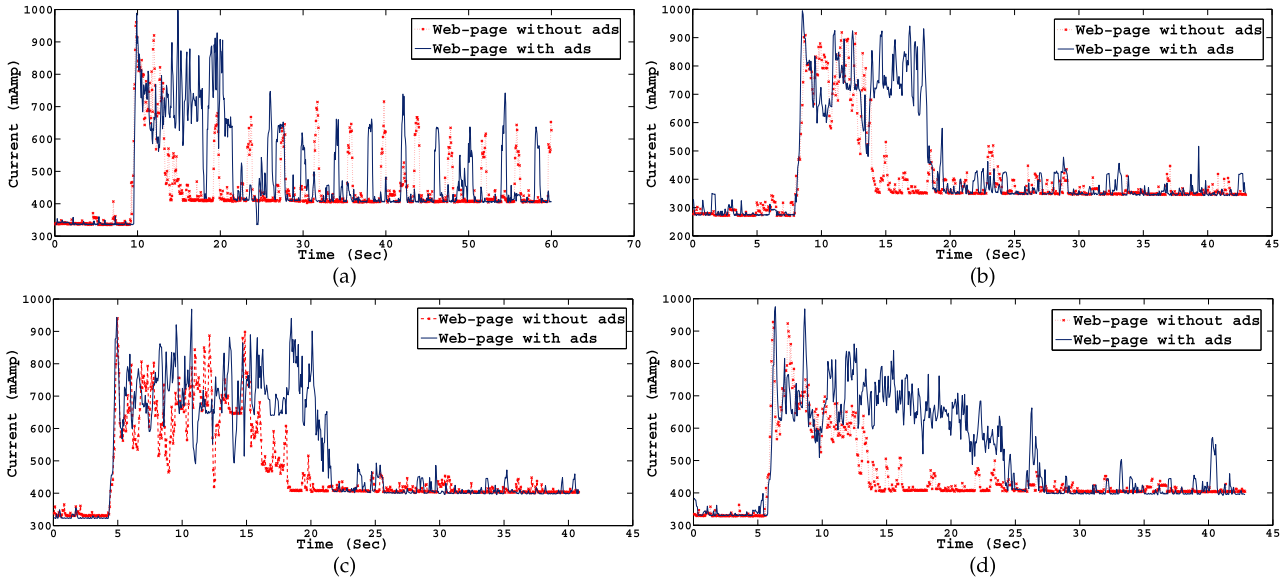

FIGURE 5. The power footprint of four websites over a Wi-Fi connection with and without ads. (a) www.macleans.com. (b) www.nytimes.com. (c) www.thestar.com. (d) www.canoe.com.

website; and (iii) three samples of highly ranked Canadian magazine websites, according to Alexa [23]. We surfed the full version of each website, using *Android* web browser over *Wi-Fi* and *3G* connections. Figure 5 (the solid line) 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. 5 (the solid line), 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 *mA (milli Ampere)*, and is followed by high-energy activities that last for some of time. The high-energy-activity 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 contents, we started from an end-user point of view and closely examined Fig. 5. We noticed, for instance, that periodic energy spikes occur approximately every 5 seconds in Fig. 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 corresponds to a block of *JavaScript* code, and it requires high computational power to render and display.

a: 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 web contents. Doing so, we can identify how long fetching a web component takes, and consequently, how much energy doing so consumes. Using the test bench, we conducted an offline analysis of the traffic being transferred from and to the smartphone while a user is accessing the web. We, then, mapped the network activities (that is, the loading time for each TCP connection) onto the consumed energy. As was expected, we notice that there are multiple TCP connections established in parallel. Multiple parallel TCP connections are typically needed to speed up the retrieval of web contents. The first content that the browser downloads is typically the index HTML document. Once this document is downloaded, the browser starts to parse and script its contents. By doing so, the browser finds other web resources (images, CSS, JavaScript files, and ads) 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 statistics and other information about users. Table 1 lists the number of TCP connections (# of servers) needed for advertising purposes.

As explained briefly in Section I, for the browser to display webpages, all web contents referenced/tagged in the index HTML document, certain processes are required. The browser has to: (i) obtain the IP (Internet Protocol) address corresponding to the URL referencing an web object; (ii) handle the HTTP request that is to be made; (iii) establish the TCP connection with the server containing that web object; and (iv) render that object based on the rules specified in the CSS. These time-consuming processes were found to be

TABLE 1. Summary the traffic statistics while web browsing with and without ads.

Websites	# of servers with Ads	# of servers without Ads	Time to download the page with ads (Sec.)	Time to download the page without ads (Sec.)	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

under the time period of a high energy state, which is not very surprising considering the high computations required by the browser to accomplish the task of displaying the requested webpages. We noticed also that the overall time needed to download all the webpage contents 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 paper is to quantify the web advertisement energy and bandwidth impacts and not to focus on analysing the energy consumption of web browsers.

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 are opened. 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 energy is consumed, as shown in Table 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.

b: ENERGY IMPACT OF ADS 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 III-A.2, and repeated the same experiments that were conducted for the full versions of the websites. Next, we compared the energy needed to download webpages with and without ads. Figure 5 shows the energy consumption for a number of websites. The dashed line refers to the webpages that are ads-free and the solid line refers to the full version of webpages (webpages 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 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*. To ensure more reliable measurements, the experiments were repeated multiple times. We wanted to keep the same settings throughout the experiments, and, most importantly, minimise end-user interactions with the smartphone. Therefore, we developed our own web browser. Doing so: (i) enabled us to repeat the same experiments a number (7 times) of times at consistent time intervals of 30 seconds; (ii) meant that no information was allowed to be cached; and (iii) minimized end-user interactions with the phone; that is, a single tap on the app's icon was enough to start the browsing.

Figure 6 shows the power 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 in some cases a 100% match was noticed.

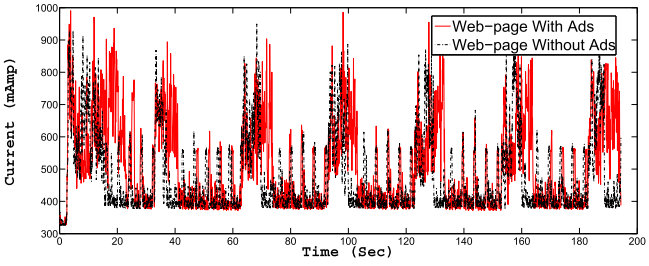

FIGURE 6. Energy measurement of our browser.

Table 2 summarizes the energy consumed by a number of websites, with and without ads. The difference in energy consumption ranges from 6–17.5%. 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 that webpages with ads take longer to load. As it can be seen in Fig. 5, around 4 to 10 seconds of extra time is needed to download and display ads. These times are reflected in the form of extra energy, as the measurements in Table 2 show. We compared the extra

TABLE 2. The cost of ads over a Wi-Fi/3G interfaces.

Website	Browsing	Energy (J) WiFi/3G	Energy Overhead due to Ads WiFi/3G	Difference in the Estimated Battery Life (min) WiFi/3G
www.therecord.com	browsing **with ads** browsing **without ads**	68.5/87.7 59/76.7	16%/14%	27/22
www.nytimes.com	browsing **with ads** browsing **without ads**	60/72.8 55/68.2	9%/7%	23/15
www.macleans.com	browsing **with ads** browsing **without ads**	65/80.5 58/74.4	12%/8.2%	17/14
www.thestar.com	browsing **with ads** browsing **without ads**	67/82.6 57/70.2	17.5%/17.5%	35/28
www.canoe.com	browsing **with ads** browsing **without ads**	61.5/73 58/69	6%/6%	8/5

times introduced by the existence of ads just for downloading (Table 1) and the overall time added due to downloading and rendering ads (Fig. 5.) In some cases, most ads energy consumptions were 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 power footprint in Fig. 5(d) shows an extra 10 seconds of high activity energy state. Moreover, Table 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 ads in webpages.

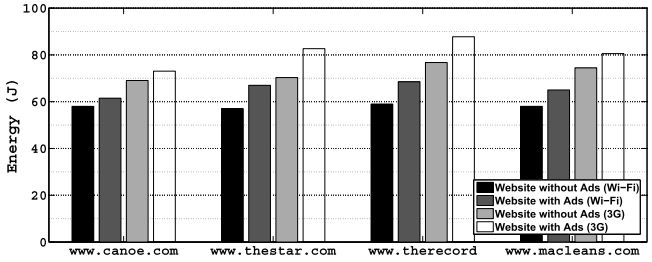

FIGURE 7. Energy cost of browsing five different websites over 3G and Wi-Fi networks.

c: ENERGY IMPACT OF ADS OVER 3G

We repeated the experiments of Section III-B.1.b over a 3G connection. Table 2 summarizes the energy consumed by browsing five websites, with and without ads. We noticed that browsing over a 3G connection consumes more energy than Wi-Fi 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 ads-free version of the same website over 3G connection, as shown in Fig. 7. With 3G usage, batteries die faster than with Wi-Fi, which was certainly expected taking into consideration the observations mentioned in the beginning of this paragraph. The higher energy consumption over the 3G network is due to the energy tails and the capacity dissimilarity, according to [24]. "Energy tail" refers to the high energy state that the 3G radio interface stays in 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 longer, resulting in higher energy consumption. 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 Section. IV.

2) TEST CASES FOR BANDWIDTH MEASUREMENT

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

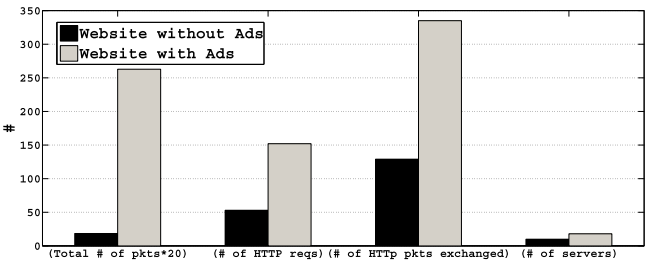

FIGURE 8. Traffic statistics while web browsing with and without ads.

a: BANDWIDTH COST OF ADS

Using Wireshark traces, we investigated the traffic needed to download ads components in webpages, and found that in one case, surprisingly enough, ads traffic comprised almost 50% of the traffic needed to download some news webpages. More detailed traffic statistics are shown in Fig. 8, where we

TABLE 3. Traffic breakdown per webpage.

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%)

see that when ads are enabled, 8 extra servers are contacted. We also observe 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 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 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 necessarily continuously. In this hour, she would spend 2 minutes per website; therefore, on average, she would access at least 30 websites. Based on Tables 2 and 3, 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) [25], $ 12.48 would be spent just on downloading ads.

IV. PROPOSED SOLUTION

Our Smart Mobile Web (*SMoW*) browsing technique is motivated by the measurements and findings of Section III-B. In the SMoW browsing technique, the page delivery mechanism is a function of the smartphone's energy and network-type resources and the amount of ads allowed to be displayed on the webpages. In other words, the solution strategy we take crystallizes in making the server smart enough to sense the smartphone's current operational resources and provide it with a customized webpage. In this system, unlike other web adaption solutions [14], no end user involvement is required; rather, the client application (browser) sends the current available resources, Battery Level and Network Type to the server, along with the HTTP request. Based on those information, the server sends back adapted web contents to the client. Adaptations of the web contents are effected in the form of how much ads are allowed to be displayed on the webpage. In [26], we have shown the framework of *SMoW* from a high-level view, and in this study we describe the proposed technique in more details with the support of implementation and the validation of results.

A. DESIGN REQUIREMENTS AND OBJECTIVES OF SMOW

This section presents the key design decisions of our *SMoW* browser as follows:

- *Minimize the number of ads when needed.* As shown in Section III-B, ads can be expensive in terms of energy and bandwidth. Therefore, the amount of ads to be delivered to the end user should be based on a device's available resources, so that publishers do not exhaust the smartphone's resources. *SMoW* adapts the webpages that are sent to the smartphone based on the device's current battery level and network type. A adaptation of web contents 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 to the existing browsers and web servers are not complex- and the *SMoW* technique satisfies the requirement. On the server side, the necessary changes can be packaged in the form of some plug-in modules that can be installed on the existing servers. And for the client side (browsers), the needed components can simply be added to existing browsers in the form of software updates. In Section IV-A.1, we explain the details of the required modifications needed to accommodate the features of *SMoW*. The main challenge was to decide the protocol layer where we embed the required information on the client side. We decided to choose the application layer due to the simplicity of adding optional fields in the header of the HTTP protocol. This approach facilitates the need to make minimum changes to existing web browsers.

To achieve the two design goals, our solution strategy requires the server to sense the smartphone's operational environment and provide the device with customized webpages. Having done that, *SMoW* 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

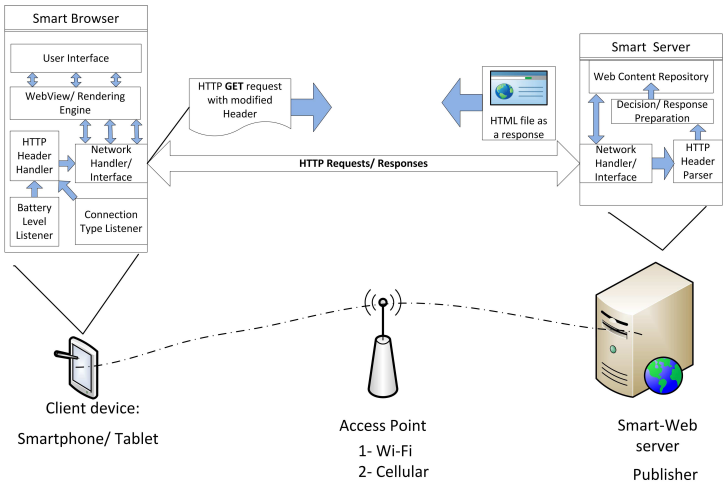

FIGURE 9. System Architecture.

connected to and accordingly provides it with full pages that require less bandwidth to download. In this system, unlike other web adaptation solutions, *e.g.* [14], no user involvement is required, and the client application (browser) sends the current available resources (Battery Level and Network Type) to the server in the HTTP requests. Based on that information, the server sends web content back to the client.

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. 9, similar to other conventional web-browsing techniques. The details of the system are as follows:

- **Smart Browser**: As shown in Fig. 9, 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 internal basic components of conventional browsers, and how our system can be implemented in general. The function of each component is described in Table 4.
- **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 in the HTTP requests. Then, the information is passed on 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. Next, a customized webpage is prepared and sent back to the client. The function of each component is described in Table 5.

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

TABLE 4. Functions of the client's components.

Component	Description and Function
User Interface	This displays what the 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 proposed modifications 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 that the smartphone is connected to and passes it to HTTP Header Handler.

TABLE 5. Functions of the server's components.

Component	Description and Function
Network Interface	This handles the received HTTP requests, and passes them to HTTP Header Parser.
HTTP Header Parser	This component extracts the embedded information from the received HTTP header and then passes it to Decision and Response Preparation Unit.
Decision and Response Preparation Unit	This unit applies certain policies and makes a decision about the type of web contents to be sent to the client.
Web Content Repository	This is a repository of the websites's resources (different versions of webpages).

On the server side, The server receives the request, extracts the information needed to make the decision, applies certain policies, adopts the webpage, and then sends the webpage to the smartphone. Table 6 explains the policies that server applies to make the adaptation decision. For instance, if a user makes a page request when the phone *Battery_Level* is

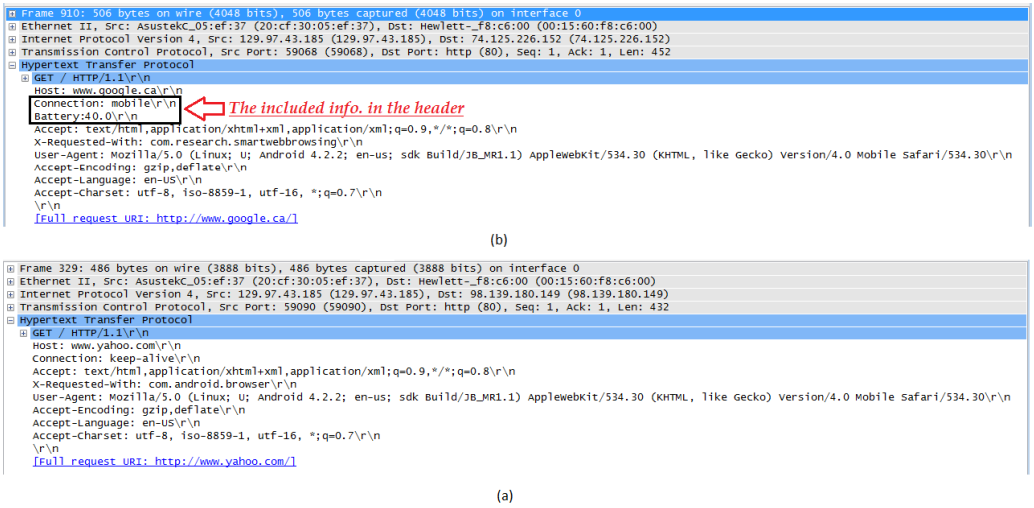

FIGURE 10. HTTP header: (a) standard header; (b) modified header.

TABLE 6. The server's applied policies.

Battery Level (B)	Wi-Fi	Cellular
(B) $\geq$ 70%	Full version of webpages	Webpages with only textual ads
40% $\geq$ (b) $<$ 70%	Webpages with only textual ads	Ads-free webpage
(b) $<$ 40%	Ads-free webpage	Ads-free webpage

75% and the *Connection_Type* is cellular, the server sends an webpage with only textual ads. On the other hand, if the phone *Battery_Level* is 85% and the *Connection_Type* is Wi-Fi, the server sends back the full version of the website, (including the ads), and so on. That is how the server basically works.

2) SYSTEM IMPLEMENTATION

We implemented a prototype system in which the client side was implemented on the *Android* operating system and the server side was implemented in Java.

3) CLIENT SIDE CONFIGURATION

The adaptations that need to be made to the client's browser are described next:

- *Battery Level Listener:* This listener's function is to obtain the smartphone's battery level every time a user makes a web request. The battery level is then passed on to the HTTP Header Handler. Modern operating systems of smartphones gather this information from the battery of the device via the communication pin, as we explained in Section III-A.3.
- *Connection Type Listener:* This listener's function is to identify the type of network that the smartphone is connected to and pass it on to the HTTP Header Handler. This information is gathered by the operating system from the device's protocol stack that is being used.
- *HTTP Header Handler:* This handler modifies the HTTP request's header. It inserts the battery level and the type of network to the optional fields of the HTTP protocol header. Figure 10(b) shows how the header looks after the new fields are added to the standard HTTP header (Fig. 10(a).) These snapshots are for our system, taken from Wireshark to illustrate the modifications that the system makes to the HTTP packets.

4) 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: (i) a HTTP Header Parser that extracts the information inserted by the browser in the HTTP request; (ii) Decision and Response Preparation unit, where the server applies the specified policies. Accordingly, a desired webpage is prepared and sent back to the client based on its current available resources.

B. VALIDATION AND RESULTS

1) SYSTEM PARAMETERS

The following system parameters are associated with SMoW browsing technique:

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: (i) the cellular network is a metered plan, based on usage; and that (ii) the Wi-Fi network is completely free.

These two parameters are used to evaluate the effectiveness of our system in Section.IV-B.

In this section, we validate the SMoW browsing technique by means of measurements. We compare its performance with the conventional browsing techniques and measure the energy and bandwidth costs under three different scenarios.

All the energy and bandwidth measurements were conducted using the test benches described in Section III. Our

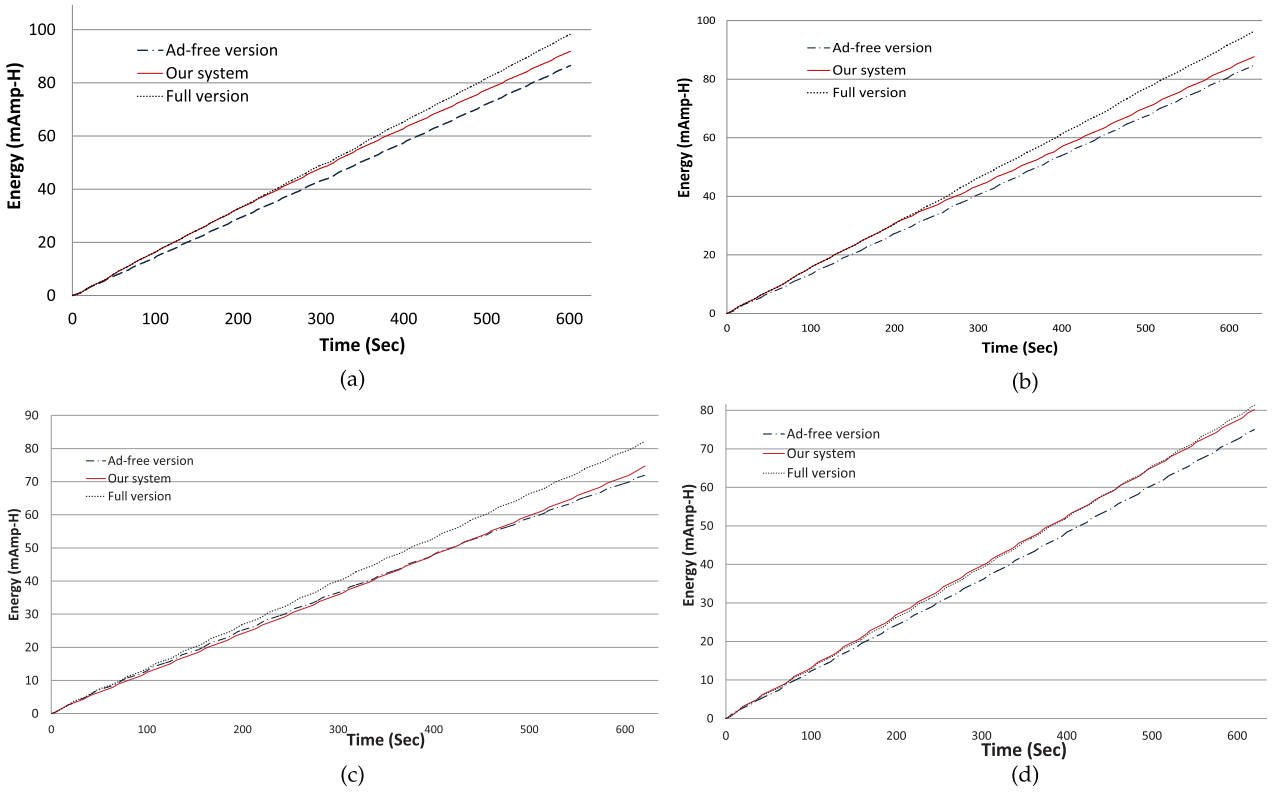

FIGURE 11. Comparison of the energy consumption over a Wi-Fi connection. (a) www.macleans.com. (b) www.therecord.com. (c) www.thestar.com. (d) www.nytimes.com.

SMoW browser was installed on a *Galaxy Nexus* smartphone running *Android V4.2.1*. The results are presented and discussed in the following subsections.

2) ENERGY AND BANDWIDTH RESULTS

Out of the three, 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 made every 30 seconds, for a total of 20 page requests. We set our server to deliver three kinds of web contents: (i) the full version of the website; (ii) a webpage with only textual ads; and (iii) a webpage with no ads at all. In this scenario, the server sends back the full version for the first three minutes, then sends back the webpages with only textual ads for the next 3 minutes, and finally, it sends back the ads-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 websites again. We started by browsing the website's full version and repeated 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 ads-free webpages, and repeated the same browsing experience. Figure 11 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 SMoW. These numbers are considered significant given the fact that battery capacity increases by only 5% every year [27].

Our second scenario involved repeating the first experiment over a 3G network. Hence, the webpage adaptation decisions were made based only on the current *Battery_Level*. Figure 12 shows the results obtained for the same four test cases.

The third and final scenario considers both *Battery_Level* and *Connection_Type*. This scenario shows the overall effectiveness of our system. Figure 13 shows the overall performance of the system. In this scenario, Case # 1 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 10 minutes over a 3G network. Case # 2 was browsing using SMoW, also 20 minutes in total. Case # 3 used only normal browsing for 20 minutes over a 3G network. We found that SMoW outperforms in this test case. Then we used the energy results in Fig. 13 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. 13, we present in Table 7 a summary of the estimated battery life under the three different browsing cases: (i) Case # 1, normal

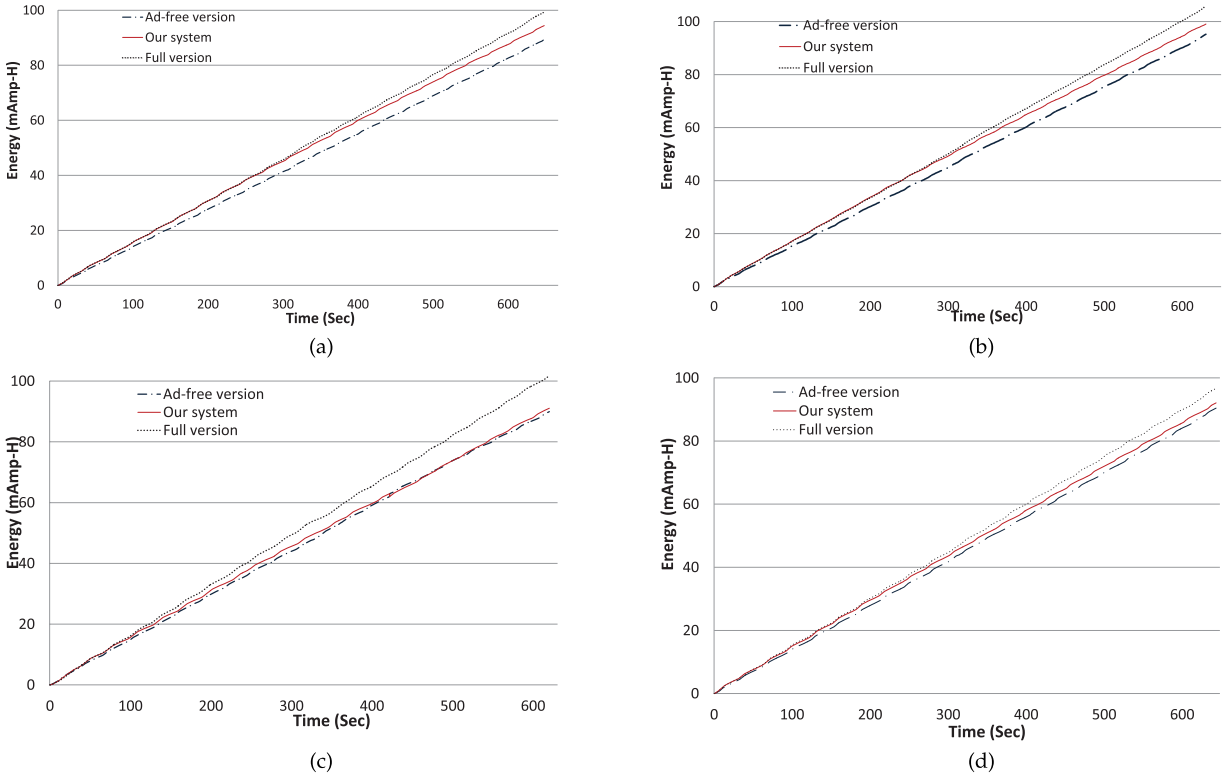

FIGURE 12. Comparison of the energy consumption over a 3G connection. (a) www.macleans.com. (b) www.therecord.com. (c) www.thestar.com. (d) www.nytimes.com.

TABLE 7. Summary of the estimated battery life under three different cases.

	Battery Life Estimation in Hours				
Test Cases	Case # 1	Case #2: SMoW	Case # 3	Δ (SMoW - Case # 3)	Δ (SMoW - 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

browsing, that is, browsing is done half of the time over a 3G network and in the other half it is done over a Wi-Fi network; (ii) Case # 2, browsing using SMoW for the whole time; and (iii) Case # 3, normal browsing over 3G network only. Applying those scenarios, our results show that SMoW can extend battery life up to an hour, compared to Case # 3.

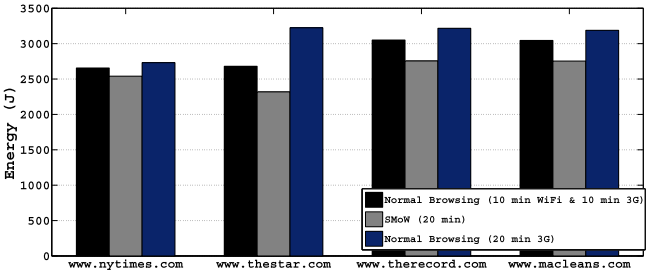

FIGURE 13. Energy consumption comparison between SMoW and normal browsing.

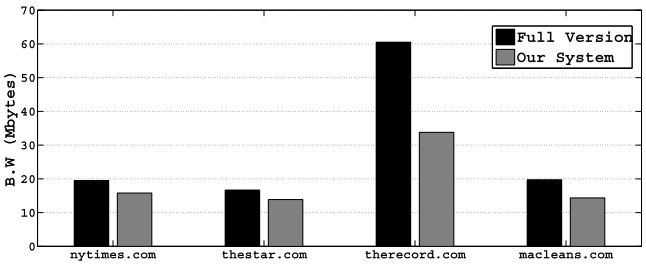

FIGURE 14. Bandwidth measurements.

The bandwidth required to download webpages was measured in the test cases that required a Wi-Fi connection. 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 14 shows the results obtained from running our system for ten minutes. In this period of time, a total of 20 webpage requests were made.

The results confirm that SMoW can save wireless bandwidth up to ∼44% compared to normal web browsing. Regarding the bandwidth consumed when a 3G network is used, it will be similar to the case of Wi-Fi network since the contents to be downloaded are the same.

We made empirical calculations to estimate how much money a user saves using SMoW. According to [28], users spend approximately 53 minutes a day on mobile browsing. If we assume that she spends a minute on each webpage, the total number of requested webpages would be 53. Based on our results in Fig. 14, ∼ 7 to 70 MBytes per day can be saved using SMoW under this scenario. That number accumulates to ∼ 210 to 2100 Mbytes per month. Based on some metered data plans in US (usage-based pricing plans) reported in [25], ∼ $ 11.2 to $ 336 would be saved using our system.

V. CONCLUSIONS

In this work, we focused on the energy and bandwidth impact of the appearance of ads in webpages and made many interesting observations. We investigated the energy consumption due to ads in four well-known websites, and found that ads can consume ∼3.5 to 12 Joules over Wi-Fi and 3G networks, that is ∼6–18% of the total energy of web browsing. Then, we proposed a framework for mobile browsing that adapts webpages delivered to a smartphone, based on the smartphone's current battery level and the network type. Adaptation of the web content comes in the form of controlling the amount of ads to be displayed on the webpage. 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 was explained in detail. By means of experimental validation, we showed that our system can extend smartphone battery life by up to ∼25% and save wireless bandwidth up to ∼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 research, 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 current resources of the smartphone.

REFERENCES

[1] (2013). *The World in 2013: ICT Facts and Figures*. [Online]. Available: http://www.itu.int/ITU-D/ict/facts/material/ICTFactsFigures2013.pdf
[2] comScore. (2012). *comScore Reports September 2012 U.S. Mobile Subscriber Market Share*. [Online]. Available: http://www.comscore.com/Insights/Press_Releases/2012/11/comScore_Reports_September_2012_U.S._Mobile_Subscriber_Market_Share/
[3] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proc. 10th Annu. Conf. Internet Meas.*, 2010, pp. 281–287.
[4] V. Moonsamy, M. Alazab, and L. Batten, "Towards an understanding of the impact of advertising on data leaks," *Int. J. Secur. Netw.*, vol. 7, no. 3, pp. 181–193, 2012.
[5] J. Y. Chung, Y. Choi, B. Park, and J. W.-K. Hong, "Measurement analysis of mobile traffic in enterprise networks," in *Proc. 13th Asia-Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2011, pp. 1–4.
[6] S.-W. Lee, J.-S. Park, H.-S. Lee, and M.-S. Kim, "A study on smartphone traffic analysis," in *Proc. 13th Asia-Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2011, pp. 1–7.
[7] G. Maier, F. Schneider, and A. Feldmann, "A first look at mobile handheld device traffic," in *Passive and Active Measurement*. Berlin, Germany: Springer-Verlag, 2010, pp. 161–170.
[8] 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 *Proc. Eur. Wireless Conf. (EW)*, May 2009, pp. 255–259.
[9] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, " AppScope: Application energy metering framework for Android smartphone using kernel activity monitoring," Tech. Rep., 2012.
[10] A. Mitchell. (2012). *The Explosion in Mobile Audiences and a Close Look at What It Means for News*. [Online]. Available: http://www.journalism.org/analysis_report/future_mobile_news
[11] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones?" in *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, 2011, pp. 91–96.
[12] M. Butkiewicz, H. V. Madhyastha, and V. Sekar, "Understanding website complexity: Measurements, metrics, and implications," in *Proc. ACM SIGCOMM Conf. Internet Meas. Conf.*, 2011, pp. 313–328.
[13] Y. Zhu, A. Srikanth, J. Leng, and V. Reddi, "Exploiting webpage characteristics for energy-efficient mobile web browsing," *Comput. Archit. Lett.*, vol. 13, no. 1, pp. 33–36, Jan. 2014.
[14] S. Chava, R. Ennaji, J. Chen, and L. Subramanian, "Cost-aware mobile web browsing," *IEEE Pervasive Comput.*, vol. 11, no. 3, pp. 34–42, Mar. 2012.
[15] A. Dubey, P. De, K. Dey, S. Mittal, V. Agarwal, M. Chetlur, and S. Mukherjea, "Co-operative content adaptation framework: Satisfying consumer and content creator in resource constrained browsing," in *Proc. 22nd Int. Conf. World Wide Web Companion*, 2013, pp. 221–222.
[16] B. Zhao, Q. Zheng, and G. Cao, "Energy-aware web browsing in 3G based smartphones," in *Proc. IEEE 33rd Int. Conf. Distrib. Comput. Syst. (ICDCS)*, vol. 1. 2013, pp. 1–6.
[17] C.-C. Huang, J.-L. Huang, C.-L. Tsai, G.-Z. Wu, C.-M. Chen, and W.-C. Lee, "Energy-efficient and cost-effective web API invocations with transfer size reduction for mobile mashup applications," *Wireless Netw.*, vol. 20, nos. 1022–0038, pp. 1–18, 2014.
[18] M. W. Kim, D. G. Yun, J. M. Lee, and S. G. Choi, "Battery life time extension method using selective data reception on smartphone," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Feb. 2012, pp. 468–471.
[19] Privoxy. [Online]. Available: http://www.privoxy.org/, accessed Jun. 23, 2013.
[20] Monsoon Solutions Inc. *Power Monitor*. [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/, accessed Jan. 13, 2013.
[21] (2009). *Bell HSPA/UMTS*. [Online]. Available: http://mobilesyrup.com/2010/12/12/review-bell-novatel-wireless-u547-turbo-stick-42-mbps/
[22] A. Abogharaf, R. Palit, K. Naik, and A. Singh, "A methodology for energy performance testing of smartphone applications," in *Proc. 7th Int. Workshop Autom. Softw. Test (AST)*, Jun. 2012, pp. 110–116.
[23] Alexa. (2013). *Website Ranking by Alexa*. [Online]. Available: http://www.alexa.com/
[24] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3G networks," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 137–150.
[25] J. van den Brande and A. Pras, "The costs of web advertisements while mobile browsing," in *Information and Communication Technologies*. Berlin, Germany: Springer-Verlag, 2012.
[26] A. Albasir, K. Naik, and T. Abdunabi, "Smart mobile web browsing," in *Proc. Int. Joint Conf. Awareness Sci. Technol. Ubi-Media Comput. (iCAST-UMEDIA)*, Nov. 2013, pp. 671–679.
[27] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, Berkeley, CA, USA, 2010, p. 4.
[28] M. Dong and L. Zhong, "Chameleon: A color-adaptive web browser for mobile OLED displays," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 724–738, May 2012.

ABDURHMAN ALI ALBASIR received the M.Sc. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2013, and the B.Sc. degree in electrical and electronic engineering from University of Tripoli, Tripoli, Libya, in 2007. He has three years of experience in spectrum management and monitoring with the Ministry of Telecommunications (Regulator of Communication Services). His research activity includes wireless networks, mobile computing, energy efficiency of smartphones, and energy performance testing of mobile applications.

KSHIRASAGAR NAIK received the B.Sc. (Eng.) degree from Sambalpur University, Sambalpur, India, and the M.Tech. degree from IIT Kharagpur, Kharagpur, India. He was a Software Developer with Wipro Technologies, Bangalore, India, for three years. He received the M.Math. degree in computer science from the University of Waterloo, Waterloo, ON, Canada, and the Ph.D. degree in electrical and computer engineering from Concordia University, Montreal, QC, Canada. He was a faculty member with the University of Aizu, Aizuwakamatsu, Japan. He is currently a Full Professor with the Department of Electrical and Computer Engineering, University of Waterloo. He was a Co-Guest Editor of three special issues of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. He was an Associate Editor of the *Journal of Peer-to-Peer Networking and Applications* from 2008 to 2013. He is serving as an Associate Editor of the *International Journal of Parallel, Emergent and Distributed Systems* and the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. In addition, he is serving as a Regional Editor (America) of *Journal of Circuits, Systems, and Computers*. His research interests include dependable wireless communication, resource allocation in wireless networks, mobile computing, peer-to-peer communication, vehicular networks, energy efficiency of smartphones and tablet computers, energy performance testing of mobile applications, and communication protocols for smart power grids. His book entitled *Software Testing and Quality Assurance: Theory and Practice* (Wiley, 2008) has been adopted as a text in many universities around the world. His second book entitled *Software Evolution and Maintenance* (Wiley, 2014) is in press.