2015 IEEE/ACM 37th IEEE International Conference on Software Engineering

Truth in Advertising: The Hidden Cost of Mobile Ads for Software Developers

Jiaping Gui*, Stuart Mcilroy†, Meiyappan Nagappan‡ and William G. J. Halfond*
*University of Southern California, Los Angeles, CA, USA
Email: {jgui, halfond}@usc.edu
†Queen's University, Kingston, Canada
Email: mcilroy@cs.queensu.ca
‡Rochester Institute of Technology, Rochester, NY, USA
Email: mei@se.rit.edu

***Abstract*—The "free app" distribution model has been extremely popular with end users and developers. Developers use mobile ads to generate revenue and cover the cost of developing these free apps. Although the apps are ostensibly free, they in fact do come with hidden costs. Our study of 21 real world Android apps shows that the use of ads leads to mobile apps that consume significantly more network data, have increased energy consumption, and require repeated changes to ad related code. We also found that complaints about these hidden costs are significant and can impact the ratings given to an app. Our results provide actionable information and guidance to software developers in weighing the tradeoffs of incorporating ads into their mobile apps.**

***Index Terms*—Mobile advertisements, mobile devices**

I. Introduction

Mobile advertising has become an important part of many software developers' marketing and advertising strategy [1]. This development has come about in just a matter of a few years. In 2010, the mobile advertising industry's revenue was just over half a billion dollars [2], but by 2013 it reached over 17 billion dollars [3], and in the first quarter of 2014, had already reached over 11 billion dollars [4]. By 2017, analysts predict that revenue from mobile advertising will exceed that of TV advertisements [5] and account for one in three dollars spent on advertising [6].

The presence of mobile ads has become pervasive in the app ecosystem with, on average, over half of all apps containing ads [7]. This has been driven by the development of large scale advertising networks, such as Google Mobile Ads and Apple iAD, that facilitate the interaction between developers and advertisers. To earn ad revenue, developers display ads in their apps by making calls to APIs provided by an advertising network. When the ads are displayed on an end user's device, the developer receives a small payment. A typical business model for a developer is to place ads in their apps and then release the app for free with the hope that the ad revenue will offset the cost of the app's development. In general, this model is perceived as a win-win situation for both developers and end users: developers receive a steady, and sometimes large, ad-driven revenue stream, and end users receive a "free" app.

A key problem in this model is that it depends on the perception that, aside from app development, there are no additional costs to either the end user or software developer. While this is true for direct costs, this fails to account for the indirect hidden costs of embedding mobile ads in an app. On the end users' side, indirect hidden costs come in several forms: loading ads from a remote server requires network usage, for which many users are billed by the amount of bytes; loading and rendering ads requires CPU time and memory, which can slow down the performance of an app; and finally, all of these activities require battery power, which is a limited resource on a mobile device. Developers have hidden costs as well. It is necessary to maintain the code that interacts with the advertisements, which requires developer effort. The ratings and reviews a developer receives can also be affected. Studies have shown that over 70% of users find in-app ads "annoying" [8] and such users may give an app a lower rating or write negative reviews. This negative response may then affect the number of downloads of an app, which in turn can affect the developer's future ad revenue.

In this paper we present the results of our investigation into the hidden costs of mobile advertising for software developers. To carry out this investigation, we performed an extensive empirical analysis of 21 real world apps from the Google Play app store that make use of mobile advertising. Our analysis considered five types of hidden costs: app performance, energy consumption, network usage, maintenance effort for ad-related code, and app reviews. The results of our investigation show that there is, in fact, a high hidden cost of ads in mobile apps. Our results show that apps with ads consume, on average: 48% more CPU time, 16% more energy, and 79% more network data. We also found that developers, on average, make ad related changes in 23% of their releases. The presence of mobile ads also has a rating and review cost, as we found that complaints related to ads and these hidden costs were relatively frequent and had a measurable impact on an app's rating. Overall, we believe that these findings are significant and will help to inform software developers so they can better weigh the tradeoffs of incorporating ads into their mobile apps, understand the impact ads have on their end users, and improve end users' app experience.

978-1-4799-1934-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ICSE.2015.32

II. Motivation

Ads occupy a unique position in the mobile app ecosystem. Strictly speaking, they are not required for the correct functioning of an app. Yet they are essential for monetizing the app and ensuring that developers can profit from their work. When considering their profit, developers typically assume that the only cost they have associated with the app is development and normal maintenance. Conversely, end users typically assume that their only cost comes in the form of viewing ads and, perhaps, paying an upgrade fee to get an ad-free version. At some level, these are reasonable assumptions, since these costs, or lack thereof, are clearly visible to both parties. As we argue in Section I, there are, in fact, other costs. We refer to these as *hidden costs* because for both parties they can go unnoticed, and even if recognized as costs, can be difficult to quantify without additional infrastructure and analysis. In this paper, we perform a systematic investigation to quantify five such hidden costs. Three of these directly affect end users' mobile devices: network usage, energy consumption, and app runtime performance. Two of them more directly affect developers: ad-related maintenance and user ratings. We chose to investigate these specific hidden costs because they represent categories for which we have identified a process for measuring their costs and also quantifiable ways of showing their impact on both developers and end users. Below we formally present our research questions (RQs) with respect to these hidden costs and motivate their inclusion in our study.

RQ 1: What is the performance cost of ads?

Mobile apps display ads by using ad network provided APIs. As with all other invocations, executing these methods requires the device to commit processing power (e.g., CPU) to carry out the ad related functionality. The consumption of this processing power by the ad libraries represents processing power that could have been available to the app to improve its own performance. Runtime performance is important to end users because it influences how "fast" or "responsive" they perceive an app's implementation to be. In this research question, we focus on the runtime performance of an app and how this is affected by the additional processing necessary to carry out ad related functionality.

RQ 2: What is the energy cost of ads?

Mobile devices are energy constrained devices because they run on battery power, which is limited. Therefore, energy efficiency is an important concern for apps that run on mobile devices. Components, such as display and network, are two of the most energy consuming components on a mobile device [9], [10]. These two components also serve an important role in the mobile ad ecosystem since they are used to retrieve and show ads. In this research question we quantify the energy impact of ads in mobile apps. This energy cost is hidden to users because, although they are aware of battery limitations, they do not have any way to isolate and evaluate the energy cost of the ad functionality which is embedded in the mobile app. A high energy cost is impactful because running out of battery power renders a device unusable or requires extra recharging cycles.

RQ 3: What is the network cost of ads?

Network access plays an essential role in the ad network infrastructure. Developers insert invocations to ad network APIs that then send requests to ad servers for an ad to display. In turn, the ad servers transmit advertising content back to the mobile apps to be displayed. All of these require network usage by the mobile device, even if the app containing the ad does not require network access itself. In many cases, network usage has a cost for end users who must pay for Internet access or pay data charges for data access over a certain preset limit. Although there is a direct cost associated with network usage, end users lack visibility into how network is consumed. At best, they may use tools, such as Shark or Root [11], to monitor their apps' network usage, but do not have any mechanism to distinguish how much of this usage is related to ads. Therefore, this remains a hidden cost to them.

RQ 4: What is the rate of app updates related to ads?

Part of the development cost of an app is maintenance. This includes responding to bug reports, adding new features, and evolving the app due to changes in the underlying OS and platform. Prior work has shown that app developers frequently add, remove, or update ad related code in an app [7], [12]. This finding suggests that there may be a high maintenance cost associated with the use of ad libraries. This motivates further investigation to determine how much maintenance effort is caused by the use of ad libraries. In this research question, we examine ad-related code in the apps and track its evolution over different app versions in order to isolate the ad related maintenance effort.

RQ 5: What is the impact of ads on an app's ratings?

The Google Play app store allows users to write reviews and provide a rating (between one and five stars) for the apps that they have downloaded. Good app ratings and reviews are essential for the success of an app in the marketplace. Prior research has shown that app ratings are highly correlated with app downloads [13]. Prior work has also shown that surveyed end users generally have unfavorable perceptions of ads in mobile apps [14], [15], [16]. Therefore, it is possible that these unfavorable reactions carry over and influence end users to give poor reviews for the app. In this research question, we examine end user reviews and determine what the possible impact of mobile ads is on the rating of an app.

III. Case Study Design

The goal of our case study is to investigate the hidden cost of mobile advertisements to end users and software developers. To carry out this investigation we designed a case study to capture and analyze ad-related information and various other types of runtime metrics. In this section we explain how we selected the apps for the study, the process for identifying and instrumenting ad behavior, the creation of repeatable and automatically replayable workloads, and the monitoring and analysis framework. We explain each of these aspects of the case study design in more detail below.

A. Selection of Subject Applications

For our case study, we had five criteria for selecting the set of subject applications. These were: (1) successful apps — indicating that the developers had found a balance of functionality and ad usage; (2) representative of different categories of apps — to enable our results to generalize to a broader pool of apps; (3) actively maintained with frequent releases — so we could examine maintenance costs over time; (4) use of mobile ads; and (5) convertible to and from Java bytecode using the *dex2jar* [17] tool — since we need to perform bytecode manipulation of the apps' classes to facilitate the monitoring and analysis.

To obtain apps that met the first two criteria, we took the top 400 apps in each of the 30 categories of Google Play as ranked by Distimo [18], an app analysis company that ranks apps based on user ratings, number of downloads, and other measures of success. Not all categories had 400 apps in the list of top apps. Therefore in the end we had a list of 10,750 apps from all 30 categories of Google Play. We crawled the Google Play app store everyday using a farm of systems for eight months (from Jan 2014 to Aug 2014), to download every new release of the app and its associated meta-data, such as average user-rating, number of users rating the app, and user-reviews (500 at a time), among other things. To satisfy the third criteria, we sorted the 10,750 apps by the number of releases that each app had in the time frame of data collection (from Jan 2014 to Aug 2014). Then, we selected the top 21 apps from this list, which represented 14 different categories of apps (e.g., travel, media, etc.). To satisfy the fourth criteria, we identified the apps in the corpus that made use of the Google Mobile Ads (GMA) network. We identified an app as making use of the GMA if it contained invocations of APIs provided by the GMA and had visible ads displayed in some part of the user interface. We focused our investigation on only one ad network to control for variability of costs between ad networks and chose GMA in particular because it is the most popular and widely used, representing over 46% of the mobile ad business for the first quarter of 2014 [3]. Finally, we converted each app to Java bytecode, using *dex2jar*, repackaged it using the Android SDK tools, and then manually verified that it executed without failure to ensure it met the fifth criteria. Descriptive information about each of these apps is shown in Table I. In this table we list the app's name, provide it with a unique ID that we use to identify it in the evaluation graphs, its package name, physical size of the app's APK file, and the category assigned to it by the Google Play app store. We also include the number of versions, the number of reviews, and the average rating of each app for the time period between January 2014 and August 2014.

B. Instrumentation of the Subject Applications

To address the research questions outlined in Section IV requires that we have two versions of each app, one with ads and the other without. To create the no-ads version of an app, we used instrumentation based techniques to remove all invocations of APIs defined by the ad network. Note that some prior approaches have simply replaced the ad library with "dummy" implementations [19]; however we chose to completely remove the invocations since there is a non-zero time and energy cost associated with even an invocation of an empty method [20], [10]. To perform the instrumentation, we first converted each app's APK into the corresponding Java bytecode using *dex2jar*. Then we used the ASM library [21] to analyze the bytecode of each class of each app and identify ad-related invocations. These invocations could be identified by matching the package name (e.g., "ads", "mobileads", and "mobads") of the invocation's target method with that of known ad networks. The package names of ad networks can be found by examining their API documentation. For each ad-related API invocation identified, we wrote instrumentation code to remove the invocation and, where possible, any other support objects created as arguments to the invocation. In some cases, it was not possible to remove all references to the ad library. Namely, if an invocation unrelated to ads had an ad-related argument, then we could not remove the initialization of that argument. In our subject apps there were 141 such problematic invocations, out of a total of 716 ad-related invocations. After instrumentation, we repackaged the app and then verified the removal of the ads with two checks. First, we manually executed the apps and verified that there were no visible ads. Second, we used *tcpdump* on the smartphone's local network to see if there were any signs of an ad API accessing the network. To create the version of the app with ads, we decompiled, then repackaged each app, without removing the ads. We did this to control for any bytecode level transformations introduced by *dex2jar*, *asm*, or *dx*, which would have also occurred to the no-ads version.

C. Generation of Subjects' Workloads

For each app, we created workloads to execute the app and exercise its functionality. The goals for each workload we created were: (1) complete as possible with respect to the app's primary functions; (2) repeatable across multiple executions of the app; and (3) long enough to ensure several ad reload cycles.

To generate workloads, we leveraged the RERAN tool [22]. This tool records a user's interaction with an app as a series of events and can then replay these events at a later time. To generate an initial workload, we interacted with each app and tried to invoke as much functionality as possible. For example, we clicked the different buttons or labels on a screen, stayed for some time on a new page, returned or went to another new page, and entered values for text and search boxes. Although these workloads may not be representative of realistic usage, they provide us with a more or less complete coverage of the apps' key functions. On average, we interacted with each app for 1.5 – 4 minutes. This amount of time was chosen because GMA can be set to refresh every 30 – 120 seconds and with this interaction length we would ensure several ad reloads. After creating an initial workload, we repeated the execution of the workload several times and manually verified that the execution of the app was deterministic with respect to

TABLE I
SUBJECT APPLICATIONS

ID	App Name	Package Name	Category	Size (MB)	# Versions	# Reviews	Avg. Rating
M1	Restaurant Finder	com.akasoft.topplaces	travel & local	3.7	24	464	4.35366
M2	Smileys for Chat (memes,emoji)	com.androidsx.smileys	communication	15.9	16	613	4.32011
M3	Arcus Weather	com.arcusweather.darksky	weather	2.8	30	513	4.32317
M4	Polaris Navigation GPS	com.discipleskies.android.polarisnavigation	travel & local	7.8	29	960	4.41557
M5	3D Sense Clock & Weather	com.droid27.d3senseclockweather	travel & local	10.7	20	399	4.42509
M6	Drudge Report	com.iavian.dreport	news & magazines	1.5	20	1317	4.24225
M7	Podcast Republic	com.itunestoppodcastplayer.app	news & magazines	3.6	39	1723	4.58928
M8	Followers For Instagram	com.noapostroph3s.followers.instagram	social	2.4	17	1337	3.75924
M9	Public Radio & Podcast	com.nprpodcastplayer.app	news & magazines	3.2	21	671	4.2379
M10	English for kids learning free	com.oman.english4spanishkids	education	8.0	21	90	4.13483
M11	Lomo Camera	com.onemanwithcameralomo	photography	29.5	20	942	4.33325
M12	Smart Booster - Free Cleaner	com.rootuninstaller.rambooster	tools	3.5	22	1258	4.50653
M13	Pixer	com.sixtyphotos.app	social	4.4	27	599	4.36689
M14	The Best Life Quotes	com.socialping.lifequotes	entertainment	2.6	31	784	4.42554
M15	SofaScore LiveScore	com.sofascore.android	sports	9.9	39	1158	4.72082
M16	Player dreams	com.team48dreams.player	music & audio	1.9	40	693	4.40827
M17	VLC Direct Streaming Pro Free	com.vlcforandroid.vlcdirectprofree	media & video	2.3	25	868	4.28025
M18	Translator Speak & Translate	com.voicetranslator.SpeakAndTranslateFree	travel & local	5.3	22	1024	4.38482
M19	7Zipper	org.joa.zipperplus7	communication	7.6	19	699	4.66026
M20	Guess The Song	quess.song.music.pop.quiz	trivia	19.5	40	3426	4.62825
M21	Radaee PDF Reader	radaee.pdf	productivity	4.6	21	587	4.41044

the sequence of actions and identified any system state (e.g., resetting system settings and killing service related processes) that needed to be restored prior to the replay of the interaction. In many cases, the execution of the no-ad version would require a systematic shift of the X and Y coordinates of certain user events (e.g., a touch), due to the absence of a displayed ad, and we corrected the RERAN traces at this time.

D. Monitoring and Analysis of Subject Applications

To collect runtime data on the hidden costs, we ran both versions of each app (with ads and no-ads) while monitoring its execution. The mobile device we used was the Samsung Galaxy SII smartphone with a rooted Android 4.3 operating system. For each version, we first restored the system environment to its original state. Then we loaded the app on the mobile device and started its execution. Before beginning the replay, we allowed the system to sleep for 15 seconds to ensure that the initial page had completely loaded and displayed. Then we began the RERAN replay. During the replay execution of the app, we recorded statistics about the execution. This process was repeated four times for each experiment to minimize the impact of background noise, and in each iteration, the order of the apps and both versions was changed. The specific statistics and measurements taken during the execution varied according to the addressed research question. We elaborate on the measurements and metrics for each research question in Section IV.

IV. RESULTS AND DISCUSSION

In this section we discuss the details of the experiments we carried out to address each of the RQs defined in Section II. For each RQ, we describe the approach we employed to capture the relevant metrics and measurements, present the results we obtained, and discuss the implications of these results with respect to each of the RQs. Essentially, each of the subsections in this section describes the monitoring and analysis portion of our case study (Section III-D) as it was customized to address the RQs.

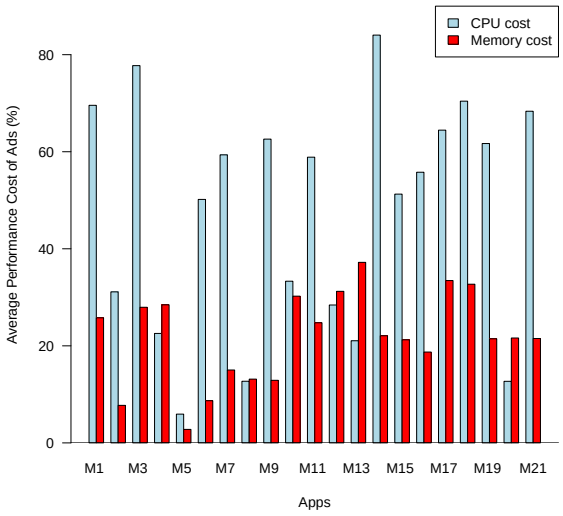

Fig. 1. Relative performance cost of the with-ads version over the no-ads version.

A. RQ 1: What is the performance cost of ads?

Approach: To determine the performance cost of mobile ads, we measured two performance metrics, CPU utilization and memory usage, on both the with-ads and no-ads version of each app. To obtain these metrics, we ran the standard *top* utility on the mobile device while it was executing both app versions. We set the *top* tool to record the two performance metrics on a one second interval. Specifically, *top* recorded the CPU percent utilization and the amount of memory in the

Resident Set Size (RSS), which indicates how many physical pages are associated with the process. Since the running app was the only one that had a process foreground visible during the replay, the RSS reflected the physical pages of the app's process. We then calculated the average value of the metric for each app. Note that even though running *top* can affect the mobile device's performance, we verified through experiments that the effect of *top* was consistent across app executions and versions.

Results: The results of these experiments are shown in Figure 1. The dark red bar shows memory usage and the light blue bar shows CPU utilization. Each bar in this graph represents the percent difference between the performance metrics for the with-ads and no-ads versions listed along the X axis. A positive number means that the with-ads version had a higher value for the metric. As the results show, the with-ads version had a higher performance cost for all of the subject apps. The median memory increase was 22% and the median CPU utilization increase was 56%.

Discussion: Overall, the results show a markedly higher resource consumption for apps with mobile ads. We expect that this result is due, in part, to managing the increased network usage that we find is associated with ads in Section IV-C. We also expect that retrieving and updating ads occurs when an app might otherwise be in an idle state waiting for a user event. Mobile apps actually spend a significant amount of their perceived runtime in an idle state [10]. Therefore, even the addition of a small amount of activity, such as managing ad interactions, can lead to a surprisingly large increase in CPU utilization. Case in point, in our experiment the median with-ads and no-ads actual CPU utilization was 20% and 7%, respectively. We hypothesized that the increase in CPU utilization was also likely to indicate that end users would experience a slow down in response time. To evaluate this, we instrumented the Android event handlers and activities for a subset of the subject apps using an instrumentation tool we had developed for Android apps in prior work [23], [20], [24] that is based on the efficient path profiling technique by Ball and Larus [25]. We were unable to instrument all of the apps because the tool used BCEL, which was limited in its ability to process some of the apps' bytecodes. Nonetheless, using this tool we were able to instrument and measure the execution time of eight with-ads and no-ads versions. We found that, on average, the with-ads versions took 7% longer to complete their event handling and activities. This suggests that including mobile ads has a measurable impact on the level of responsiveness of the app.

B. RQ 2: What is the energy cost of ads?

Approach: To determine the amount of energy consumed by mobile ads, we measured the energy consumption of each app's with-ads and no-ads version during the workload replay. We connected the mobile device to a Monsoon Power Monitor (MPM) [26], which sampled the energy consumption of the device at a frequency of 5KHz. Before beginning the replay, we started the MPM and began recording the power measurements. Then we noted the time at which the replay began and ended. The total energy consumption of an app was calculated as the sum of all energy consumed between the beginning and ending timestamp. The difference between the energy consumption of the with-ads and no-ads version represented the energy cost associated with running the mobile ads. Unlike the performance metrics measured in Section IV-A, we included the energy incurred while the app was idle. The reason for this is that display represents a significant portion of an app's energy consumption and visible ads are directly responsible for a portion of that display energy. Note that the MPM is a passive measurement device so it does not affect the energy consumption of the hardware or mobile software.

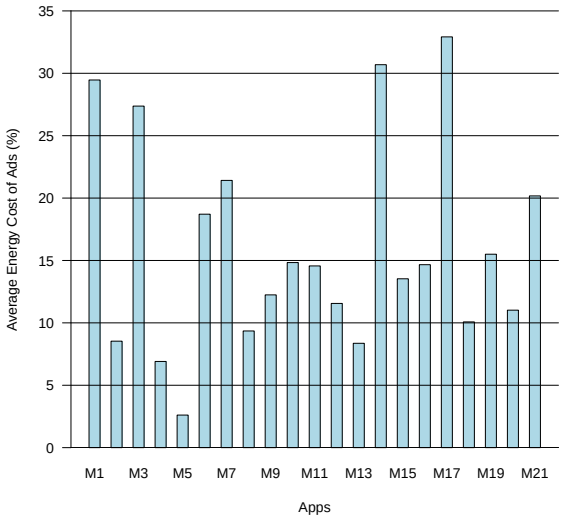

Fig. 2. Energy cost for the with-ads and no-ads versions.

Results: The results of the energy comparison are shown in Figure 2. For each app along the X axis, the chart shows the relative energy consumption increase of the with-ads version over the no-ads version. For all apps, there was always an increase in energy consumption associated with the with-ads version. The energy increase ranged from 3% to 33%, with a median of 15%.

Discussion: For some of the apps, the energy consumption related to ads is quite high. We found that six apps had an increase of over 20% in their energy consumption due to their use of mobile ads. With energy, it is important to note that a high cost does not directly translate into a high financial cost. For example, even a 33% energy increase only represents 50 Joules. The cost to recharge a battery with this amount of energy is negligible. However, a high energy cost can translate into a usability cost for a mobile device. Consider the following illustrative scenario. A typical battery for the

Samsung Galaxy SII smartphone contains 2.5 hours of charge. If the SII was to run only the with-ads and no-adds version of the app with the median energy consumption, then the charge would last 2.1 hours instead of 2.5 hours. With the most expensive ad energy cost, this number decreases to 1.7 hours. This means that an end user would have to recharge their phone 33% more often to compensate for the most expensive ad's energy cost. Overall, this decreased battery lifespan could impact the usability of a mobile device as users would have to charge it more often and have a shorter amount of time in which they could use their phones.

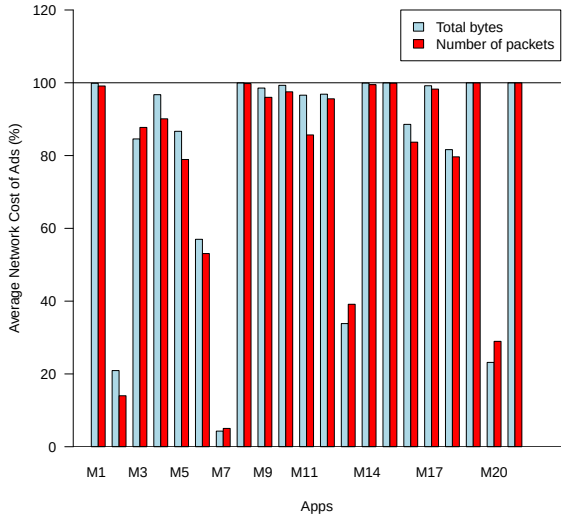

Fig. 3. Network cost metrics.

C. RQ 3: What is the network cost of ads?

Approach: To address this research question, we collected measurements of an app's network usage during the replay of the app's workload. The first of these measurements, data usage, is the total number of bytes sent and received by the app during the replay, and the second measurement, number of packets, is the count of network packets sent and received by the app during the replay. To obtain these measurements, we ran *tcpdump* on the smartphone's network connection to record every packet sent and received on the smartphone during the workload replay. We then analyzed the captured network trace to compute the measurements. This process was repeated for the with-ads and no-ads version of each app. We then calculated the relative difference of both metrics for the two versions of each app.

Results: Figure 3 shows the results of this experiment. The Y axis shows the relative difference for each of the apps listed along the X axis. The light blue bar represents the percent difference in data usage and the dark red bar represents the percent difference in the number of network packets. A positive number indicates that the with-ads version had a higher value for the measurement. The results show that the with-ads version always had a higher data usage and packet count than the no-ads version. For the subject apps, the median increase in data usage over the no-ads version was 97% and for packet usage it was 90%. The results also show that the differences for data usage and package count were generally within a few percentage points of each other for each of the apps.

Discussion: Overall, the results show that there is a very high network cost associated with mobile ads. Moreover, there were several cases in which the percent increase was 100%, which indicated that almost all of the network traffic for the app was due to ads. There were also four apps with relatively low network cost increases. These four were the heaviest network users. For example M7 and M20 played songs during their replay, so the increase due to ads was smaller relative to the overall network usage of the app. We also analyzed the data in more detail to better understand the potential impact of the ad related network traffic. We looked at the impact in terms of potential cost in dollars and energy inefficiencies. For dollar cost, we calculated the median absolute increase in network usage, which was 243,671 bytes and multiplied this by the average cost per MB of a major US based carrier (AT&T), which was $0.07 in 2013 [27]. From this we determined that each execution of the with-ads version could potentially cost end users $0.017 more in terms of network charges. Although this type of cost would only apply in situations where users were paying for metered data; we note that in the case of data overage charges, this would be the amount that could be directly attributed to the ads. With regards to energy inefficiency, mobile devices are energy-inefficient when they send packets that are smaller than the maximum packet size. This is because there is a fixed overhead for sending packet headers, which is amortized over larger packets sizes [28]. So we looked at the average percentage of packets involved in communication with the ad networks that were smaller than the maximum size packet and, therefore, were suboptimal with respect to energy usage. For this analysis, we focused on the 16 apps that had almost all (over 80%) of their network traffic due to advertisements, so we could more accurately characterize only ad-related traffic. We also excluded all TCP control-related packets (e.g., SYN and ACK packets). We calculated this number for both the with-ads and no-ads version, and then subtracted the no-ads count from the with-ads count to isolate the number of suboptimal sized packets that were due just to ad traffic. The results of this analysis indicate that for these apps, over 10% of the advertising traffic was within this size range and, therefore, making sub-optimal use, energy-wise, of the network resources. Taken together, the results of these additional analyses show that the increased network usage of embedded ads can also result in real dollar costs for end users and often represents an energy-inefficient use of network resources.

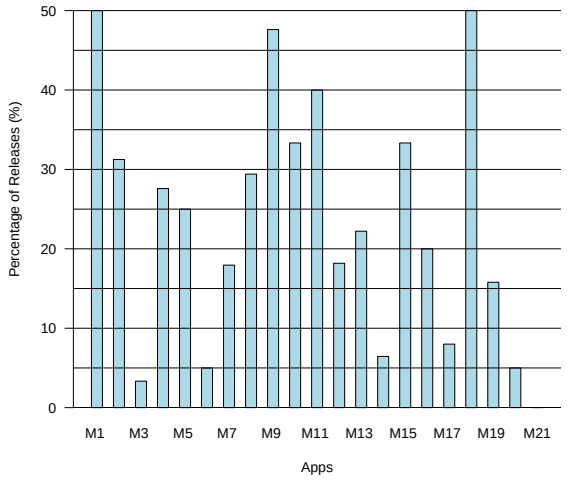

Fig. 4. Percentage of releases that included ad-related maintenance.

D. RQ 4: What is the rate of app updates related to ads?

Approach: Our goal in this research question is to determine the cost, in terms of maintenance effort for the developer, of including ads in the apps. Since, we do not have the development history or the source code repository of the subject apps used in our case study, we approximate maintenance effort as the number of releases of the app in which a developer performed ad library related changes in their apps. Note that this metric does not imply that a release was only related to an ad-related change, only that the release includes ad-related changes. For every release of each app, we decompiled the release and extracted the calls to the GMA network. For each app, we determined if the ad related calls were different from one release ($Release_i$) to the next ($Release_{i+1}$). We defined different by treating the collection of ad related calls as a multiset of tuples, where each tuple was comprised of the containing method of the call and the target method of the call. If there was a difference in the two multisets, then we concluded that there was an ad related change occurring between those two releases. Note that our definition does not include simple changes, such as a call changing in location within its original method. We performed this analysis for all releases of all apps to determine how many versions of the app had ad related changes.

Results: In Figure 4, we report the ratio of the number of app versions that had ad related changes to the number of app versions that have been released. Overall, the median value for this metric was 22%. This indicates that half of the subject apps had ad related changes in almost one of every four releases. We also found that this metric had a wide range. For example, the app com.akasoft.topplaces and com.voicetranslator.SpeakAndTranslateFree had an ad related change in every other release. While at the other end of the spectrum, radaee.pdf did not have any ad related changes in its 21 releases.

Discussion: Our results found that a considerable portion of releases had ad related changes. This was counter-intuitive as the ad network libraries are generally straightforward to use and stable. So we investigated our results further to try to understand the reason we saw such high numbers. First, we compared the number of ad related changes against the number of updates that had been performed to the GMA network libraries in the same time period. We determined the update number to be five by looking at the release history of the GMA library. Of the subject apps, there were 11 that had either the same number or fewer number of ad related changes, which could offer a possible explanation for their changes. However, there were still 10 apps that had a higher number of ad related changes. By investigating the reviews, we found another possible explanation, that users were reacting negatively to ad related changes in the apps and developers were responding to these complaints by modifying ad related behavior. For example, one of the users of the app com.discipleskies.android.polarisnavigation, wrote a one star review for the app, stating: 'Last update full of annoying ads. Don't update.'. Polaris was also one of the apps above both the median and average number of ad related changes. Similarly, one user of the app com.noapostroph3s.followers.instagram complained (in a one star review) that 'The update didn't fix any bugs it only added ads!!!'. This app also had a higher-than-average percentage of its releases involve ad related changes. We also note findings by Khalid and colleagues [29], reporting that 11% of all user complaints in their subject apps occurred immediately after updates. We hypothesize that app developers may be changing their apps' ad related behavior to possibly increase ad revenue, and then adjusting the ad behavior in response to user complaints. In future work, we plan to investigate this hypothesis using more sophisticated static analysis based code change techniques. Regardless of the reason behind the ad related changes, it is important to note that maintenance is an expensive part of the software lifecycle and code that results in higher-than-expected maintenance efforts can represent a hidden cost to the developer.

E. RQ 5: What is the impact of ads on an app's ratings?

Approach: To address this research question we investigated the impact of ads and hidden costs on the reviews of the app. To gather the review and rating information, we crawled the Google Play app store and collected the reviews for each of the subject apps on each day between January 2014 and August 2014. Since Google Play only allowed us to retrieve 500 user reviews per day, we retrieved up to 500 reviews on the first day and then, on each subsequent day, retrieved all of the latest reviews (up to 500). Thus if an app got fewer than 500 reviews, we were able to retrieve all of the reviews, but if there were more than 500 reviews, then we only got 500 of the most recent. In total, we collected 20,125 reviews for the subject apps. Of these reviews, we only considered the

one and two star reviews, since they have been shown, via sentiment analysis, to reflect user complaints [30]. This gave us 2,964 reviews. We then analyzed the reviews to determine if any of them had keywords related to ads (regex = ad/advert*) or any of the hidden costs defined in RQ1–3 (regex = power/drain/recharg*/battery/batery/network/bandwidth/slow/hang). We chose these particular keyword variations based on our prior experience in manually examining user reviews for different types of user complaints [29].

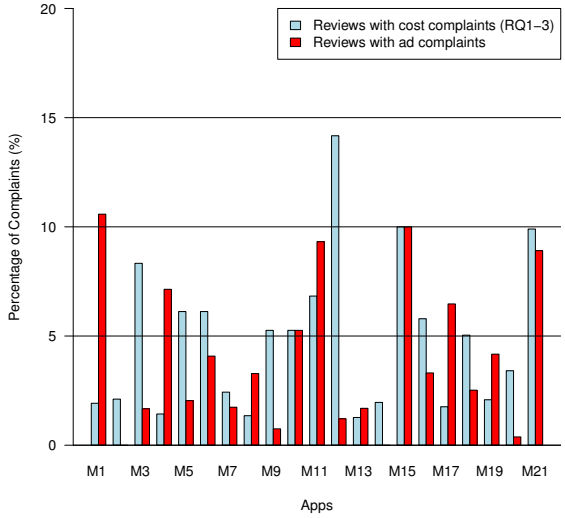

Fig. 5. Complaints about ads and the hidden costs.

Results: In Figure 5 we present the percentage of one and two star reviews where users complain about ads or one of the hidden costs. Only two apps (com.androidsx.smileys and com.socialping.lifequotes) had no ad related complaints and all apps had complaints related to at least one of the hidden costs. Overall, over 50% of the apps had at least 3.28% of their user complaints dealing with ads and 5.04% dealing with hidden costs defined in RQ1–3. These numbers should be considered a lower bound since we only considered complaints that explicitly mentioned one of the keywords and it is possible we did not consider all possible ways to complain about a particular topic.

Discussion: In an absolute sense, the percentage of complaints about either ads or one of the hidden costs may appear small. However, findings by Khalid and colleagues [29] put these numbers into context. In their work they found 12 categories of user complaints by manually analyzing the reviews from 20 iOS apps. They also found that seven of the 12 complaints had an occurrence frequency less than 3.28%. Therefore, we consider the complaint occurrences of ads and hidden costs to be higher than average. One might wonder if the costs are indeed hidden since they have a higher-than-average number of complaints about these topics, but it is important to note that these reviews together comprise only a little more than one percent of all of the reviews and, as such, are not likely to register with the developer. Nonetheless, they do have a measurable, albeit small, impact on the ratings. We recalculated each app's new rating if the reviews complaining about either ads or one of the hidden costs were to be removed. The average increase in rating would be about 0.003 stars. Here again, this is a small number, but it should be noted that a 0.003 change is sufficient to change even the ranking of several of our subject apps if they were ranked by rating.

We also performed a manual investigation of the complaints in order to better understand the nature of the complaints. We found that an overwhelming amount of the reviews were about the interference of the ads with the UI of the app (53% of all the reviews related to ads). Specifically, users complained that

- there were too many ads (e.g., for app com.akasoft.topplaces, where a user says - 'Too much adverts'),
- the ads were too big or took up too much of the screen space (e.g., for app com.akasoft.topplaces, where a user says - 'Annoying full screen ads every time you run the app. Uninstall')
- the ads blocked essential functionality (e.g., for app com.onemanwithcameralomo, where a user says - 'Ads right in the middle of my photos uninstalling')

Therefore, we conclude that, in specific, ad placement within the UI can cause many negative reviews and should be a developer concern when adding ads to their apps.

The next most frequent complaint from the users about ads is having ads in the app even after getting a paid-for version of the app (28% of all reviews related to ads). More specifically, users complained when

- they paid for an app (e.g., for app com.noapostroph3s.followers.instagram)
- they downloaded a paid version for free as part of a promotion (e.g., for app com.onemanwithcameralomo)
- they referred the app to other users (e.g., for com.sofascore.android)

Therefore, we conclude that the presence of ads in paid-for versions of an app is a trigger for complaints. Developers should carefully weigh the benefits of extra ad revenue versus the possibility of upsetting paying customers.

We also noticed several interesting trends for different apps. The users of the apps com.discipleskies.android.polarisnavigation and com.sixtyphotos.app thought that the ads were very intrusive. In this case, the apps displayed ads even after the app was closed. Finally, one user in one app (radaee.pdf) directly complained about the power consumed by the ads. In our results in Section IV-B, we found that this app had an approximate energy cost of 20%. There were five apps with higher energy costs that did not receive any such complaints, suggesting that such costs, although high, are indeed hidden to the end user.

TABLE II
GENERALIZING RQ 1-5 FOR FOUR APPS WITH TWO OTHER MOBILE AD NETWORKS

Metric (RQ)	(21 apps with GMA) Min	Median	Max	(AMA+GMA) G1	G2	(MMA+GMA) G3	G4
CPU (1)	6	56	84	43	43	27	26
Memory (1)	3	22	37	15	25	14	21
Energy (2)	3	15	33	20	15	20	17
Bytes (3)	4	97	100	14	70	62	28
Packets (3)	5	90	100	12	78	61	29
Updates (4)	0	22	50	18	17	24	31
Complaints (5)	0	3.28	11	1.85	0	2.47	7.61

V. GENERALIZABILITY

As described in Section III-A, we chose only apps that used the GMA. While this helped to control for ad network variance, it raises a possible threat to the external validity (generalizability) of the results. In a small study, we evaluated whether the results we described in Section IV held for other ad networks as well. Therefore we chose two other popular ad networks — Amazon Mobile Ads (AMA) and Mopub Mobile Ads (MMA).[1] We then chose two apps for each mobile ad network with the same criterion as our subject apps (i.e., successful apps that are actively maintained). These apps are: G1: com.bambuna.podcastaddict (AMA + GMA), G2: com.x2line.android.babyadopter.lite (AMA + GMA), G3: com.fivemobile.thescore (MMA + GMA) and G4: com.slacker.radio (MMA + GMA). Each of these four apps had their respective new ad networks and the GMA (we could not find apps that had only other ad networks and not GMA). We then repeated our experiments for all five of our research questions on these four apps.

Table II shows the results for these four apps. To place them in context we show the minimum, median and maximum values for the corresponding results of the GMA apps. We find that in some cases the results for the four new apps are similar to the median value, such as the memory cost in RQ1. For other questions, such as the network costs in RQ3, the results for G1 (14% and 12%) and G4 (28% and 29%) are far from the median values of 97% and 90%. However, for all the questions, the results for the four new apps are between the minimum and maximum values. This shows that the apps with other ad networks do not have costs significantly better or worse than apps that have only GMA. Therefore we can be more confident that our results and conclusions may be applicable to other ad networks as well.

VI. THREATS TO VALIDITY

In this section, we discuss the threats to validity of our study and explain the steps we took to mitigate those threats.

External Validity: The results and conclusions of our case study were based on a set of 21 Android apps. Since this set size is significantly smaller than the population of all Android apps, this could impact the generalizability of the conclusions. To mitigate this threat, we ensured that all apps were real-world apps downloaded from the official app store for Android apps, i.e., the Google Play app store. Additionally, these apps represent popular and successful apps, have a high ranking via the Distimo listing, a high number of versions, and represent 14 different categories of apps. Nonetheless, we acknowledge several limitations to the generalizability of our results. First, our work targeted only Android apps, so the results may not generalize to iOS based apps, which also represent a significant portion of the app marketplace. However, we expect that since the underlying mechanisms of ad display are similar, we would see similar results. Second, we biased our app selection for popular and successful apps. It is possible that less successful apps may have a different hidden cost. We hypothesize that since these apps are less successful, it is likely that they have higher hidden costs, so our results would represent a lower bound on the average hidden costs. However, exploring this hypothesis is something we intend to do in future work.

Internal Validity: In our study, we used different tools or commands to measure each metric on the smartphone. We chose standard tools that have been used in previous research studies. To ensure the reliability of our results, we repeated measurements four times. Below we present details about the tools and the steps we took to mitigate specific threats to internal validity.

First, we instrumented Java bytecode which was generated from Dalvik bytecode by reverse engineering. The *dex2jar*, *asm*, and *dx* tools may introduce differences on the instrumented version from the original app. To address this threat, we ran the tools on both versions (with-ads and no-ads) so that any effect would be consistent.

Second, we used RERAN to record and replay workloads in our experiments. However, it is difficult to execute two versions (i.e., with-ads and no-ads) of an app in exactly the same environment. To mitigate the effects of the environment on the final results, we adopted different strategies as we described in Section III. We measured the cost of mobile ads through several groups of runs for each app. The average time difference between two consecutive runs under the same workload was 0.32%, with the highest value being 0.78%, which is negligible compared to the total duration of each run. This indicates that, for the most part, we were able to maintain similar execution environments for the apps' replay.

Construct Validity: The goal of our study is to measure the hidden cost of mobile ads. We investigated this by defining and measuring several metrics for end users and developers. In Section III, we explained why we chose these metrics. However, the relative importance of these metrics may vary in different usage and development contexts, and it is likely that there are other cost metrics, not addressed in our work, that may be important in other settings. Thus, the cost of ads presented in this paper is a lower bound. We hope that this work will encourage further identification and quantification of additional hidden costs.

[1]Mopub can be considered as an ad mediation service [31], but serves as an ad network for the purpose of our study.

Content Validity: Our experiments and conclusions assume that developers would want to minimize costs to end users. An intriguing question and threat to the validity of our conclusions is that developers may disregard these costs in order to drive end users to paid ad-free versions of their apps. Related work has shown that paid versions of apps have significantly lower costs in terms of network usage [32]. However, it is not clear if this is intentional and developers must still strike a balance between ad placement and the quality of the user experience to avoid driving away potential paying customers.

VII. Related Work

In this section we present related work focused on different aspects of mobile advertisements. We discuss four areas of related work and the overlaps and differences of the related work with our work.

Cost of mobile ads: Broad empirical studies [7], [12] find that ad library updates in mobile apps are frequent, and certain specific ad libraries can result in poor app ratings. However, the hidden costs of mobile ads regarding the maintenance of ad libraries and app ratings are not investigated. Wei and colleagues [32] quantify the network usage and system calls related to mobile ads, based on carefully constructed rules, and quantify the difference between free and paid versions. Vallina-Rodriguez and colleagues [31] conduct an in-depth analysis on the characteristics of mobile ads from traffic traces of a major European mobile carrier with over three million subscribers. They use a custom-built app with an ad slot at the bottom of the screen for the evaluation of the energy consumption of three popular ad networks. Nonetheless, only pure ad traffic is evaluated with respect to the energy cost and the impact of background traffic is unknown. Other related works give a high level picture of how users respond to mobile ads, in contrast, our work studies this topic through ad libraries within apps. Different methods or models [14], [15], [16] are proposed to identify factors that influence consumers' response to mobile ads. However, these studies are based on users' feedback in surveys. In contrast to the above studies, which look at one or two types of ad related costs for mobile apps, our work investigates five different costs for 21 apps from the Google Play app store.

Improving mobile ads: Mohan and his colleagues [19] propose an ad prefetching system to reduce ad energy overhead. They present an approach to prefetch mobile ads with a negligible revenue loss. Shekhar and colleagues [33] propose *AdSplit*, which guarantees protection and integrity to advertisers by separating mobile ads from applications. They evaluate the performance of *AdSplit*, but do not focus on the performance cost of mobile ads. Khan and colleagues [34] propose an ad delivery framework, which predicts user context to identify relevant ads. They report a non-negligible amount of network traffic generated by ads. Vallina-Rodriguez and colleagues [31] implement a prototype with prefetching and caching techniques and show an improvement in the energy consumption and network usage of ads. All of these approaches focus narrowly on one or more cost indicators and research on how to improve mobile ads. However, we focus on evaluating the cost of mobile ads with respect to several hidden costs.

Other aspects of mobile ads: *DECAF* [35], *SmartAds* [36], and *PUMA* [37] can detect violations of ad layouts, enable contextual advertising, and detect ad fraud, respectively. Mobile ads are personalized through attributes, trends [38], and behaviour [39], etc. Rasmussen and colleagues [40] analyzed the effects of advertisement blocking methods on energy consumption. Book and colleagues [41] observe a steady increase in the number of permissions that Android ad libraries are able to use, after examining a sample of 114,000 apps.

Cost of apps as a whole: Several studies have looked at how to estimate or measure the energy consumption of mobile applications [20], [10], [23], [28], [42]. Hindle [43] proposes a methodology to relate software changes to power consumption. Pinheiro and colleagues [44] develop systems to address power conservation for clusters of PCs. Qian and colleagues [45] design *ARO*, a tool that exposes the cross-layer interaction among various layers and profiles resource usage for mobile applications. *eProf* [46] uses a power model to report the energy usage of the various hardware components of mobile device. All these studies have looked at mobile apps as a whole or the mobile device. However, in our study we specifically examine the cost of ads in mobile apps.

VIII. Conclusion

Millions of smartphone users install and use thousands of free apps, which are often monetized by the developers via mobile advertisements. In this paper we postulate that the apps are merely free-to-download and in fact have several forms of hidden costs due to the ads. We carry out experiments on 21 highly-rated and actively maintained Android apps to investigate these costs. We find that the cost of ads in terms of performance, energy, and bandwidth are substantial. The median relative CPU, memory, energy, and bandwidth costs of ads are 56%, 22%, 15%, and 97% respectively. We also find that 50% of the apps have ad related changes in one out of almost every four releases, and that 4% of all complaints in the reviews of the apps in the Google Play app store are with respect to ads. Although this is an intuitive observation, such results have not been formally demonstrated and are hard to quantify without extensive measuring infrastructure. We believe that our study provides strong evidence of hidden costs due to ads in apps, and that developers need to optimally use them in the apps. The take-home message of our study is that both the research community and the ad library networks need to take these costs into consideration and to make ads more cost efficient.

Acknowledgments

This work was supported by National Science Foundation grant CCF-1321141.

REFERENCES

[1] O. Consulting. (2013, September) Marketer Perceptions of Mobile Advertising - 2013. Interactive Advertising Bureau.

[2] S. Vranica and C. S. Stewart. (2013, October) Mobile Advertising Begins to Take Off. Wall Street Journal.

[3] (2014, March) Driven by Facebook and Google, Mobile Ad Market Soars 105% in 2013.

[4] PricewaterhouseCoopers. (2014, June) At $11.6 Billion in Q1 2014, Internet Advertising Revenues Hit All-Time First Quarter High. Interactive Advertising Bureau.

[5] I. Gartner. (2014, January) Gartner Says Mobile Advertising Spending Will Reach 18 Billion in 2014.

[6] J. P. M. Consulting. (2014, January) IAB Interactive Advertising Outlook 2014. Interactive Advertising Bureau.

[7] I. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. Hassan, "Impact of Ad Libraries on Ratings of Android Mobile Apps," 2014.

[8] F. Consulting. (2014, May) The Value Of Rewarded Advertising. Rewarded Ad Formats Boost Consumer Receptivity To In-App Ads. Commisioned by TapJoy.

[9] D. Li, A. H. Tran, and W. G. Halfond, "Making Web Applications More Energy Efficient for OLED Smartphones," in *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. ACM, 2014, pp. 527–538.

[10] D. Li, S. Hao, J. Gui, and W. G. Halfond, "An Empirical Study of the Energy Consumption of Android Applications," in *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, September 2014.

[11] "https://play.google.com/store/apps/details?id=lv.n3o.shark."

[12] I. J. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. E. Hassan, "On Ad Library Updates in Android Apps."

[13] M. Harman, Y. Jia, and Y. Zhang, "App Store Mining and Analysis: MSR for App Stores," in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR)*, June 2012, pp. 108–111.

[14] M. Leppaniemi and H. Karjaluoto, "Factors influencing consumers' willingness to accept mobile advertising: a conceptual model," *International Journal of Mobile Communications*, vol. 3, no. 3, pp. 197–213, 2005.

[15] S. Soroa-Koury and K. C. Yang, "Factors affecting consumers responses to mobile advertising from a social norm theoretical perspective," *Telematics and Informatics*, vol. 27, no. 1, pp. 103–113, 2010.

[16] M. M. Tsang, S.-C. Ho, and T.-P. Liang, "Consumer Attitudes toward Mobile Advertising: An Empirical Study," *International Journal of Electronic Commerce*, vol. 8, no. 3, pp. 65–78, 2004.

[17] "http://code.google.com/p/dex2jar/."

[18] "http://www.distimo.com/leaderboards/google-play-store/united-states/top-overall/free/month."

[19] P. Mohan, S. Nath, and O. Riva, "Prefetching mobile ads: Can advertising systems afford it?" in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 267–280.

[20] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating Mobile Application Energy Consumption using Program Analysis," in *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, May 2013.

[21] "http://asm.ow2.org/."

[22] L. Gomez, I. Neamtiu, T. Azim, and T. Millstein, "RERAN: Timing- and Touch-Sensitive Record and Replay for Android," in *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 72–81.

[23] D. Li, S. Hao, W. G. Halfond, and R. Govindan, "Calculating Source Line Level Energy Information for Android Applications," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, July 2013.

[24] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "SIF: A Selective Instrumentation Framework for Mobile Applications," in *Proceedings of the 11th International Conference on Mobile Systems, Applications and Services (MobiSys)*, June 2013.

[25] T. Ball and J. R. Larus, "Efficient Path Profiling," in *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*. IEEE Computer Society, 1996, pp. 46–57.

[26] "Monsoon Solutions. Power monitor. https://www.msoon.com/LabEquipment/PowerMonitor/."

[27] "http://www.att.com/shop/wireless/data-plans.html#fbid=EhuxYcdIz02."

[28] D. Li and W. G. Halfond, "An Investigation into Energy-Saving Programming Practices for Android Smartphone App Development," in *Proceedings of the 3rd International Workshop on Green and Sustainable Software*. ACM, 2014, pp. 46–53.

[29] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What Do Mobile App Users Complain About? A Study on Free iOS Apps," *Software, IEEE*, 2014.

[30] H. Khalid, M. Nagappan, E. Shihab, and A. Hassan, "Prioritizing the Devices to Test Your App on: A Case Study of Android Game Apps," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering*. ACM, 2014.

[31] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, K. Papagiannaki, H. Haddadi, and J. Crowcroft, "Breaking for Commercials: Characterizing Mobile Advertising," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 343–356.

[32] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "ProfileDroid: Multi-layer Profiling of Android Applications," in *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, 2012, pp. 137–148.

[33] S. Shekhar, M. Dietz, and D. S. Wallach, "AdSplit: Separating Smartphone Advertising from Applications." in *USENIX Security Symposium*, 2012, pp. 553–567.

[34] A. J. Khan, V. Subbaraju, A. Misra, and S. Seshan, "Mitigating the True Cost of Advertisement- Supported Free Mobile Applications," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 1.

[35] B. Liu, S. Nath, R. Govindan, and J. Liu, "DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps," in *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2014.

[36] S. Nath, F. X. Lin, L. Ravindranath, and J. Padhye, "SmartAds: Bringing Contextual Ads to Mobile Apps," in *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. ACM, 2013, pp. 111–124.

[37] S. Hao, B. Liu, S. Nath, W. G. Halfond, and R. Govindan, "PUMA: Programmable UI-Automation for Large Scale Dynamic Analysis of Mobile Apps," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 204–217.

[38] P.-T. Chen and H.-P. Hsieh, "Personalized mobile advertising: Its key attributes, trends, and social impact," *Technological Forecasting and Social Change*, vol. 79, no. 3, pp. 543–557, 2012.

[39] D. J. Xu, S. S. Liao, and Q. Li, "Combining empirical experimentation and modeling techniques: A design research approach for personalized mobile advertising applications," *Decision Support Systems*, vol. 44, no. 3, pp. 710–724, 2008.

[40] K. Rasmussen, A. Wilson, and A. Hindle, "Green Mining: Energy Consumption of Advertisement Blocking Methods," in *Proceedings of the 3rd International Workshop on Green and Sustainable Software*. ACM, 2014, pp. 38–45.

[41] T. Book, A. Pridgen, and D. S. Wallach, "Longitudinal Analysis of Android Ad Library Permissions," *arXiv preprint arXiv:1303.0857*, 2013.

[42] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring." in *USENIX Annual Technical Conference*, 2012, pp. 387–400.

[43] A. Hindle, "Green Mining: A Methodology of Relating Software Change to Power Consumption," in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. IEEE Press, 2012, pp. 78–87.

[44] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," in *Workshop on compilers and operating systems for low power*, vol. 180. Barcelona, Spain, 2001, pp. 182–195.

[45] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling Resource Usage for Mobile Applications: A Cross-layer Approach," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 321–334.

[46] 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*. ACM, 2012, pp. 29–42.