Breaking for Commercials: Characterizing Mobile Advertising

Narseo Vallina-Rodriguez†, Jay Shah†, Alessandro Finamore‡
Yan Grunenberger◇, Hamed Haddadi§, Konstantina Papagiannaki◇, Jon Crowcroft†
University of Cambridge†, Politecnico di Torino‡, Telefonica Research◇
Queen Mary University of London§
name.surname@cl.cam.ac.uk†, finamore@tlc.polito.it‡, {yan,dina}@tid.es◇
hamed@eecs.qmul.ac.uk§

ABSTRACT

Mobile phones and tablets can be considered as the first incarnation of the post-PC era. Their explosive adoption rate has been driven by a number of factors, with the most signifcant influence being applications (apps) and app markets. Individuals and organizations are able to develop and publish apps, and the most popular form of monetization is mobile advertising.

The mobile advertisement (ad) ecosystem has been the target of prior research, but these works typically focused on a small set of apps or are from a user privacy perspective. In this work we make use of a unique, anonymized data set corresponding to one day of traffic for a major European mobile carrier with more than 3 million subscribers. We further take a principled approach to characterize mobile ad traffic along a number of dimensions, such as overall traffic, frequency, as well as possible implications in terms of energy on a mobile device.

Our analysis demonstrates a number of inefficiencies in today's ad delivery. We discuss the benefits of well-known techniques, such as pre-fetching and caching, to limit the energy and network signalling overhead caused by current systems. A prototype implementation on Android devices demonstrates an improvement of 50% in terms of energy consumption for offline ad-sponsored apps while limiting the amount of ad related traffic.

Keywords

Energy, Advertisement, Smartphones, Traffic, Cellular, Caching

Categories and Subject Descriptors

D.4.8 [**Performance**]: Measurements

1. INTRODUCTION

Mobile application (app) markets have been a major success and adoption factor for smartphones, allowing individuals and organizations to develop and sell apps to interested users. The App Store from Apple and Google Play (previously named Android Market) from Google are the two major platforms where developers sell or freely share their apps. Both Apple and Google have played a major role in democratizing revenues related to mobile apps. In particular, considering that 73% of the apps in Google Play are free [1], it can be expected that free apps tend to obtain a larger number of downloads than paid apps[1]. The revenue model adopted by many free apps includes advertisements (ads) that are embedded in the app and displayed at various points during use. The adoption of the ad model in mobile apps has strong implications for both the users and the network. As an example, a recent study shows that 65-75% of the energy consumed in a gaming app (Angry Birds) on Android devices is spent by third party advertising modules [2].

Mobile advertising has been the focus of recent research, targeting the system design [3], energy [2, 4], and privacy aspects of ad services [5, 6]. However, there is still very little known about the ad delivery mechanisms adopted in current mobile networks. This happens primarily because existing work tends to focus on the inspection of traffic generated by a small number of popular apps. In this work, we aim to characterize and quantify ad traffic in real mobile networks. For that, we study a data trace covering one day of traffic for more than 3 million subscribers of a major European mobile carrier.

In such an attempt, we develop a methodology for the classification of ad traffic that incorporates the inspection of the SDKs provided by ad networks, traffic inspection of mobile apps, as well as rules extracted through a web log obtained from the aforementioned network. Using a rule set comprising 122 rules, we classify traffic into *i)* ad networks, *ii)* analytics, and *iii)* mediation services, and study its characteristics along a number of dimensions (traffic volume, frequency, type of content, etc.) for three major mobile platforms found in the trace. Our analysis reveals several properties of the mobile ad ecosystem:

- The mobile ad ecosystem is overcrowded and unmoderated, with AdMob and other Google services being the leaders.
- Ads are not just a strain on Android devices but are also prominent on Apple devices.
- Ads account for 1% of all mobile traffic in our data set, a significant component of the daily traffic of each device. Re-

[1]Although no ground truth data has been publicly available to confirm this conjecture.

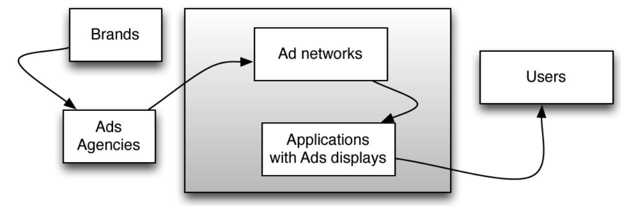


Figure 1: The mobile advertising ecosystem and the interaction of the players within it.

Provider	Retrieval Mechanism		Refresh Interval (s)	
	Push	Pull	Min	Max
AdMob	✓	✓	12	120
Millennial Media		✓	15	N/A
InMobi		✓	20	N/A

Table 1: Description of the refresh intervals and retrieval mechanisms supported by three popular ad networks.

sults show that for 3% of Android devices, ads account for more than 1 MB, while for Apple devices this is even higher and corresponds to more than 3 MB.

- Mobile ad traffic is mainly composed of *static* images and text files that are likely to be *re-downloaded*, with refresh intervals in the order of just a few seconds.
- Mobile ad traffic is the only type of network activity for some apps.

On a wired network, the transmissions of traffic can be assumed to be "free”, however on a mobile network these are accompanied by control channel signalling overhead, use of scarce spectrum resources and battery implications. Low refresh intervals for downloading objects that have already been fetched previously further depletes the scarce resources of mobile networks. Moreover, solutions, such as pre-fetching and caching, are well understood in the system and networking community to readily provide solutions to a more spectrum-aware and energy-aware delivery of ads. Leveraging these techniques, we developed a system, called *AdCache*, that enables energy efficient and network friendly cache-based ad delivery, built on the intrinsic characteristics of ad traffic and mobile apps. AdCache enables the retrieval of ads under optimal network connectivity conditions for later display to the user, hence avoiding excessive energy wastage, signalling strains on the network and worsening the app responsiveness. We implement and test AdCache on Android devices and show that it is able to reduce the energy consumption of mobile advertising in offline apps by up to 50% even with a low ad refresh interval of 20 seconds.

2. EXTRACTING AD TRAFFIC

The mobile ad ecosystem, as detailed by Leontiadis et al [1], comprises multiple players: *brands* wanting to attract consumers, *ad agencies* designing ad campaigns for brands, *ad networks* used for distribution, *publishers* who create and publish mobile apps, and *users* to which ads are shown. *Mediation services* are an additional player that integrates several ad networks, allowing publishers to combine different ad networks and switch between them on the fly. Their main advantage is that they can potentially increase the publisher’s revenues as if one ad network fails to return an ad at its slot, it can try another ad network to fill this gap.

Rank	Application Name	Category	Ad Provider
1	Facebook	Social Network	N/A
2	Talking Pierre	Entertainment	MobClix
3	Ceramic Destroyer	Arcade	AdMob
4	WhatsApp	Communication	N/A
5	Cartoon Camera	Photo	MobFox, MadVerti
6	Skype	Communication	N/A
7	Angry Birds	Arcade	Burstly
8	Onavo	Tools	N/A
9	Talking Tom Cat 2	Entertainment	MobClix
10	Viber	Communication	N/A

Table 2: Usage of ad networks on the top 10 most popular free mobile apps in the UK (As of 27th Feb. 2012).

URL domain	Object path	Type	Role
media.admob.com	adk-core-v40.js	Ad Net	Conf. Script
*.g.doubleclick.net	mads/gma	Ad Net	Get Ad
*.googlesyndication.com	pagead/simgad	Ad Net	Get Ad
*.googlesyndication.com	pagead/js	Ad Net	Static content
*.googlesyndication.com	pagead	Ad Net	Static content
*.g.doubleclick.net	aclk	Ad Net	Report Click

Table 3: Extract of the rule set for AdMob.

Ad networks and publishers are pursuing common objectives. Ad networks wish to maximize the number of clicks on ads through targeting the right users to satisfy the demands of the advertisers. Meanwhile, publishers are looking to maximize their revenue by increasing their click-through rate (the number of clicks on an ad divided by the number of times an ad is shown), using mediation services to fill up their advertising space, and obtain profiling information for targeting. Rather than inspecting all the relations between the players in this complex ecosystem, we focus our interest on the distribution mechanism used by ad networks and ad-sponsored apps running on the device, as depicted in Figure 1.

2.1 Understanding ad networks’ SDKs

The ecosystem leverages the relative simplicity in incorporating ads in mobile app development. Ad networks provide a Software Development Kit (SDK) that enables integration of ads into mobile apps, hiding the protocol peculiarities. As shown in Table 1, popular ad networks such as AdMob, Millennial Media and InMobi allow developers to define which kind of ads are embedded, how they are delivered (push/pull techniques) and how often they are refreshed. The most common type of ad in mobile apps are banners (they are placed at the top or bottom of the screen and span its width) and interstitials (full-screen ads, covering a large part or all of the screen for a short period of time). Unlike banners, interstitials are typically shown as users transition between different activities in the app. Banners are usually composed by text, images, and Javascript code.

The protocols used by ad networks for fetching and reporting are generally based on plain HTTP requests using REST APIs, with most using HTTP GET methods. However, the ad networks studied differ slightly in the way they interact. As an example, AdMob acts as an internal mediation service to aggregate all Google’s services (e.g. Doubleclick and AdSense) whereas InMobi only requires a single HTTP POST request per action. Millennial Media needs two HTTP connections with two different servers: one to get the ad, and the other one to get the associated static content.

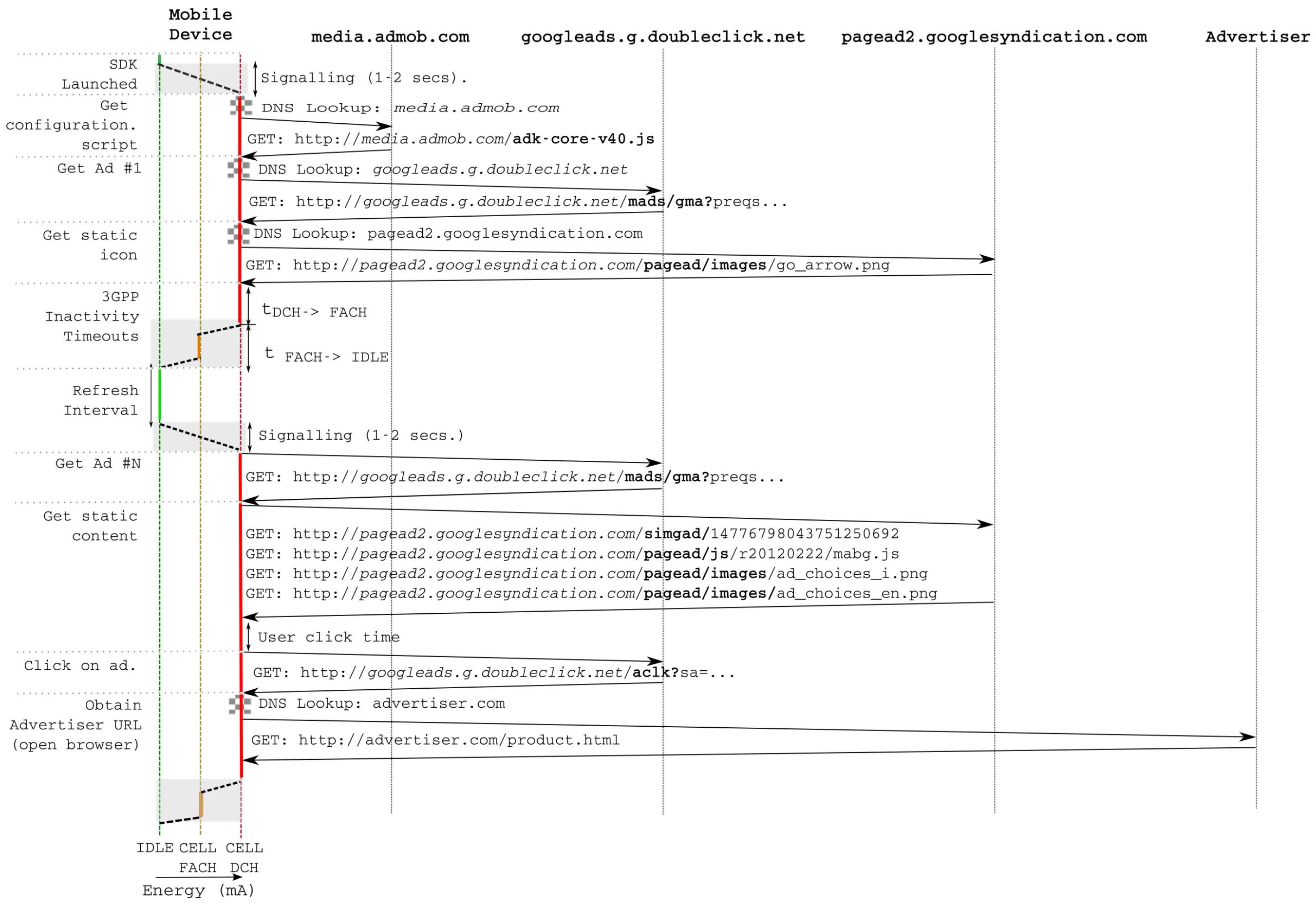

Mobile Device
media.admob.com
googleads.g.doubleclick.net
pagead2.googlesyndication.com
Advertiser
SDK Launched
Signalling (1-2 secs).
Get configuration. script
DNS Lookup: media.admob.com
GET: http://media.admob.com/adk-core-v40.js
Get Ad #1
DNS Lookup: googleads.g.doubleclick.net
GET: http://googleads.g.doubleclick.net/mads/gma?preqs...
Get static icon
DNS Lookup: pagead2.googlesyndication.com
GET: http://pagead2.googlesyndication.com/pagead/images/go_arrow.png
3GPP Inactivity Timeouts
tDCH-> FACH
t FACH-> IDLE
Refresh Interval
Signalling (1-2 secs.)
Get Ad #N
GET: http://googleads.g.doubleclick.net/mads/gma?preqs...
Get static content
GET: http://pagead2.googlesyndication.com/simgad/14776798043751250692
GET: http://pagead2.googlesyndication.com/pagead/js/r20120222/mabg.js
GET: http://pagead2.googlesyndication.com/pagead/images/ad_choices_i.png
GET: http://pagead2.googlesyndication.com/pagead/images/ad_choices_en.png
User click time
Click on ad.
GET: http://googleads.g.doubleclick.net/aclk?sa=...
Obtain Advertiser URL (open browser)
DNS Lookup: advertiser.com
GET: http://advertiser.com/product.html
IDLE
CELL FACH
CELL DCH
Energy (mA)

Figure 2: Example of a real work flow for AdMob. The device incorporates the three power modes for mobile interfaces, and the transitions between them. Transport layer details are not included. The HTTP requests include the domain name and the strings used to identify the semantics in bold.

Figure 2 shows an example of a real work flow for an offline app requesting AdMob ads and the action triggered on a user click. The figure highlights the semantics of the HTTP requests (shown in Table 3) and some of the domains used by AdMob. Once a radio channel has been established, the client performs a DNS lookup and the necessary HTTP transactions. As we can see, AdMob comprises several services and the number of HTTP requests can be considerably high. A number of intermittent connections are going to be initiated depending on the refresh interval, causing costly transitions between the different radio power modes. To our benefit however, the resulting HTTP ”conversations” tend to have a consistent structure, allowing one to derive a number of regular expressions that could be used in the classification of ad traffic.

2.2 Identification of ad traffic

Table 2 lists the top 10 most popular free apps on Google Play, as of 27th Feb. 2012, and the category that each app belongs to. Manually inspecting the traffic on a device, we also identified the ad networks used by each app. We can see how, only considering this small set, the ad ecosystem is highly diverse with a number of different players, potentially following different approaches in the delivery of ads.

To classify ad traffic accurately, we need to be able to incorporate such diversity. We perform such a task in two ways: *i)* by capturing traces from app execution to understand the cause-effect relationships (e.g. launching an ad-sponsored mobile app, requesting and clicking on the ad displayed), or *ii)* by inspecting traffic traces that are collected within a provider in order to identify peculiar characteristics of the traffic (e.g. hostnames or URL parameters). In the former case, we will be able to only inspect the ad delivery strategy chosen by the app publisher. In the latter case, we can further obtain information about the strategies followed by multiple publishers, as well as the diversity in ad delivery, as manifested through the use of different mediation services, and ad networks. In both cases, however, the fundamental challenge at hand is to be able to derive a comprehensive set of rules to classify the flows that are related to ad networks, mediation services, and any possible analytics traffic, that facilitates the targeting of users with relevant ads.

With that in mind, we have used four different techniques. First, we manually inspected the ad network SDKs and their documentation (if publicly available) to capture the way they use the network. Second, we installed a number of popular ad-based apps on an Android phone, and ran *tcpdump* to inspect what kind of services they use and the network traffic they generate, including the HTTP requests. Third, we created our own app that displayed an ad banner, and connected it with the most popular ad networks, to characterize the network behavior observed across a variety of ad networks. Lastly, we analyzed a trace containing all HTTP transactions car-

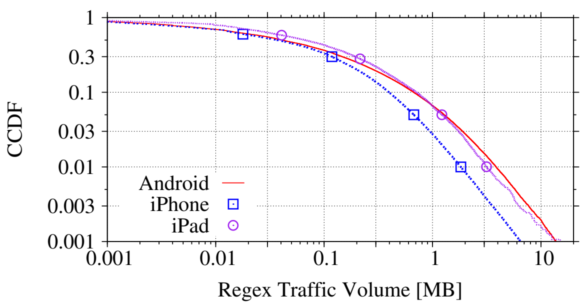


Figure 3: CCDF of the volume of ad traffic per user.

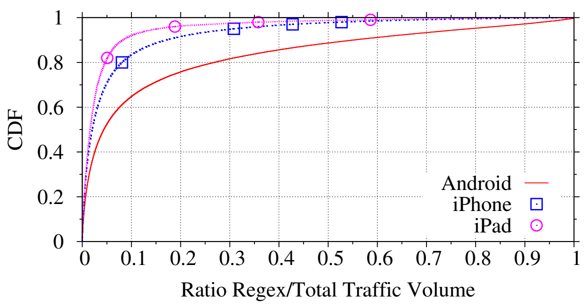


Figure 4: The fraction of mobile ad ecosystem traffic over the total traffic.

ried out by more than 3 million subscribers during a whole day on a mobile network covering an entire country in Europe. In this latter case, we made sure to filter out traffic resulting from non-mobile devices, given that the network in question allows tethering and offers 3G USB dongles that are used by non-mobile devices to connect to the mobile network.

The combination of the four processes, mentioned above, helped us obtain a set of 122 rules featuring regular expressions that are able to parse a URL and classify traffic as *i)* ad network, *ii)* analytics, *iii)* mediation service, or *iv)* other. The rule set also helps us to identify the type of action to be performed given a HTTP request. In Table 3, we show some examples of the rules obtained. Each rule, defined by a domain and URL object path, identifies the type of service, goal of the request (e.g. obtain configuration script, an ad or reporting a click), ad network and HTTP request method (e.g. POST vs. GET) used. The compiled dictionary is publicly available at [7].

3. AD TRAFFIC CHARACTERISATION

In this section we characterize ad traffic leveraging on a data set containing 1.7 billion traffic connections, which corresponds to 22TB of volume downloaded on 13th Aug. 2011 by more than 3 million subscribers of a major European mobile network. The data set comprises all TCP traffic, excluding HTTPS, seen by acceleration proxies installed on the network, that also cache and compress content for more efficient delivery. To the best of our knowledge, and according to the level of traffic at the peering point of this provider, we estimate the proportion of the secure traffic to be 11.5%.

The monitoring activity at each vantage point is reported in a set of text log files. Each entry in the logs contains a set of standard information, such as IP addresses, port numbers, number of bytes downloaded, and other HTTP specific information, such as content type, HTTP user agent, and HTTP response code. User information is anonymized but consistently hashed to a unique ID. Each line of the logs corresponds to a different TCP connection performed by a user.

The operator under study allows tethering and 3G connectivity through a USB dongle. In order to ensure that our analysis focuses on ad traffic generated truly by mobile devices and apps, we have filtered out this traffic. To achieve this we used a methodology based on the inspection of the HTTP user agent, normally including information about the Operating System (OS) of the device. All PC users, as well as those found to be using more than one OS during the day, have been filtered out from the original data set. This amount of traffic is not negligible, corresponding to 29.6% of the bytes downloaded by 7.2% of the users. This step further allows us to study ad delivery against the type of the platform used, e.g. Android, iPhone and iPad.

In what follows, we study the ad, mediation, and analytics traffic (called regex traffic for short) extracted from the sanitized mobile data set. We want to highlight that analytics services provide a standard framework to control the general activity of apps so they are not adopted in ad-sponsored apps only. However, we decided to include them in the analysis for completeness. We focus our characterization along five main axes: *i)* overall volume of ad services, *ii)* main actors, *iii)* type of ad content, *iv)* generating apps, and *v)* ad frequency.

3.1 Ad volume

Figure 3 shows the Complementary Cumulative Distribution Function (CCDF) of the regex traffic volume. While this traffic only corresponds to 31 kB per day for 50% of the devices, the distributions are characterized by their long tails. We can see how 10% and 1% of the iPhone devices download more than 400 kB and 2 MB of advertising related content respectively, while Android and iPad devices consume even more. Unexpectedly, the top 5 users together have consumed 1.35 GB, 701 MB and 124 MB just for regex traffic on Android, iPhone and iPad respectively. The single top users account for 458 MB, 167 MB and 31 MB for the three platforms.

These values might be the indication of outliers in the population. To better evaluate the weight of the regex traffic, Figure 4 reports the Cumulative Distribution Function (CDF) of the ratio between the volume of regex traffic and the total volume of traffic consumed by each user in the entire day. We can see that the traffic caused by the mobile ad ecosystem players is a strong component of the users' traffic. In particular, for 20% of iPad devices this traffic accounts for more than 4.7% of their traffic, while for iPhone devices it is 7.6%. This phenomenon is even more critical for Android users with 50% of them having more than 5% their total volume related to regex traffic.

3.2 Main actors

The regex data set contains three classes of ad traffic: ad networks, analytics services, and mediation services. These classes do not have the same weight on the aggregate volume. Table 4 reports the percentage of users, flows, and bytes for the three classes of traffic across the three most popular mobile platforms. We can see some differences across the device types. In particular, Android and iPhone present similar shares while for iPad both mediation and analytics services are less adopted. More than 90% of the volume is delivered by the ad networks, but we can also notice a sharp component of "control" traffic related to analytics and mediation services.

	%Users			%Flows			%Bytes		
Devtype	AN	AS	MS	AN	AS	MS	AN	AS	MS
Android	81.3	61.0	32.2	65.7	15.6	18.3	90.7	2.5	6.8
iPhone	77.3	60.0	23.6	65.5	22.2	12.3	89.4	5.7	4.9
iPad	88.4	35.8	13.7	87.2	6.9	5.9	96.9	1.5	1.6

AN = Ad Net., AS = Analytics Serv., MS = Mediation Serv.

Table 4: Breakdown of regex traffic with respect to class of traffic and device type.

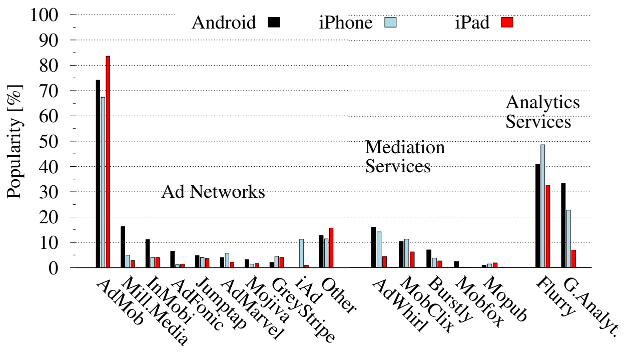


Figure 5: Ads and analytics services popularity.

In Figure 5 we show the main ad networks, mediation and analytics services found in our data, along with their popularity (in terms of the percentage of users using them). The figure is partitioned into three areas, marking each type of service. The three different bars capture the statistics for each type of mobile platform. Services are sorted with respect to the popularity for Android devices with iAd (Apple's advertising service) added for completeness even though it is available only for Apple devices. AdMob is clearly the dominant service across all the ad networks serving 74.5%, 67.5% and 84.2% of all users. The remaining portion of the market is shared across many other ad networks with Millennial Media and InMobi leading only on Android devices. Furthermore, more than 10% of the users communicate with services outside the top 10, underlining the overcrowded nature of this ecosystem of services. Interestingly, the market share of iAd in iPad is modest, being overtaken even by smaller ad networks, such as GreyStripe and Jumptap. Mediation services are dominated by AdWhirl, MobClix, and Burstly, but in this case the differences are less significant than for ad networks.

Google's main source of income is advertising[2]. The prominent presence of Google in the mobile ad ecosystem is also clearly visible as AdMob, Google Analytics, and AdWhirl (open source but under Google's umbrella) are the dominating services. This dominance can be seen in terms of both popularity (Figure 5) and also in terms of volume and flows (Figure 6). In more detail, Google services on Android devices account for 73% and 80% of ad flows and bytes respectively whereas for iPhone devices the fraction of volume is lower due to the presence of iAd which accounts for 8% of the total bytes. AdMob's presence is even stronger on the iPad as it accounts for almost 90% of the total ad traffic on the platform.

Finally, Google Analytics and Flurry are the only two analytics services we could identify in the data set. The analysis reveals that these services are very popular across mobile apps, going beyond the values obtained by mediation services. The limited popularity of mediation services across mobile users indicates that mobile apps are more likely to interact directly with ad networks, namely

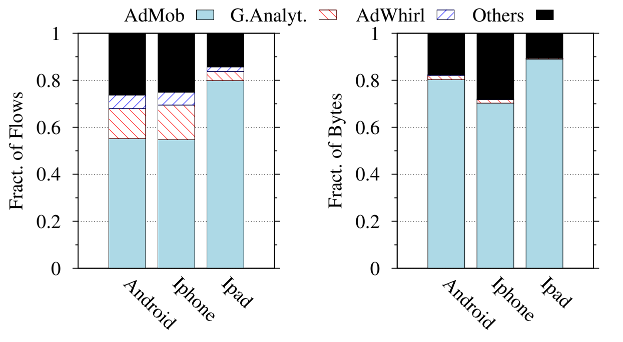


Figure 6: The fraction of flow and bytes of Google services.

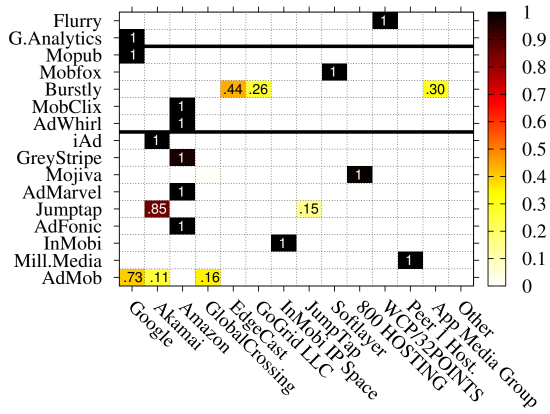


Figure 7: Distribution of the volume of each ad and analytics service among the organization contacted. The y-axis reports the ad service names sorted as in Figure 5 and the x-axis reports the set of organizations used by the ad services.

AdMob, instead of relying on third party agencies. We hypothesize that this might be related to service quality, economical aspects, or implementation details, however we cannot demonstrate this with our data collection.

3.3 Type of ad content

As we schematically reported in Figure 2, ad traffic content is a combination of images, HTML and Javascript code. Inspecting the HTTP content type header field, we found that static content such as `'image/*'` accounts for 31.4%, 41.7%, and 49.1% of Android, iPhone, and iPad devices' ad volume respectively. The remaining portion is instead shared between `'text/javascript'` and `'text/html'` which are generally used to configure the client, dynamically load an ad and define its visual layout. Displaying such kind of content on a mobile app also affects performance as it requires a browser component to run completely embedded within a native app.

While images are static objects by definition, different scripts are also used to define the ad layout and behavior or client configurations, so they are more subject to change. To inspect the time variability of the content, we set up a simple experiment. We selected the top 1000 most popular objects for each device type, i.e., the content requested by the majority of the devices in the data set, with each set of objects requested once per hour from a PC over a day. Comparing the objects returned we found that for 95% of

[2] `http://investor.google.com/financial/tables.html`

Top AdMob Android Apps

Rank	App Name	Category	Users (%)
1	Angry Birds	Arcade	11.48
2	Advanced Task Killer	System Tools	9.77
3	Soccer Scores (FotMob)	Sports	3.53
4	Drag Racing	Arcade	2.69
5	Bubble Blast	Arcade	2.69

Top AdMob iPhone Apps

Rank	App Name	Category	Users (%)
1	TV Guide	Entertainment	5.96
2	Grindr	Dating	4.21
3	iFooty	Sports	4.01
4	Words with Friends	Arcade	3.51
5	Solitaire	Arcade	2.80

Table 5: Top apps requesting AdMob ads both in Android and iPhone. The right column indicates the percentage of users that had installed the app out of the total users with AdMob traffic for a given platform.

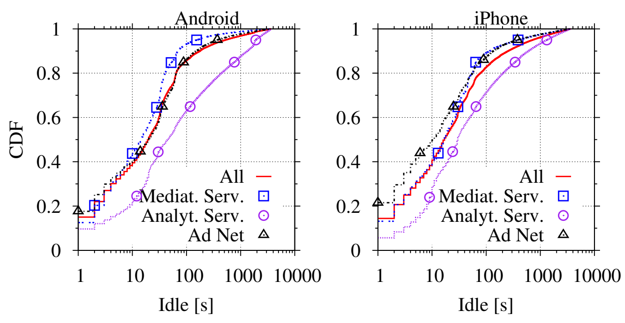


Figure 8: CDF of the interleave between consecutive activity periods.

the cases there are no differences and many objects in the remaining 5% corresponds to scripts differing only because of timestamps embedded in the code, proving that ad content for mobile apps is static.

Inspecting the hostnames and server IPs we have noticed that the content is usually served through Content Delivery Networks (CDNs). For each ad service we identified the CDN/hosts they use, measuring the amount of bytes served by the CDN/host and normalizing the values with respect to the total volume of each ad service. To retrieve this information we relied on a commercial database provided by MaxMind[3] that maps an IP address to the name of the organization (AS, CDN, network operator, hosting company, etc.) that owns it. In Figure 7, we report a heat map to show the relationships between ad services and the CDN/hosts serving their content. The y-axis reports the ad service names sorted as in Figure 5. The services are grouped together according to the three classes of services previously introduced: analytics services, mediation services and ad networks. The x-axis reports the set of organizations used by the ad services. From the market point of view, the heat map is sparse, given that most ad services use a different organization for serving content. Most of the ad services are served by a single organization, except AdMob, Burstly and Jumptap, all of which balance the volume downloaded across 2 or 3 organizations. Beside Google, the only exception is Amazon which is preferred to Akamai by many ad services.

3.4 Greedy apps requesting ads

Traffic classification of mobile apps is not a trivial task. The methodologies available in the literature are usually based on the inspection of both the user agent and URL of HTTP requests [8]. However, we found these techniques inaccurate especially for Android where the user agent is usually not customized by app publishers. Considering the URLs of ad traffic, we noticed some parameters related to configuration (e.g., format and size of the ad), and tracking info (e.g., country code, GPS position) but we found also some identifiers related to the app name generating the traffic. In particular, some ad services identify apps using a hash code, the package name (e.g., *com.rovio.angrybirds*) or the real name (e.g., *Angry Birds*). While package names can be mapped to the real name using information available on the market, this is not true for hash codes which are created by the service when the publisher registers the app. Given these constraints, we decided to focus only on app using AdMob since *i*) is the most popular ad service (as seen in Figure 5) and *ii*) is the only service we found having the documentation describing the URLs parameters related to the app name[4].

In Table 5 we report the top 5 apps, on Android and iPhone, requesting ads from AdMob either directly using the SDK or via mediation services. For each app listed, we detail the category and the popularity, defined as the ratio between the number of devices using the app and the number of devices contacting AdMob. Most of the apps listed do not require the network access to use their intended functionality. This is the case with games such as *Angry Birds*, *Bubble Blast*, and system tools such as *Advanced Task Killer*. Other apps related to sport, TV/cinema or social networking, such as *Grindr*, instead require network access to perform properly. As we will see later in more detail, in the first case ads not only represent a “waste” in volume but they also increase the energy consumption since downloads are likely to be happening outside of other network activity.

Comparing the device types, we notice differences in the distribution of the popularity and app categories. For Android, the popularity is skewed with the first two apps accounting for 20% of all devices. For iPhone, instead the differences in the popularity are smoother. Interestingly, Angry Birds is top for Android but it is 20th for iPhone with a popularity of only 0.66%. We also noticed that AdMob served more ads to games on Android than to iPhone as eight out of the top 10 Android apps would operate offline if it did not display mobile ads.

3.5 Ad traffic frequency

Ad traffic is, by nature, periodic, and one of the important parameters controlling it is the refresh interval, i.e. how frequently the mobile app requests an ad. We have also seen that this traffic is mainly static and, given the finite catalog of ads available from a service, an object might be requested multiple times. In this section we investigate the frequency of ad traffic considering both these effects, starting from the characterization of traffic aggregates and then moving more to the details of the specific objects.

3.5.1 *Request interval*

As described in Section 2.1, different ad services adopt different protocols to deliver and manage ads. Reverse engineering is a difficult and time consuming task so we decided to opt for a more general approach to capture the frequency of the traffic grouping flows

[3] `http://www.maxmind.com/app/organization`

[4] For iPhone devices the app name is usually carried in the `app_name` parameter. For Android devices it can be deduced from the app package name specified in the `msid` URL parameter.

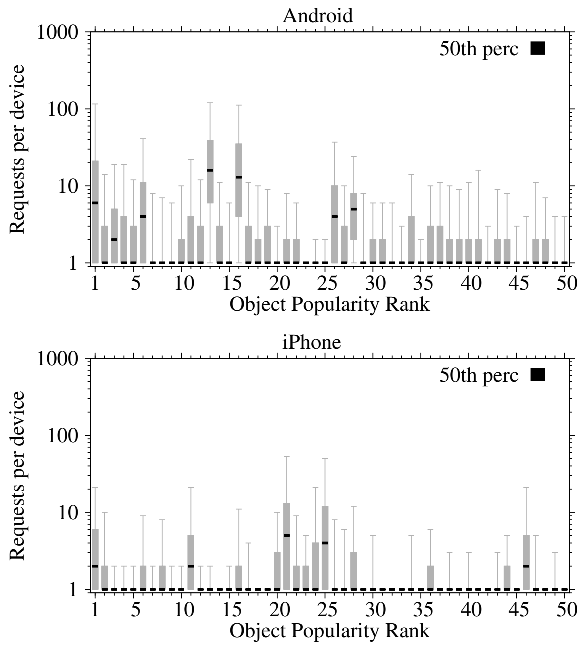


Figure 9: Box plots of the number of requests each device performs for the top 50 most popular objects on Android and iPhone devices.

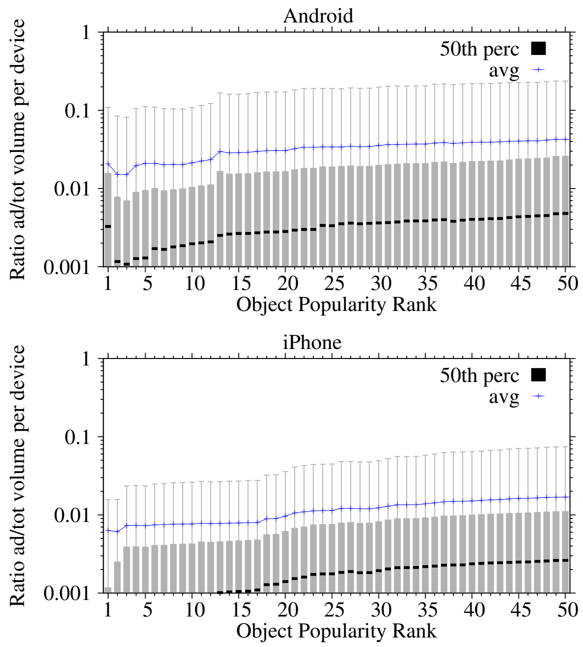


Figure 10: Box plots of the fraction of cumulated volume related to the 50 most popular objects on Android and iPhone devices.

in time: we define an *activity period* as a group of concurrent flows such that two consecutive flows *flow-A* and *flow-B* are part of the same group if $start(flow_A) < start(flow_B) < end(flow_A)$. In this way, we cluster the traffic in time and studying the interleave between activity periods gives us an indication of the traffic frequency.

In Figure 8 we report the CDF of the interleave between two consecutive activity periods. Each of the three classes of ad traffic is considered individually and we also report the aggregate for completeness. We can see that Android and iPhone handsets present similar distributions (this is true also for iPad but not reported due to its similarity) but differences emerge between the different classes of traffic. In fact, analytics services are less interactive than mediation services which typically generate multiple flows in a very short period of time as they need to communicate both with its servers and the ad network to report the action and obtain the ad respectively (as seen in Figure 2). Despite the static nature of ad traffic, 40% of the activity periods are interleaved by less than 10 seconds and more than 80% in less than 100 seconds.

3.5.2 Re-downloads

Given the static nature of ad traffic content, it is reasonable to expect that the SDKs provide some caching capabilities to limit the number of re-downloads of the objects, but instead they use standard HTTP libraries. Studying the HTTP response codes, we found that only 5% of the requests from Android devices receive a "`HTTP 304 Not Modified`" reply while this accounts for only 2% of requests from Apple devices. The limited adoption of conditional HTTP requests suggests that, in presence of multiple requests for the same object, it is very likely that it is re-downloaded.

In order to verify this assumption, for each object requested we computed its popularity as the ratio between the number of devices requesting such an object and the total number of devices that had ad traffic. For each object ranked in the top 50, we computed the distribution of the number of times such an object was requested by each device. The box plots shown in Figure 9 report the 5th, 25th, 75th and 95th percentile of each distribution. The median is equal to 1 for most of the objects but their distribution is heavy tailed. In particular, this is more notorious for Android devices, likely due to potential bugs in the HTTP libraries as reported by [9]. Nevertheless, different objects present different distributions. In particular, the objects presenting the highest number of requests (13/16th for Android and 20/25th for iPhone) correspond to requests generated by Angry Birds.

Figure 10 reports the distribution of the cumulated volume caused by all HTTP requests, to the top 50 objects, as a fraction of the total traffic generated by each device. Considering Android devices, we can see that the volume related to the top 10 objects is limited, as most of the devices have a median of 0.002%. However, the distributions are heavy tailed and for 25% of the devices the top 10 objects account for 1% of the total volume. Despite the fact that 1-2% of user volume may appear to be negligible, this fraction relates to very few objects. It is interesting that such a small set of content can have this impact of the overall traffic of a device. Moreover, this waste is not just related to volume as the unnecessary transmissions also have energy costs.

4. ADS AND MOBILE NETWORKS: ENERGY IMPLICATIONS

Most mobile devices boast a 3G network interface running on the UMTS standard, with an IP stack of upper layers protocols. In order to maximize the efficiency of spectrum allocated to these networks, each terminal (or *user equipment* in UMTS terminology) is associated with the *Radio Resource Control* (RRC) state machine that is responsible for the actual behavior in terms of bandwidth, power consumption and latency of the physical layer. While the specific parameters might have different values from one network provider to another [10], most networks define three power modes: IDLE which corresponds to no connection; CELL_DCH (dedicated

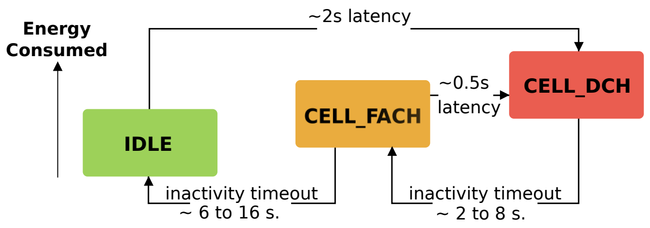

Energy Consumed
~2s latency
IDLE
CELL_FACH
CELL_DCH
~0.5s latency
inactivity timeout ~ 6 to 16 s.
inactivity timeout ~ 2 to 8 s.

Figure 11: A common 3G wireless RRC state machine.

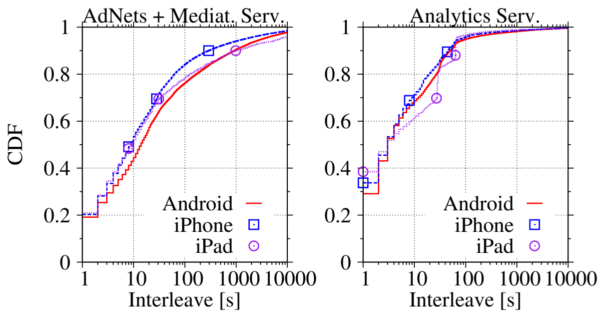


Figure 12: CDF of the interleave for each pure activity period.

channel), the highest power state with highest throughput and lowest latency; CELL_FACH (forward access channel) used to reduce the latency caused by going from IDLE to CELL_DCH in case another transmission occurs within a few seconds after the previous transmission[5]. All promotions are based on traffic volume whereas demotions are triggered by inactivity timers defined by the network operators, as shown in Figure 11.

In the previous sections we have seen how ad traffic has an intrinsic intensity due to the frequent communications between apps and ad networks, and we also observed that some requests lead to a waste of traffic as it may result in objects being re-downloaded. In this section, we will characterize and quantify the effect of ad traffic on power consumption. We start by investigating the isolation of ad traffic with respect to the RRC state machine, and then present the power consumption results obtained using an extensive set of active experiments in a controlled environment.

4.1 Ad traffic isolation

In Section 3.4 we have seen that some of the most popular adsponsored apps are likely to use the network only to download ads. Furthermore, a recent study revealed that 60% of the free apps in the Android market require network access against 30% of paid apps [1]. This suggests that ad traffic will likely be isolated, i.e., found when the device is not generating any other traffic. Using the concept of activity periods as previously introduced, we can define that ad traffic is isolated if the activity periods are *i*) *pure* - only contain ad traffic, and *ii*) *interleaved* - the device has been silent for some time before sending ad traffic. Due to the limitations of our dataset, the isolation factor can be only computed without including other non-TCP traffic such as DNS lookups so the results obtained represent the upper bound. Moreover, since analytics services are not strictly related to advertising, for this analysis we will consider them separately, while grouping ad networks and mediation service traffic into a single class.

We found that 81.1%, 68.2% and 69.7% of activity periods are pure for Android, iPhone and iPad devices respectively. While it is possible that some apps are actually using the network, the lower percentages obtained for Apple devices is likely due to *push notifications*, a background service characterized by persistent long lived connections used by the servers to send updates to the devices only when needed. When a device is using this service, it is common to have at least one push notification in each activity period, even if it might not generate any traffic during that period of time. This intuition is confirmed by the fact that filtering this traffic the percentages of pure activity periods increase to 78.5% and 74.3% for iPhone and iPad respectively. Interestingly, we found that 12-20.4% of the pure activity periods contains only analytics traffic, while only less than 1% has all three services. This shows that analytics traffic is not generated simultaneously with ad networks and mediation services traffic.

For each pure activity period identified, we measured the interleave, i.e. the amount of silence preceding the group of flows. In Figure 12 we report the CDF of the interleaves separating analytics services from the other types of traffic and detailing the results for different device types. Considering analytics services, we can observe that the distributions are very similar between the device types. Nearly 50% of the activity periods happen within the first 2-3 seconds from the last communication, and 94.8% within 1 minute. Instead, ad networks and mediation services present opposite characteristics. Moreover, interleaves are larger than for analytics services, with 20% of activity periods having more than 1 minute for iPhone devices, while 40% and 32% happen within 6 seconds.

We also note that for ad networks and mediation services the interleaves distribution is very similar to the results of Figure 8. Analytics services present clear differences as they are not strictly related to ad traffic. We noticed that 80% of the pure analytics activity periods are preceded by mixed activity periods, and that the opposite holds for ad networks and mediation services. Overall, we state that ad traffic is isolated and the high interleaves found indicate that the device radio is likely to be in the IDLE power mode when they request ad traffic.

4.2 Energy consumption of ad networks

The previous sections revealed several inefficiencies caused by mobile ad traffic with implications on mobile networks and the battery life of mobile devices. However, the mobile network data set does not capture the activity on the RNC or lower layer traces, so we were not able to estimate the real impact of ad traffic on the battery life from the network traces.

In this section we describe a set of active experiments run on a real device with a power meter to accurately estimate the power impact of mobile ad traffic. We take advantage of the fact that ad traffic is generally found in isolation, so there is generally no background traffic. While the traffic analysis has exhibited the presence of analytic services traffic, we have seen that the nature of this traffic is less interactive than traffic generated by mediation services and ad networks.

4.2.1 Methodology

We used a purpose-built app for evaluation, requesting ads from the three most popular ad networks (AdMob, Millennial Media and InMobi). The app only contains an ad slot at the bottom of the screen, to avoid incurring any extra CPU, I/O, and network costs. This allows us to focus the energy evaluation solely on the cost

[5]There is also an optional CELL_PCH (Cell paging channel) between IDLE and CELL_FACH.

Average current consumption (mA)

Power Mode	Mobile Network	Wi-Fi
Airplane Mode. Idle	2.1(0.1)	
Airplane Mode. Min Bright.	109.4 (0.1)	
Airplane Mode. Max Bright.	140 (1.4)	
Idle	3.3 (0.1)	5.4 (0.4)
Min brightness	110.6 (0.2)	118.6 (1.2)
Max brightness	141.3 (0.3)	149.4 (1.0)

Table 6: Average current consumption for a Samsung Galaxy Nexus S. In brackets, the standard deviation.

of fetching, displaying, and refreshing ads. The ad size used was 480×80 pixels across all networks tested (standard banners).

Measurements were taken with different refresh intervals (20, 40, 60 and 80 seconds), on 3G and Wi-Fi, each done ten times for three minutes under controlled conditions. Since we could not present detail on the amount of ad traffic potentially offloaded on Wi-Fi networks in our traffic analysis, we included the wireless interface on the energy evaluation, as many users use Wi-Fi when available. For 3G, the experiments were carried under good network conditions on a large European operator. The network type was 3G and the signal strength varied from 16 to 21 ASU (max is 31). The Wi-Fi experiments were done on a public Wi-Fi access point connected to a DSL line with background traffic. The intervals were chosen to give a good estimate of the real cost that publishers are adding to offline apps by including ads.

The popular Monsoon high-resolution Power Monitor was used to measure the current consumption by the device while running the app. To ensure that the conditions were kept constant, the brightness and volume were set to the minimum level, whereas notifications were disabled and all other apps were closed. The display is required to be active as otherwise the ads do not refresh.

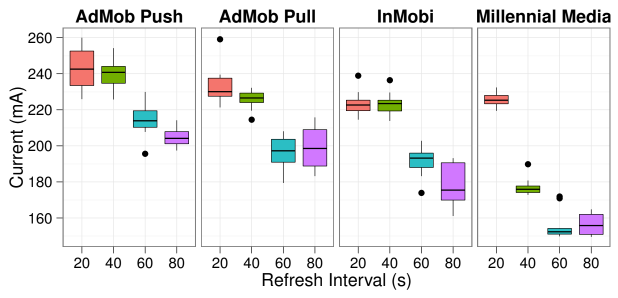
4.2.2 *Baseline energy consumption*

Table 6 shows the energy costs in mA[6] for the basic power modes of a Samsung Galaxy Nexus S. These values are specific for the device used in the study, and will vary for other devices. These values aim to describe the standard configuration of a modern mobile device without any CPU and network-intense activity being executed (only basic OS activity was allowed). As can be seen, one of the most energy-intense resources is the display. The current values obtained are similar to previous experiments which tried to benchmark the energy consumption in modern mobile devices [11]. Being connected to a mobile network is more efficient than keeping the device connected to Wi-Fi APs. It should be noted that the higher variability in the results for Wi-Fi is caused by background broadcast traffic existing in the network. Wi-Fi results are highly dependent of the access point in use. The obtained results represent a worst case scenario as the energy consumption of Wi-Fi clients depends on the support of power saving mode on the access point and the characteristics of the mobile terminal itself [12].

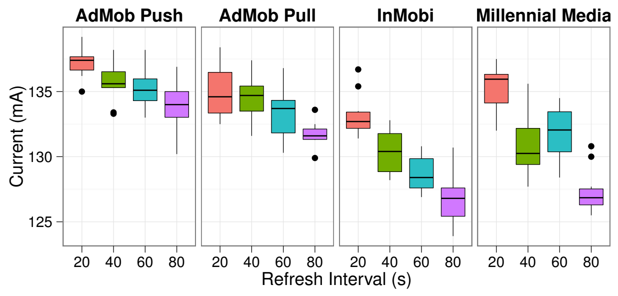
4.2.3 *Energy cost: fetching ads under mobile networks and Wi-Fi*

The parameters that publishers can control such as the refresh interval and retrieval mechanism significantly impact the power costs. Figure 13 shows the current expenditure for different refresh intervals for the ad networks under study on both 3G and Wi-Fi. The

[6]To obtain power values in mW, values should be multiplied by the voltage (4V). To obtain the impact on the battery life (assuming that the app is running continuously), the battery capacity (in mAh) should be divided by the current consumption.



(a) Mobile Networks (3G)



(b) Wi-Fi

Figure 13: Current drain of AdMob, Millennial Media and InMobi for banners at different refresh intervals and type of connectivity: Mobile Networks (3G) (a) and Wi-Fi (b).

results reveal that Google's Cloud-To-Device[7] push technologies can add some energy overhead due to their encryption mechanisms. Nevertheless, independently of the technology, a low refresh interval can more than double the current consumption (if there is no background traffic) in comparison to the baseline measurements described in Section 4.2.2. Importantly, the power overhead caused by the refresh interval is more notorious on 3G than in Wi-Fi due to their characteristic power models. Nevertheless, the results might vary in a real-world scenario as other apps might generate other flows that overlap, hence the power overhead is subsidized among different flows.

Using *tcpdump*, we identified that publishers' control on the retrieval mechanism and refresh interval used are not the only source of energy inefficiency on devices. Usually, the HTTP/1.1 request created by the SDK have the KeepAlive option enabled in order to fetch several objects in the same request. Even if each ad network relies on a single content provider as reported in Figure 7, different objects are usually downloaded from different servers and not transferred on a single TCP connection. In this case the KeepAlive feature is worthless and often forces the 3G radio interface to go from IDLE or CELL_FACH to CELL_DCH. This is the case of AdMob (also observed in other Google's services such as Gmail and Google Search), in which servers and clients have a connection timeout of 6 seconds, forcing the mobile interface to remain active (or even to go from IDLE to DCH) only to send TCP's FIN notification. Figure 14 shows the throughput for AdMob and Millennial Media for 3 minutes from an offline app requesting ads every 40 seconds. We can identify small throughput peaks caused by the FIN-ACK notifications in AdMob, whereas the active close used in Millennial Media presents a cleaner figure. These small

[7]`https://developers.google.com/android/c2dm/`

peaks can immediately cause RNC transitions and energy waste on the devices. This is the main reason behind AdMob having a higher power cost in comparison to Millennial Media and InMobi, both of which tend to close the connection immediately after the transaction is completed. Nevertheless, we have seen that when the server is the one performing the active close, as found in InMobi, it often experiences *Failure to send `FIN` notification promptly*. If this happens, the client will try to close the connection with the server after 1 minute (the default `fin_wait` timeout, specified in the proc filesystem) also forcing the 3G radio interface to go to a dedicated channel (CELL_DCH) in vain as the gateway/NAT in the mobile network might have closed the connection already. Millennial Media's client and server try to simultaneously perform the active close, reducing such inefficiencies. This guarantees that most of the network usage is compressed in the shortest time interval. Interestingly, AdMob and InMobi servers identify the type of network of the client to adapt the behavior of the system accordingly. The TCP connections remain open when the phone is connected with Wi-Fi despite the memory overhead for the servers required to keep an open TCP connection.

The results shown in this section are the worst case scenario for a pure offline app that does not require any network access but for displaying mobile ads. If the app is online by nature, the power overhead caused by ad traffic can be amortized whenever the app requires network access for its normal activity. However, a more detailed evaluation of the impact of background traffic requires a good characterisation of mobile traffic for different apps which is still an open and interesting research problem by itself. In the future, proposals, such as SPDY[8] from Googlem could offer better efficiency in downloading resources on the Internet by using parallelization and flow management to outperform HTTP. SPDY might present an elegant solution to the problem we have depicted here, but it will rely mostly on the global adoption of the solution in all web servers and web browsers, something proven to be difficult with previous proposals, such as WebM[9].

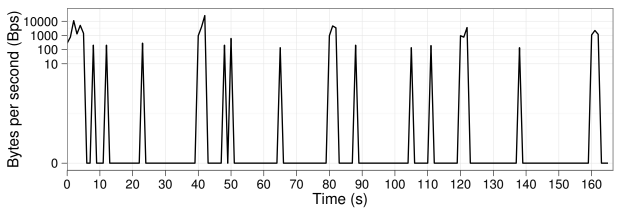
5. EVALUATING IMPROVEMENT STRATEGIES THROUGH ADCACHE

The mobile ecosystem is maintained by a model that encourages publishers to (ab)use the features provided by ad networks to increase their revenues. Ad traffic is usually found in isolation so it forces offline apps to be online in order to obtain mobile ads. As a consequence, the volume of ad traffic per user is significant; 1% of mobile users have more than 2 MB of ad traffic per day, much of which is usually composed by objects such as images and Javascript that are constantly being re-downloaded. Such high download rates cause an important energy overhead, which is also aggravated by multi-party services and an inefficient protocol design. In Section 4.2, we observed that displaying ads on an app over a period of three minutes can more than double the power consumption at the lowest refresh interval.

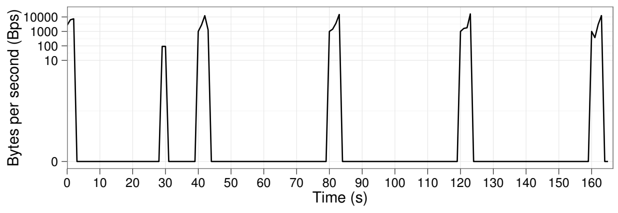
Aggressive advertising strategies have pushed some users to adopt defensive strategies, for example, millions of users have installed the AdFree ad blocker app on Android. However these solutions are not ideal as they often involve changes to the `hosts` file to redirect traffic to localhost, generating firewall rules (by `iptables`) or turning off connectivity altogether. Furthermore, they are also

[8]`http://dev.chromium.org/spdy`

[9]`http://www.appleinsider.com/articles/12/03/14/mozilla_considers_h264_video_support_after_googles_vp8_fails_to_gain_traction.html`



(a) AdMob



(b) Millennial Media

Figure 14: Throughput for AdMob (a) and Millennial Media (b) at a refresh interval of 40 seconds. We can clearly see the throughput peaks caused by the FIN notifications in AdMob, forcing the radio interface to change up to (or remain in) higher power states for longer periods of time.

a major disadvantage to publishers as it completely breaks their intended revenue model. The ideal solution is an ad delivery mechanism that is friendly to all the end-users, ad networks and publishers.

There are two feasible approaches to reduce the traffic volume and power overhead of ad networks: *i)* avoiding redundant transmissions and *ii)* reducing the number of transitions between the power modes in mobile networks. Conceptually, ad traffic contains static content by default, even for targeted ads. Mobile ads are not generated in real time, they are the result of planned campaigns by the advertisers mainly distributed over CDNs, as described in Section 3.3.

Based on these constraints, we designed *AdCache* for Android devices. The following sections will detail its architecture, describing the way it fulfils the requirements for state of the art commercial ad platforms and their features such as user profiling. AdCache is bandwidth and energy-aware by exploiting simple techniques such as connectivity awareness, batching and caching. Although privacy was not an initial objective, AdCache's design allows preserving users' privacy in a similar manner to MobiAd [5] and PrivAd [6].

5.1 Mobile agent

Figure 15 describes the AdCache architecture on a mobile device. As a proof of concept, the server functionality is provided by a mock ad network responsible for handling and serving the requests from the mobile clients. The mobile agent is a continuously running background service that prefetches ads into a data structure and serves them locally on requests by apps at the minimum power cost. The data structure is persistent (even when phone reboots), and is filled completely when the agent is initially started with a set of ads specified by the ad network and is then updated periodically. Ads are prefetched based on their Time To Live (TTL) value, defined by the ad network to remove outdated or invalid ads.

The mobile agent acts as the coordination point for delivering ads to apps installed on the device, making sure that no single app re-

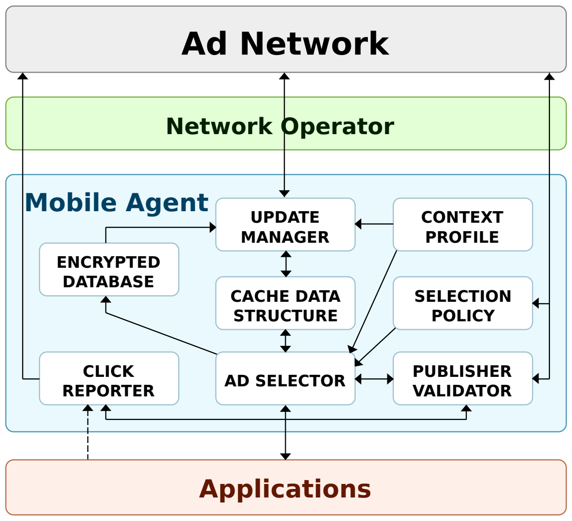


Figure 15: The AdCache architecture.

quests ads excessively and that radio access is given consciously. The ads are served according to a policy defined by the server, which is injected on the mobile agent whenever it has to be updated. Among other features, the policies assign priority to a given set of ads that takes into account the importance of some campaigns over others.

Cache updates and synchronisation tasks are event-based, triggered by satisfactory network conditions, timeouts (re-sync period set 2 hours by default but can be customized by the ad network), app requests, and expiration of all cached ads. These design decisions have three immediate effects: *i)* there are no cache misses *ii)* ads will not be pre-fetched if there are enough valid ads or there is no app demand hence reducing traffic volume and *iii)* allows AdCache to efficiently use network and energy-intense resources by exploiting batching techniques. The server also has the ability to push new ads to the client if needed.

Targeted ads

AdCache includes an opt-in profile in the mobile agent which the user is able to fully control. If they opt-in, this private information will be used by the ad network to deliver targeted ads, otherwise, standard ads will be delivered. Users' data is securely stored locally and apps do not have access to it.

For location awareness, location information is obtained using passive methods such as Android's Passive Location Provider[10] or mobile network information. In the latter case, AdCache delegates the task of performing the network positioning lookups on services such as Skyhook wireless to the server rather than adding an additional HTTP request on the device. Location is also used to update the cache. If the user is moving and there are apps requesting ads, the update interval is reduced dynamically to a value predefined by the network in order to provide more relevant ads quicker to the user if needed.

Reporting support

Statistics for the views are securely stored locally in an encrypted database. The reports are batched and uploaded to the server during cache updates rather than reporting them immediately. To avoid fraudulent actions, on receiving a message from a new publisher (based on their app package and publisher ID), the mobile agent validates the publisher's authenticity by contacting the ad network. The database is secured with tokens obtained from the ad network when starting the cache which are refreshed on each cache update. We use a dynamic token instead of a static token to avoid reverse engineering attacks. The downside of this is that if the client service is killed, the token, that had been stored in memory, is lost and the database has to be thrown away. This loss is not ideal, however it can be considered acceptable when compared to allowing an attacker (once the password is discovered) to delete and/or generate fake reports. As a consequence, user clicks are reported immediately to the server as the network interface will already be in a connected power mode when downloading the advertised content. This reduces the possibility of losing a click report if the service is stopped (unless network connectivity is not available at this time, in which case it will be cached), while also allowing AdCache to synchronize the reports and update the cache. This decision is based on the fact that click actions for AdMob (using the rule reported in Table 3) are not popular across many mobile users. Only 4.06%, 4.34% and 5.19% of the users for Android, iPhone and iPad respectively had performed at least one click on an AdMob ad. This metric should not be confused with the click-through rate, generally used to measure the success of an online advertising campaign for a particular website, and requires further investigation that we would like to explore in the future. The current consumption of performing such action on the device under study is 350 mA on 3G and 200 mA on Wi-Fi approximately.

Smart network usage

AdCache monitors the network conditions of the mobile interface (i.e., signal strength and type of network). This is done in order to temporarily defer the update if the network conditions are not ideal. In fact, the benefits of using AdCache on Wi-Fi are minor when compared to the 3G case.

When the mobile device is connected by Wi-Fi, AdCache operates like an ordinary ad network, by fetching one ad at a time as the power and network overhead is minimum. This is done so AdCache can deliver the most relevant ads to the user without any significant power cost. Nevertheless, under these network conditions, the cache attempts to prefetch fresher ads or update the metadata of existing ads which will be served once Wi-Fi connectivity is lost.

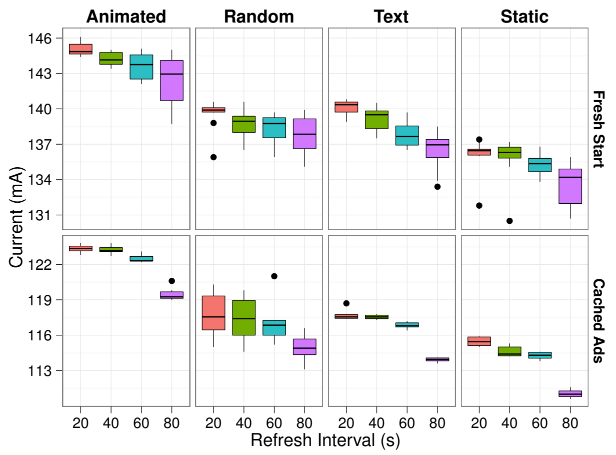
Privacy

The privacy and security issues highlighted in studies such as [1], [13] and [6] were considered. AdRisk [3] found that sensitive information accessed by some SDKs included call logs, user phone numbers and lists of all the apps a user has installed. A side effect of the AdCache design is that the permissions required for advertising are decoupled from the ones required for the intended purpose of the app. In fact, local user profiling helps to preserve user privacy as AdRisk [3] proposes.

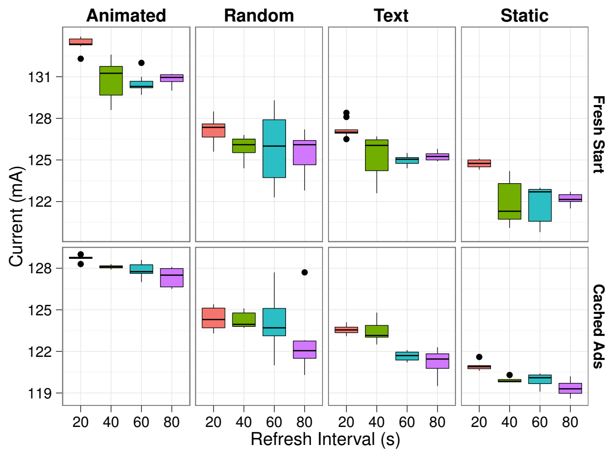
5.2 AdCache evaluation

To evaluate the power cost of using AdCache, we used the same refresh intervals from Section 4.2.1 for three different type of ad: static banner, animated banner and text ads. A fourth case in which AdCache serves a randomly chosen type was also evaluated to emulate the behavior of existing ad networks. The animated ad type was purposely built to be particularly costly in terms of CPU usage, thus we could establish an upper bound on cost of supporting such a feature. We collected two sets of results, one where the mobile

[10] Android's PassiveLocationProvider reports location changes obtained from A-GPS or network positioning systems only if another app is actively doing so.



(a) Mobile Network (3G)



(b) Wi-Fi

Figure 16: Current drain of AdCache with different refresh intervals for animated and static banner ads, text ads and random ads under different connectivity conditions: Mobile Network (3G) (a) and Wi-Fi (b).

client has an empty cache (worst case), and the second with the cache already running and with valid ads already pre-fetched (best case). On a fresh start, 15 ads (10 image-based and 5 text-based ads) were downloaded to the device from a mock server hosted in the US. The time required to download all the ads on a fresh start varies from 10 to 12 seconds on 3G, this includes the channel allocation time.

As we show in Figure 16(a), AdCache has a significantly lower average current consumption when compared to existing ad networks (Figure 13). The power overhead compared to the baseline values for minimum brightness shown in Table 6 is small as a result of batching and pre-fetching. In Table 7, we compare the current drain of AdCache with the existing ad networks under study for the minimum and maximum refresh interval on 3G. For a 20 second refresh interval, *the power consumption is halved when compared to existing ad networks over a 3 minute period.* Interestingly, the effect of the refresh interval is not as notorious as in existing ad networks (Figure 13) where it reduces the number of network

Ad Network		Refresh Interval (s)	
		20	80
AdMob	Push	242.7 (12.2)	204.6 (5.5)
	Pull	233.4 (10.6)	198.8 (12.2)
InMobi		223.7 (6.8)	178.6 (12.0)
Millennial Media		225.5 (3.7)	156.4 (6.1)
AdCache *Fresh Start*	Animated	140.1 (0.6)	142.8 (2.1)
	Random	139.5 (1.4)	137.8 (1.7)
	Text	145.1 (0.6)	136.5 (1.7)
	Static	136.1 (1.6)	133.6 (1.8)

Table 7: Comparison of the average current consumption (standard deviation in brackets) for the different ad-networks with the maximum and minimum refresh intervals on 3G and AdCache if the cache is empty (worst case scenario).

transactions. For a 1,500 mAh battery, this implies that if the app was running continuously, the battery life could be extended from 6 hours to more than 12 hours. On the other hand, if the mobile agent has prefetched ads already, the ad is displayed almost instantly at a minimal power cost as I/O and CPU cost is negligible compared to wireless interfaces. Likewise, the cost over Wi-Fi (Figure 16(b)) is also slightly improved over existing ad networks. The small variability below 10 mA on the current cost for the Wi-Fi measurements is caused by background traffic. In a real deployment scenario, the current consumption can vary between the upper and the lower bounds depending on the advertising campaigns, user mobility and the expiration time of the ads.

Data plans can have relatively low data allowances, thus AdCache is also important in monetary terms for the end user. AdCache does not repeatedly download the same objects as was found with current ad networks, and also batches activity reports. In Section 3.5.2, we saw that 1% of the user's total daily traffic is wasted as a consequence of the repeated downloads of objects. Furthermore, mobile operators can also benefit from AdCache as the client communicates with the ad network less frequently, thus the amount of signalling traffic and the implications on the scarce radio spectrum in the mobile network are likely to be reduced significantly as a consequence of minimising unnecessary HTTP requests.

6. RELATED WORK

The ad ecosystem has been subject to several research studies focusing primarily on privacy issues. MobiAd [5] and PrivAd [6] suggest local profiling and ad serving in order to protect the user privacy, using a third party as an anonymization stage. In these solutions, a profiling agent receives a large selection of ads via a third party proxy, displaying those which match the user's interests. Clicks are also sent using third parties in order to keep the anonymity of the user intact. However these solutions perform a number of CPU-intensive cryptographic operations and are likely to increase communication overhead. In [1], the authors perform a large scale crawl of the Android app market based on the metadata about required permissions. They describe the mobile ecosystem based on free mobile apps and the imbalance of privacy in the ecosystem, as many apps are free and depend on targeted ads to generate revenue [14]. In AdRisk [3] the authors found that more than half the apps on the Android platform include aggressive ad libraries that download and run code from remote servers, while accessing personal information such as call logs and installed app lists.

While a lot of effort has been taken on privacy, only a small set of work focuses on performance issues and energy. In [4] the authors analyzed the mobile traffic patterns of 43 mobile users across two different mobile platforms using packet sniffers. They find daily traffic generation patterns of users and highlight the inefficiencies caused by the generally small transfer size of packets. Both [1] and [13] suggest the need to decouple the flow of information between publishers, users and ad networks to better optimize energy consumption. Eprof [2] is the first energy profiler for smartphone apps. It found that as much as 75% of the energy consumed by free apps is spent on mobile advertising.

The characteristics of the current mobile stacks and the tied relationship with energy management have been extensively studied in the recent literature. Studies cover topics ranging from the lowest level of the 3G network stack with the radio resource allocation management [10] to the highest level of the stack with the impact of Javascript and HTML code in web pages [15] on mobile devices. Other pieces of work tried to analyze the impact of middle boxes [16], along with other contributions such as [17] and [18].

Caching has been a popular solution for constrained systems and networks, specially those with poor connectivity conditions. Caching has been proposed already as an efficient way of saving energy on mobile networks [12]. In [19], the authors aim to reduce the bandwidth cost of mobile apps by proposing an HTTP proxy-based caching mechanism. They also highlight the potential inefficiencies that can be found in terms of energy. In our work, we prove the efficiency of an independent caching system using a variety of energy measurements on 3G and Wi-Fi. AdCache also provides the ability to separate permissions between advertising and app functionality, hence enabling AdCache to be a privacy-preserving profiler and advertising system.

The works described above are usually based on active experiments performed on a set of smartphones. To the best of our knowledge, this paper presents the first in-depth analysis of the ad network and mediation services ecosystem conducted on traffic from a real network. We did not limit our analysis to a single device type or ad agency, but instead we consistently compared Android, iPhone and iPad devices. We found several inefficiencies considering the wastage of both energy and bandwidth, and we designed AdCache to limit them.

7. CONCLUSION

In this paper, we undertook the first in-depth analysis of a large mobile ad traffic data set. Our findings confirm on a large scale that: *i)* ad networks impact a large proportion of users, especially on Android, but also on iPhone and iPad; *ii)* the ad ecosystem for mobile apps is mainly dominated by Google services (e.g. AdMob, AdWhirl and Google Analytics); *iii)* ad traffic can be a significant fraction of the total traffic of the users; *iv)* mobile ad traffic is responsible for important energy and network overhead by forcing offline apps to become online apps; and *v)* many of these requests are redundant due to the lack of caching capabilities in the SDKs. By taking a closer look at the popular AdMob service, we identify that the typical session of such traffic is quite short and very similar to the demotion timeouts used by the 3GPP network technology.

We identify a clear incompatibility of the current ad distribution mechanisms with the quasi-static nature of ad content. During the analysis of the data set, we have been surprised to see the impact of the user profiling traffic. We noticed a significant number of transmissions to analytic services with similar energy and bandwidth usage issues as seen in ad networks.

We demonstrate the advantages of a caching approach through the implementation and evaluation of AdCache, a first step into reducing the impact of ad traffic on battery life and controlling the traffic generated by ad networks. We demonstrate AdCache's viability in terms of energy savings to tackle the identified issues related to ads traffic. Our future objectives go along two points, first of which, we are planning to characterize the mobile ad ecosystem in greater detail, specially from an economics perspective, taking advantage of the potential of the rule set. Second, we would like to continue researching an energy and spectrum efficient ad delivery mechanism by introducing more embedded logic, enabling more offline capabilities such as user profiling, analytics support and greater fine-grained targeted advertising.

Acknowledgements

Narseo Vallina-Rodriguez is partially funded by a Qualcomm Innovation Fellowship and Alessandro Finamore by the EU-IP project mPlane funded by the European Commission under the grant n-318627. The authors would also like to thank the anonymous reviewers and our shepherd Chen-Nee Chuah (UC Davids) for constructive feedback on preparation of the final version of this paper. We also wish to thank Steve Uhlig (Queen Mary, University of London), Vijay Erramilli (Telefonica Research), Charalampos Rotsos, Steve Hand, Ilias Leontiadis, Christos Efstratiou, and Daniele Quercia (University of Cambridge) for insightful comments and feedback.

8. REFERENCES

[1] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. Don't kill my ads!: balancing privacy in an ad-supported mobile application market. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems &#38; Applications*, HotMobile '12, pages 2:1–2:6, New York, NY, USA, 2012. ACM.

[2] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, EuroSys '11, pages 153–168, New York, NY, USA, 2011. ACM.

[3] Michael C. Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[4] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 281–287, New York, NY, USA, 2010. ACM.

[5] Hamed Haddadi, Pan Hui, and Ian Brown. Mobiad: private and scalable mobile advertising. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, MobiArch '10, pages 33–38, New York, NY, USA, 2010. ACM.

[6] Saikat Guha, Alexey Reznichenko, Kevin Tang, Hamed Haddadi, and Paul Francis. Serving ads from localhost for performance, privacy, and profit. In *In Proceedings of the 8th Workshop on Hot Topics in Networks*, 2009.

[7] Ad Regex Dictionary. `http://www.retitlc.polito.it/finamore/mobileAdRegexDictionary.xlsx`, 2012.

[8] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement*

conference, IMC '11, pages 329–344, New York, NY, USA, 2011. ACM.

[9] Feng Qian, Kee Shen Quah, Junxian Huang, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Web caching on smartphones: ideal vs. reality. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 127–140, New York, NY, USA, 2012. ACM.

[10] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Characterizing radio resource allocation for 3G networks. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 137–150, New York, NY, USA, 2010. ACM.

[11] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association.

[12] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, pages 280–293, New York, NY, USA, 2009. ACM.

[13] Paul Pearce, Adrienne Porter Felt, Gabriel Nunez, and David Wagner. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of ACM Symposium on Information, Computer and Communications Security*, AsiaCCS'12. ACM, 2012.

[14] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–6, Berkeley, CA, USA, 2010. USENIX Association.

[15] Narendran Thiagarajan, Gaurav Aggarwal, Angela Nicoara, Dan Boneh, and Jatinder P. Singh. Who killed my battery?: analyzing mobile browser energy consumption. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 41–50, New York, NY, USA, April 2012. ACM.

[16] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Zhuoqing Mao, and Ming Zhang. An untold story of middleboxes in cellular networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):374–385, August 2011.

[17] Narseo Vallina-Rodriguez and Jon Crowcroft. Energy management techniques in modern smartphones. *Communications Surveys Tutorials, IEEE*, PP, 2012.

[18] H. Haverinen, J. Siren, and P. Eronen. Energy consumption of always-on applications in wcdma networks. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 964 –968, april 2007.

[19] Azeem J. Khan, V. Subbaraju, Archan Misra, and Srinivasan Seshan. Mitigating the true cost of advertisement-supported ”free” mobile applications. In *HotMobile'12*, 2012.