The wireless data drain of users, apps, & platforms

Ardalan Amiri Sani[†], Zhiyong Tan[†], Peter Washington[†], Mira Chen[†], Sharad Agarwal[‡], Lin Zhong[†], Ming Zhang[‡]
[†]Rice University, [‡]Microsoft Research
[†]{ardalan, zt1, pyw1, sc25, lzhong}@rice.edu, [‡]{sagarwal, mzh}@microsoft.com

Cellular data consumption is an important issue for users and network operators. However, little is understood about data consumption differences between similar apps, smartphone platforms, and different classes of users. We examine data consumption behavior in the lab, comparing different apps of the same category, comparing the same top apps across different platforms, and comparing network APIs that apps use across different platforms. We also collect data from 387 Android users in India, where users pay for cellular data consumed, with little prevalence of unlimited data plans. Our findings can inform users on how their choice of platform and apps has a drastic impact on their data bill. Our findings can also inform operators on how to use incentives to induce desired data consumption.

I. Introduction

On cellular networks, smartphone data usage can easily outstrip available capacity. Cellular operators tend to deal with this problem by applying economic pressure on users. Unlimited data plans are being replaced with either tiered data plans or "unlimited" plans that throttle data rates beyond a certain usage limit, both inside [1], [2], [3] and outside [4], [5] the US. Technologists are attempting to deal with this problem by improving cellular standards, but user demand could outpace such improvements and Shannon's limit could hold them back [4].

Beyond a plethora of market research studies that look at cellular traffic growth curves and project the rise of video streaming and other bandwidth hogging apps, there is little published work that studies the consumption problem in detail. How are smartphone users different in their data usage? How much of heavy data consumption is a result of picking one app over another? How much does the choice of underlying smartphone OS contribute to this problem? How can answers to such questions impact the way in which cellular operators sell data plans to users?

In this paper, we explore this problem from multiple angles. We conduct a number of lab experiments to compare different apps and different smartphone platforms. We examine top apps that provide the same functionality (weather forecasts). We examine popular apps written by professional developers that are available on all of the three major smartphone platforms. We exercise individual network APIs on these three platforms. Data that we collect from 387 Android users in India helps us understand user behavior across different apps. Our findings include:

- There is tremendous difference in how much wireless data is consumed by each launch of an app. This can range from 7KB to over 1MB for simply getting the current weather forecast. This huge difference is present across the top weather apps on all three smartphone platforms.
- There are several popular apps that are common to all three platforms and authored by the same professional developers – e.g. Amazon by Amazon Inc., and Netflix by Netflix Inc. Even when comparing these apps that provide the same functionality across the three platforms, we see large differences in consumption, with factors such as 5X and 10X depending on the app.
- While the basic APIs for consuming content from the network are similar across the three platforms, there are subtle differences. Compression and caching behavior is different, and maps consume different amounts of data.
- Among Android users, there are significant differences between "light" data users and "heavy" data users in terms of which apps they use and how they spend their time and data in these apps. Our findings suggest that given such information about the user, we could predict with 58% accuracy which of three data usage categories they

[1] http://www.dslreports.com/shownews/ATT-Kills-The-30-Unlimited-Data-Plan-108703
[2] http://www.dslreports.com/shownews/TMobile-Kills-Unlimited-Data-for-Smartphones-114341
[3] http://www.huffingtonpost.com/2011/07/05/verizon-unlimited-data-plan-july-7_n_890613.html
[4] http://www.fiercewireless.com/europe/story/analysys-mason-truly-unlimited-mobile-data-plans-are-rarity-western-europe/2013-05-17
[5] http://online.wsj.com/article/SB10001424052970204026804577097703931127104.html

fall into, while random guessing would achieve only 33%.

Our findings have significant monetary implications for users. Data-conscious users have to carefully pick apps and platforms. Picking the wrong weather app can result in 140X higher data cost. A user that primarily runs Amazon, Facebook, Fandango, GasBuddy, and Walgreens with equal frequency can consume 3.3X more data on the worst of the three smartphone platforms. We can construct other scenarios with more horrifying overconsumption.

As a network operator that wants to limit spectrum usage by phones, it is important to understand the user. When targeting a user that is a shopper, such as one that relies on Amazon, Fandango, and Walgreens, it is best to steer the user to a particular platform. When targeting a user that is a news and restaurant reviews fan, a different platform is lighter. An operator can stereotype the user by asking a set of questions about their behavior. Based on such classification, the operator could sell the user a tailored data plan, or offer incentives to use or stay away from certain apps and smartphones. Operators should offer free Wi-Fi to users as usage of it is mildly correlated with lower cellular consumption.

II. Methodology

We rely on two sets of experiments in this paper – lab and field. Lab experiments allow us to evaluate data consumption while controlling apps, APIs, and platforms. However, they do not capture user behavior. In our field experiments, we collect data directly from users' phones, instead of collecting network traces from cellular operators. Instrumenting phones has the advantage that we can observe Wi-Fi usage, user time in apps, and attribute consumption to the exact app. While it limits collection to those users that install our software, it allows us to target multiple operators and specific geo-regions with more ease. In all our experiments, we measure the total data usage by each app. Therefore, the data the app uses for analytics and ads is attributed to the app itself.

II.A. Lab experiments

II.A.1. Network setup

There are three components in our network setup – smartphones, a laptop, and the Rice University Ethernet network. The Rice network does not use any web proxies. The laptop runs Microsoft Windows 7 and shares its Ethernet Internet connection to smartphones over Wi-Fi using Connectify-Me [6]. The laptop also has Wireshark 1.8.4 [7] and Fiddler 2 [8] installed. We use Wireshark to inspect traffic to/from our smartphones. For in-depth HTTP analysis, we use Fiddler.

When analyzing traces, we calculate several statistics. We calculate the total number of bytes transferred, which includes IP headers (we use IPv4 in our experiments). We split the bytes transferred into four categories – video, image, text, and other. The video and image categories show the bytes for any HTTP transfer, where the "Content-Type" HTTP header field indicates a MIME type for video or image, respectively. The text category includes both "text" and "application" MIME types, the latter typically including XML and Javascript. The "other" category includes other MIME types and non-HTTP traffic such as other TCP traffic, HTTPS, or UDP. We also categorize HTTP traffic by cache and compression options. For caching, a client request can specify specific HTTP headers that will skip stale cached data in the network. The server can include specific headers in the response that will indicate to proxies and the client not to cache the response. Similarly, there are headers where the client can indicate that it will accept compressed content, and the server can specify that the response is compressed.

We use three phones in our experiments – iPhone 4 running iOS 5.1.1, Nokia Lumia 710 running Windows Phone (WP) 7.5, and Samsung Galaxy Nexus running Android 4.1.1. We anonymize the identity of the smartphone platforms when reporting results in this paper [9]. When capturing traces from one phone, we make sure the other two phones are disconnected. We evaluate a number of commercial apps and a few custom apps we wrote, that we describe next. Prior to collecting traces, for each phone, we first installed all the apps, and used each app twice. We then do not use the phones for a period of 1 week. We turn off all background activity on the phone, including app background activity, email sync, and OS updates. We then capture traces for each app and each phone individually by running the app to do a specific task and then exiting the app. We run each app twice in immediate succession, to observe how much data is presumably cached between successive runs. We capture separate traces for each execution of an app, starting the trace capture just before we launch the app, and stopping

[6] http://www.connectify.me
[7] http://www.wireshark.org
[8] http://fiddler2.com
[9] The identities of the platforms are not central to our conclusions, and indeed the platforms' behavior may change with new OS updates.

platform 1	platform 2	platform 3
Weather Channel	Weather Channel	Weather Channel
WeatherBug	WeatherBug	WeatherBug
AccuWeather	AccuWeather	AccuWeather
Weather Underground	Weather Underground	Weather Underground
Weather Eye	MyRadar	Weather View
Go Weather	Weather Free	WeatherDuck
Go Weather EX	Farenheit	Weather4Me
Yahoo! Weather	Weather+	Weather+
1Weather	Weather	Weather
wetter.com	Weather HD 2 Free	WeatherForecast USA

Table 1: The weather apps we evaluated on each of the three smartphone platforms.

app	action
Amazon	search for Kindle Fire; click on top item
CNN	click on top news story
Facebook	click on news feed for user
Fandango	click on top movie; get showtimes
Fragger	play tutorial; finish first level
GasBuddy	click on map of gas prices and nearest gas station
Walgreens	search for nearest store; click on store details
Weather Channel	click on weather and forecast
Yelp	search for pizza; click on top item
YouTube	search for & watch "Simon's Cat in Springtime"

Table 2: The popular apps we evaluated on each of the three smartphone platforms and the action we performed in each run of the app.

the trace capture just after exiting the app. In this fashion, we believe we are capturing the "typical" network consumption of these apps, as opposed to first-run behavior which may include one-time downloads of static icons or configuration.

II.A.2. Commercial apps

We evaluate two sets of apps. The first set is the top 10 apps from the "Weather" category of each of the three smartphone marketplaces. We picked only free apps, and skipped those that had specialized functionality (e.g. ski forecast). The apps are listed in Table 1 and we used the versions offered by the respective smartphone marketplaces on October 1st, 2012. We configured each app to report the weather for Houston, TX, where we ran the apps. We ran each app only so far as to get the current weather and the forecast and then we quit the app.

While those set of apps provide very specific functionality, we also evaluated a broader set of popular apps. We started with the top 50 free apps on each of the three smartphone platforms. We then picked those apps that are available on all three lists and are authored by the same company that owns the brand (e.g. Amazon by Amazon.com Inc.). We eliminated apps where it would be difficult to perform the same action twice on all three phones – e.g. VoIP apps where it would be hard to ensure the exact same audio is being transmitted, or multiplayer games where we cannot do the same game play with the same players. We then use the top 10 apps that remain on our list and install the 10 apps on each of the three smartphones. The list of these apps is in Table 2, as well as the action we performed in the app each time we ran it. For each action, we waited until the app finished it (e.g. finished loading the news story, or finished playing the video). We ran the same app on each three phones close enough in time so that the same content is shown to the user.

task	description
audio	play 5,757,568 B audio from http://ardalan.recg.rice.edu/audio.mp3
show map	centered at lat 29.94, lon -95.26 Google maps on iPhone, Android; Bing on WP
socket	open TCP socket to and receive 1,024 B from ardalan.recg.rice.edu:5661
text	show 139,878 B text file from http://ardalan.recg.rice.edu/mira/pride_and_prejudice
video	play 7,883,627 B video from http://ardalan.recg.rice.edu/video.3gp
web page	show web page that is 51,452 B in total http://www.owlnet.rice.edu/~aa15/

Table 3: Specific tasks that our custom apps perform on each of the three smartphones.

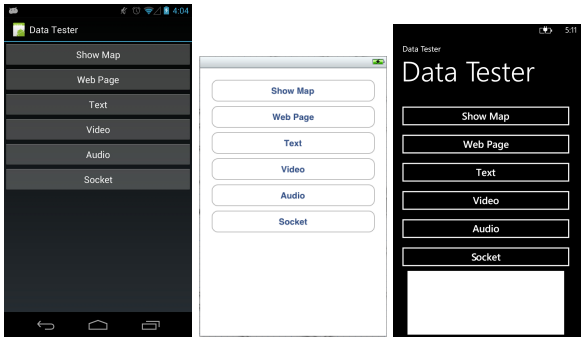

Figure 1: Screenshots of our custom apps on the three smartphones.

II.A.3. Custom apps

We also evaluate specific APIs that consume network bytes, by building three custom apps. Figure 1 shows screenshots for the apps and Table 3 lists the actions that each of the 6 buttons do. For maps, we use the native maps APIs that each platform offers to display a specific 1 square mile region in Houston, TX. For the web page and text file downloads, we use standard HTTP APIs in the three platforms and use the default options – i.e. we did not explicitly set any caching or compression options. On Android, we used the "An-

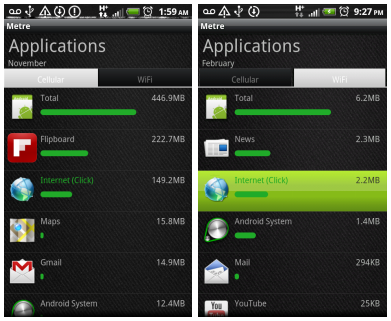

Figure 2: Screenshots of Metre UI on Android.

droidHttpClient" API. For video and audio playback, we used APIs that allow us to use URLs as the source.

II.B. Field experiments

Our lab experiments are useful for comparing different apps and platforms directly since we control what actions are performed in each app. However, there are three main limitations of our lab experiments. All the experiments are done on Wi-Fi, and it is possible that some apps may behave differently on cellular. They do not reflect network consumption by real users. They only measure foreground activity of apps, not their periodic background activity. To overcome these limitations and to understand user behavior in the wild, we deployed an app on Android.

II.B.1. Metre: data collection app

We developed an app called *Metre* for Android smartphones. The data usage of each app on Wi-Fi and cellular is measured and shown to the user, as in Figure 2. The user can select three time periods to see usage over – the current month, week, or day. For each app, Metre collects the following seven pieces of data per day: bytes sent and bytes received on each of Wi-Fi, cellular, and USB, and also the app foreground time. The app foreground time measures the time that the app is visible on the screen. At the end of every day, Metre attempts to upload that day's data (typically a few KB in size) to a server in the cloud. This data is accompanied by configuration information – the phone model, the network operator name, Android OS version, and the current time zone. If the upload fails, Metre will queue it and retry periodically.

To measure data consumption, Metre uses two Android APIs – `TrafficStats.getUidRxBytes(appID)` and `TrafficStats.getUidTxBytes(appID)`. Since these APIs do not distinguish between cellular, Wi-Fi, and USB consumption, Metre registers for notifications for network interface changes, calls the TrafficStats APIs, and attributes consumption appropriately. Since the phone uses one interface at a time, this approach is accurate in assigning the data usage to different interfaces, catching even short transitions between them. When an app uses the Android media streaming API, Android assigns that network consumption to the Android media service rather than to the app. In Metre, we monitor which app is in the foreground and appropriately re-assign those bytes to the responsible app.

task	Metre usage	Packet traces
audio	6,149,322 B	6,582,117 B
show map	91,008 B	96,330 B
socket	1,138 B	1,778 B
text	140,418 B	152,693 B
video	8,254,200 B	8,274,370 B
web page	44,090 B	46,125 B

Table 4: Metre accuracy tests.

Metre's accuracy depends on the accuracy of the underlying Android APIs. Unfortunately, those APIs do not include TCP/UDP/IP packet header overheads, retransmissions, nor DNS lookups. To quantify this inaccuracy, we present Table 4 where we compare the size of the specific network transfers in Table 3, as reported by packet traces and as reported by Metre. We feel that this difference, due to header overheads and DNS overheads, does not alter the findings from our comparative analysis.

II.B.2. User population

We deployed Metre in Google Play in February 2012 [10]. Based on users' feedback and early data analysis, we made bug fixes and performance improvements and released a major update. We finally retired the app in May 2013.

The US population is undergoing a shift from wide availability of unlimited cellular data plans (29% of US subscribers had unlimited data in December 2010 [1]) to strict caps or throttling. To avoid the impact of usage limit rate throttling on our analysis, we avoid US subscribers. Instead, we focus on the country of India, where operators tend to charge by actual data consumed [11] and unlimited plans are comparatively much more expensive and rarely used.

Android 2.2 and 2.3 together account for 60.9% of the market [12] even as of December 2012, and we be-

[10] covered under Rice University IRB approval #12-098X

[11] http://newamerica.net/publications/policy/an_international_comparison_of_cell_phone_plans_and_prices

[12] http://developer.android.com/about/dashboards/

lieve these older versions are used predominantly in India and other emerging markets. Hence we use the 2.2 and 2.3 versions of the relevant APIs to collect data consumption, and tuned our app's performance and stability for these versions. To increase participation, we advertised Metre through the AdMob mobile advertising network, specifically targeting India.

In the 2 months since the last update to our app, we have collected data for over 600 users. After filtering out non-Indian users (using time-zone and operator name), and those with fewer than 5 days of records, we have 387 users from 2012-10-01 to 2012-12-03.

III. Lab Results

We now present results from running apps in the lab. We designed these experiments to not necessarily be representative of typical user consumption, but instead to directly compare apps and platforms.

III.A. Weather apps

There are a lot of apps in each of the three smartphone marketplaces. We obviously expect some apps to consume more wireless data than others, such as a video streaming app versus a calculator app. However, we expect apps with similar functionality to consume roughly the same number of bytes. We test this hypothesis by picking weather apps. Unlike other categories such as "utilities" or "entertainment", we expect weather apps to provide similar, generally unambiguous features – weather forecasts for your location.

In Figure 3, we show the total data consumption of weather apps on each of the three platforms. Each bar is color-coded to show the size of the content types that were downloaded, as described in §II. There is a tremendous difference in how many bytes weather apps consume. One app consumes as little as 7KB in a launch, while others consume over 1MB. This is a staggering difference of over 140X. Neither extreme is a single data point – there are other apps that have similarly low data consumption, and others with similarly high consumption, on all three platforms.

These findings are surprising to us for several reasons. We ran each app simply to the point where we see the current weather and the weather forecast. We did not explore any possibly additional features of an app, such as high-definition video streams of tornadoes. Hence we expected the majority of network downloads to be simply the text of weather conditions and forecasts, which are small and highly compressible. However, as the figures show, the text fraction of the download varies.

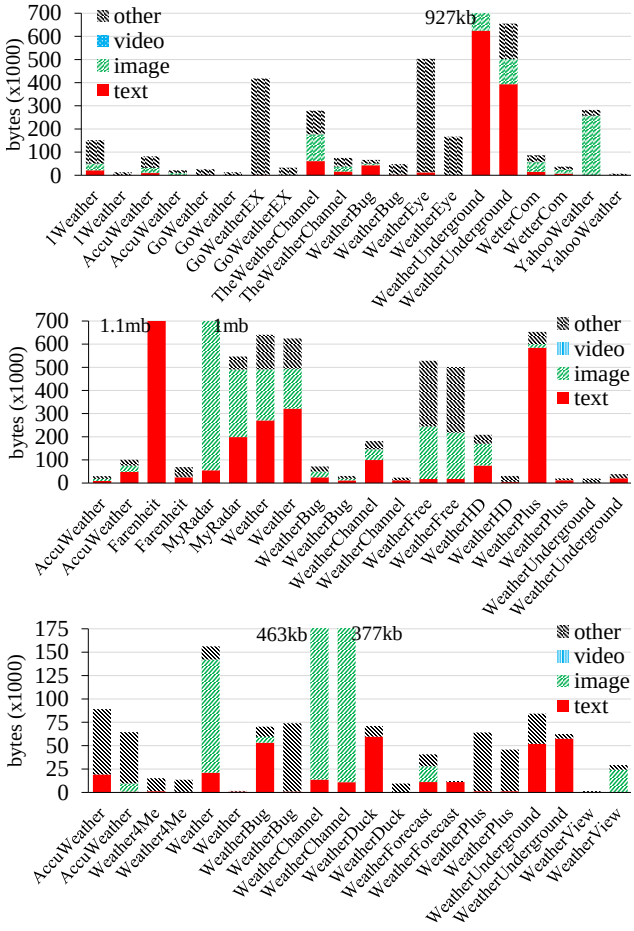

Figure 3: Data consumption by a variety of popular, free weather apps on platform 1 (top), platform 2 (middle), and platform 3 (bottom). Each app was used to request weather for Houston, TX. Each app was run twice in immediate succession and both values are plotted. The vertical axes are chopped, and lines exceeding that are labeled with the total transfer size.

To avoid including any extra transfers as a result of first time setup, we ran each app twice, followed by a week of not using the app, and then ran the app again twice to capture traces for these results. Hence we do not expect these results to have variability from some apps downloading logos and icons on first launch, and other apps including such content in the app install package itself. With 4 exceptions, in almost all cases the second run of the app consumes fewer bytes to varying degrees. This is presumably due to implicit caching of HTTP content by the underlying OS, or explicit caching by the app itself.

There are interesting implications for the user. The data-conscious user has to be vigilant even when choosing between weather apps – a type of app that

`index.html`

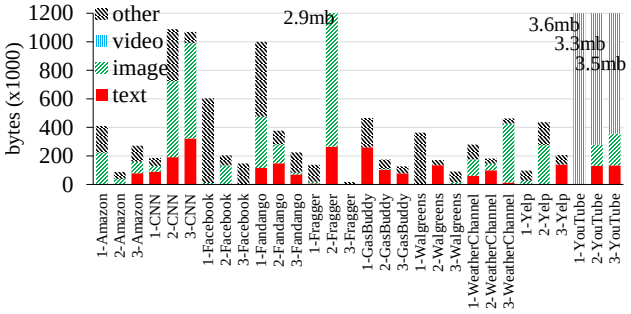

Figure 4: Data consumption by a variety of popular apps on the three platforms. Each app was written by the same company for all three platforms. The vertical axis is chopped for clarity, and lines exceeding that are labeled with the total transfer size. The name of the app is prefaced with the platform number. The results are from two runs of each app. These are not the first runs of the app after install, but rather runs after a week of no use.

one may otherwise not think would be a data hog. An OS platform or cellular operator may want to factor in the data consumption of apps before promoting one app over another in the marketplace.

III.B. Popular apps

There remain several questions. Why is there such a big difference in consumption between apps? Are some apps providing more functionality? Are any of the platforms inherently more data consuming? Are some apps not using standard cache controls or compressing network transfers? To understand these in more depth, we now examine top apps that we can directly compare across the three different platforms.

III.B.1. Differences between platforms

Figure 4 shows the data consumption of these apps across the three platforms. As before, each app has been run before, followed by a week of no use, and then run twice while collecting packet traces. For brevity, we show the first of those latter two runs.

To a large extent, we expect the functionality of the same app by the same author across all three platforms to be the same. Yet, we see a significant variation in network consumption. For example, the CNN app consumes 186KB on one platform but over 1MB on the other two. Fandango consumes almost 1MB on one platform, but 224KB and 377KB on the other two. Fragger on platform 2 consumes a huge number of bytes, and most of the bytes are spent on high resolution sprite images that the app does not appear to cache between runs. The only app with almost similar consumption is YouTube, which is between 3.3MB and 3.6MB on the three platforms. This unexpected finding is not simply explained away by the platform – no one platform is consistently lighter than another.

As a user, this can have a major impact on my data consumption depending on what apps I rely on. Suppose a user primarily uses Amazon, Facebook, Fandango, GasBuddy, and Walgreens, and uses each of them with the same frequency and performing the same operations as we did. If the user picks platform 1, his data consumption will be 2.8X worse than if he had picked platform 2, and 3.3X worse than if he had picked platform 3. A different combination of apps and different frequency of use may point to a different platform as the worst offender. Users have little insight today on the impact of the choice of smartphone platform on the cost of their data plan.

A cellular operator that sells all three smartphone platforms may choose to adjust their marketing based on such information. Advertising campaigns often target specific market demographics (e.g. the social user, or the gamer, or the movie fan), and an operator may want to use a different platform to push to each demographic. An operator that wants users to conserve data may pick the lightest platform for each demographic. Conversely an operator that wants to make the most money from overage charges will want to pick the worst platform for each demographic.

III.B.2. Differences at the HTTP layer

In these apps, the majority of network data is transferred over HTTP. There is little use of UDP, perhaps because we did not test a VoIP app such as Skype. Since the majority of traffic is HTTP, we can examine the network caching and compression behavior of these apps. Table 5 shows the percentage of HTTP transfer bytes in each app where either the client or the server explicitly set a no-cache directive, or either the client did not specify it can accept compressed content or the server did not compress the response.

Since we are examining traces for a repeated run of an app, we expect the transferred bytes to be dominated by dynamic content, such as the latest news or movies list. Interestingly, only in a relatively small number of cases is the app explicitly specifying a no-cache primitive in HTTP requests. Doing so forces the HTTP request to hit the network. It is possible that most app developers are being network friendly. Alternatively, most app developers may simply be lazy and not write the additional line of code to specify a cache header in the HTTP request. Due to this behav-

app	% of bytes where			
	no-cache by		no-compress by	
	client	server	client	server
Platform 1				
Amazon	0	0	1	100
CNN	0	5	100	100
Facebook	0	0	0	98
Fandango	0	0	97	100
Fragger	0	1	2	100
GasBuddy	0	0	54	95
Walgreens	0	0	100	100
Weather Channel	0	1	0	70
Yelp	0	0	0	68
YouTube	0	29	1	62
Platform 2				
Amazon	85	0	0	91
CNN	98	2	0	67
Facebook	4	10	1	80
Fandango	37	4	0	65
Fragger	94	0	0	100
GasBuddy	0	0	0	98
Walgreens	0	0	0	100
Weather Channel	0	0	0	91
Yelp	39	1	0	97
YouTube	0	4	87	94
Platform 3				
Amazon	52	0	52	100
CNN	0	0	24	100
Facebook	0	0	55	100
Fandango	1	1	100	100
Fragger	93	93	7	97
GasBuddy	12	0	84	84
Walgreens	1	0	100	100
Weather Channel	0	0	0	96
Yelp	0	0	49	100
YouTube	89	4	90	96

Table 5: HTTP caching and compression statistics for professional apps on three platforms

ior, we see a significant reduction in the number of bytes transferred for most of these apps if we immediately run them again.

Desktop browsers accept compressed content, which increases download speed by reducing the number of transferred bytes from the server. However, smartphone platform support for it is mixed. On WP 7.x, the underlying platform does not, but app developers may add that header to outgoing HTTP requests and use third-party libraries to decompress the response [13]. On iPhone, the HTTP API [14] supports decompressing content that has been compressed by the server, but there is confusion out there on whether the iPhone API specifies the compression option by default to outgoing requests [15]. On Android, if the app developer uses the "HttpURLConnection" API, gzip encoding is on by default [16]. If the app developer uses "AndroidHttpClient" instead, it is off by default and the "modifyRequestToAcceptGzipResponse" method has to be called to enable it [17].

Hence it is not surprising that for the outgoing HTTP requests from these apps, some specifically identify that they can accept compressed content, while others do not. However, what is surprising is that despite some use of the compress header in the client requests, almost all of the server responses are not compressed. Some of this behavior is explained by the type of content – video and image content will typically already be compressed, so the server may have rules not to apply additional compression on top of that. However, as Figure 4 shows, there is still a significant fraction of text content downloaded by many apps. We believe that most of this text content is dynamically generated content from the server. To compress such content, web server administrators have to enable it. In dynamic compression, any dynamically generated content will first be passed through gzip before going over the network. Administrators are sometimes cautioned [18] against enabling it for fear of increasing per-request server processing time, thereby limiting scalability.

Our recommendation to app developers is to take the effort to add the appropriate headers and/or libraries to turn the compression on in the app, and enable it on web servers for dynamic content. For example, on platform 1, the CNN app transfers 48% of total bytes as uncompressed text. On platform 2, the Walgreens app transfers 79% as uncompressed text. On platform 3, the Yelp app transfers 67% as uncompressed text. Plain text, XML, JSON and HTML are highly compressible, and achieving even a decent compression savings of 60% can cut the total downloaded bytes in half for some apps.

III.C. Custom apps

As we have just shown, the choice of app and platform can have a big impact on data consumption. In at least one case, the choice of API can also impact data consumption because of the presence or lack of HTTP compression. Do any of the other network-consuming APIs behave differently between the three platforms? We now examine these APIs via our custom apps that

[13] http://www.sharpgis.net/post/2011/08/28/GZIP-Compressed-Web-Requests-in-WP7-Take-2.aspx

[14] http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSURLConnection_Class/Reference/Reference.html

[15] http://www.bigevilempire.com/codelog/entry/nsurlconnection-with-gzip/

[16] http://developer.android.com/reference/java/net/HttpURLConnection.html

[17] http://developer.android.com/reference/android/net/http/AndroidHttpClient.html

[18] http://technet.microsoft.com/en-us/library/cc753681(v=ws.10).aspx

app	platform 1 bytes	platform 2 bytes	platform 3 bytes
Audio	6,582,117	6,052,078	5,984,775
Audio	6,609,732	6,032,148	6,052,678
Map	96,330	3,696,819	373,633
Map	27,045	221,603	7,733
Socket	1,778	1,853	1,564
Socket	1,636	1,448	1,468
Text	152,693	53,086	145,721
Text	147,941	0	730
Video	8,274,370	8,373,289	8,234,287
Video	8,266,237	8,419,384	8,211,878
Web page	46,125	45,140	45,197
Web page	4,756	4,457	4,419

Table 6: Bytes consumed when exercising six common APIs across the three smartphone platforms. Each network operation is done twice, and listed in order.

we described in § II.A.3

Table 6 lists the bytes consumed. In general, the numbers roughly match the sizes of objects that we transfer in Table 3. Platform 2 downloads an egregiously large number of bytes (over 3MB) for maps, while the other two platforms consume about a tenth of that. We suspect that the platform is prefetching surrounding map tiles in case the user scrolls around. Platform 3 is particularly good about caching previously downloaded map tiles. The socket transfer numbers are unsurprisingly similar to the transfer size – there is little room for overhead beyond standard packet headers and DNS lookups. When transferring the text file, we use standard HTTP APIs with default options on all three platforms – the one we use on platform 2 sets a compression header by default and hence its download size is lower. Platforms 2 and 3 do a good job of caching the previously downloaded result. There is some overhead in the video transfer, but that is largely due to headers. The web page transfers show similar numbers. The HTML part of the web page is likely being transferred as compressed content because the browser itself is embedded in the app.

III.D. Summary

Our lab experiments have quantified a number of differences in data consumption that have implications for users, app developers, OS developers, and cellular operators. There are subtle differences between APIs that the three major platforms expose that can add up to a significant number of bytes. Support for HTTP content compression is mixed and can have a big impact on text transfers. Apps that make heavy use of maps APIs should understand how many bytes the OS is transferring underneath. Lack of caching of maps and HTTP content between repeated runs of an app can also increase consumption.

These differences add up in apps that use multiple network API calls to build their user experience. Even apps that provide a "simple" function of weather forecasts can have 140X difference in consumption. Depending on which apps a user runs and how often, picking the wrong platform can cost the user.

While these experiments are useful for examining apps, platforms, and APIs in isolation, they miss the user behavior aspect of the problem. We now look at real users and their consumption patterns.

IV. Results from Field Data

We now examine the data we collected from Android users. The basic question we seek to answer is: how is data consumed in the field? We break this down into more specific questions:

- Is there a temporal pattern of data usage?
- How do various apps contribute to data usage?
- How are heavy and light data users different?
- What factors can indicate that a user is a heavy or light data user?

Answers to these questions will not only help cellular operators price data plans but also provide insights for developers to promote their apps.

IV.A. Data set and analysis methods

We focus on data from Metre from September 30 to December 3, 2012. We only use the data from 387 smartphones from India for which we have at least five days of data. The median duration of data per user is 25 days, the median number of apps that ran on a phone is 50, and the median amount of data used per user is 16 MB/day. Mathematically, the data set can be described as $\{u(i,j,d) : i \in \mathbb{P}, j \in \mathbb{A}, d \in \mathbb{D}_i\}$ where $\mathbb{P}$ is the set of users, $\mathbb{D}_i$ the set of days in which data from user $i \in \mathbb{P}$ is collected, and $\mathbb{A}$ the set of apps observed. And $u(i,j,d)$ can denote any one of the following types of use: time duration, cellular data, Wi-Fi data, and total data. Table 7 summarizes the mathematical notations we use in the rest of the paper.

Since a user could install and uninstall Metre on any day, we have different numbers of days of data for each user. Figure 5 shows the number of users we have on each day. Figure 6 shows a CDF of how many days of data we have across these users, or $|\mathbb{D}_i|$.

App Usage: Our users used a diverse set of apps. We show the data consumption of the 10 most popular apps as a CDF across all users in Figure 7. This figure conflates two factors – how much data an app consumes and how much time a user spends in the app.

symbol	description
$\mathbb{D}$	set of days of data collection
$\mathbb{P}$	set of users of data collection
$\mathbb{D}_i$	set of days for which we have data from user $i \in \mathbb{P}$
$\mathbb{A}$	set of apps observed
$u(i,j,d)$	use* by user $i \in \mathbb{P}$ through app $j \in \mathbb{A}$ on day $d \in \mathbb{D}_i$
$v(i,j)$	daily average use by user $i \in \mathbb{P}$ through app $j \in \mathbb{A}$ during $\mathbb{D}_i$. $v(i,j) = \frac{1}{\lvert\mathbb{D}_i\rvert} \sum_{d \in \mathbb{D}_i} u(i,j,d)$
$w(i,d)$	use by user $i \in \mathbb{P}$ on day $d \in \mathbb{D}_i$. $w(i,d) = \sum_{j \in \mathbb{A}} u(i,j,d)$
$x(i)$	daily average use by user $i \in \mathbb{P}$ during $\mathbb{D}_i$. $x(i) = \sum_{j \in \mathbb{A}} v(i,j) = \frac{1}{\lvert\mathbb{D}_i\rvert} \sum_{d \in \mathbb{D}_i} w(i,d)$
$y(j)$	daily average use by app j across all users. $y(j) = \sum_{i \in \mathbb{P}} v(i,j)$

Table 7: Important notations. **use* can be either time duration, cellular data, Wi-Fi data, or total data.

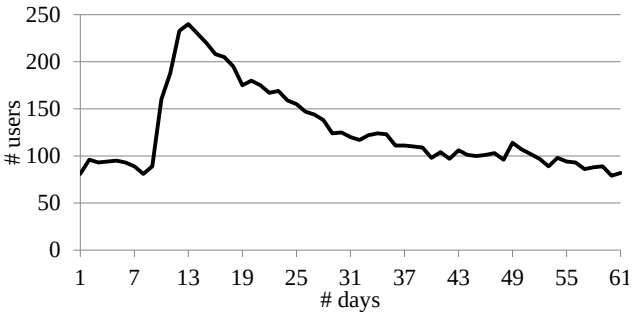

Figure 5: Number of users per day during the two months of data collection.

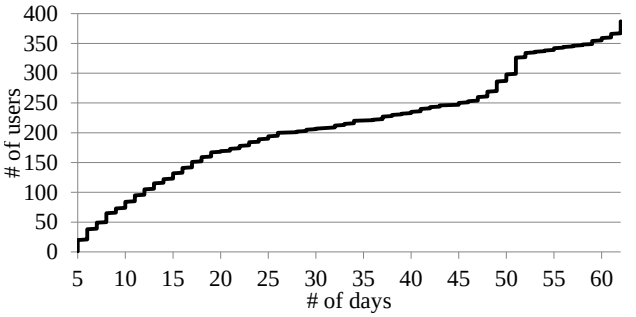

Figure 6: CDF of number of days of data collected across all users. We ignore users under 5 days.

As a result, while we would expect YouTube to be the right-most curve in this graph, it is actually in the middle, in part because users spend more aggregate time in apps such as the browser and Facebook. In the next subsection, we tease these factors apart.

Smartphone Models: The data includes at least 121 models of smartphones. Table 8 lists the characteristics of the top 10 devices by number of users.

Network Access: 32% of users used Wi-Fi for data

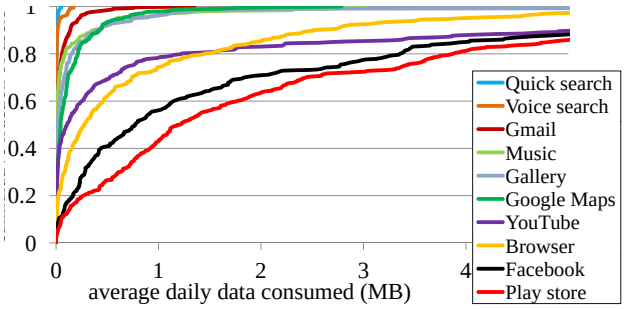

Figure 7: CDF of average bytes consumed per day for 10 most popular apps across all users. Legend lists lines in order from top to bottom.

phone model	screen		# of
	size (″)	resolution	users
Samsung GT-S5360	3	240x320	180
Samsung GT-S6102	3.14	240x320	141
Samsung GT-S5830i	3.5	320x420	72
Samsung GT-S6802	3.5	320x420	51
Samsung GT-S5300	2.8	240x320	37
Samsung GT-S7500	3.6	320x480	27
Samsung GT-S5830	3.5	320x480	26
Samsung GT-S5570	3.14	240x320	24
Samsung GT-S5670	3.3	240x320	20
Sony Ericsson ST25i	3.5	480x854	19
HTC Explorer A310e	3.2	320x480	18
Samsung GT-S5302	2.8	240x320	17
Samsung GT-I9100	4.3	480x800	17
Sony Ericsson WT19i	3.2	320x480	13
Samsung GT-I9003	4.0	480x800	12
Micromax A75	3.75	320x480	11
Samsung GT-I9070	4.0	480x800	10

Table 8: Characteristics of the most popular phones.

at least once during the data collection. All but 2 users have used cellular data at least once. 5% of them used Wi-Fi every day for which they used any data.

Data Normalization: To compare users and compare apps, we use the average daily data consumption by either users, apps, or both. This removes the bias from unequal number of days recorded from each user, and we only consider users with at least 5 days of data. We use three types of daily averages, as described in in Table 7: $v(i,j)$ and $x(i)$ and $y(j)$. When we consider top apps, either by data consumption or time spent, we only consider those apps that had at least 10 users and at least 10 days of usage across all users, to further eliminate outlier bias.

IV.B. Temporal pattern of data use

Our data set may be too small to draw definitive conclusions on temporal patterns in wireless data use. Our analysis results suggest the absence of obvious ones. Figure 8 shows how daily data use by all users changes during the two months of data collection. Daily data use goes up and down significantly twice

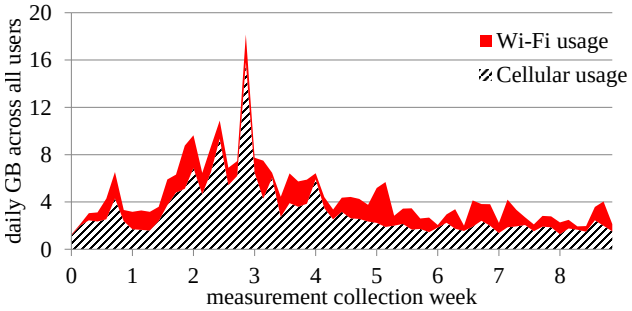

Figure 8: Daily data use, aggregated across all users.

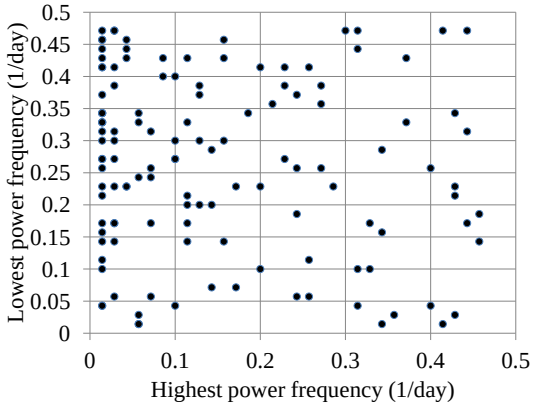

Figure 9: Frequency spectrum of daily data per user.

in the first month, corresponding to our two advertising campaigns to promote Metre in Google Play. Metre continues to gain and lose users along the way but the daily data use changes seem to be random without any apparent periodicity that may suggest a weekly or biweekly pattern.

We further examine if there is any temporal pattern in individual user's daily data use. We examine the 127 users for whom we have at least 28 consecutive days of data. We treat each user's daily data use as a time series and apply FFT to it to obtain its frequency spectrum. We then examine the spectrum to see if there is any frequency that has unusually high power to suggest there is a temporal pattern at that frequency. For conciseness, we characterize a user based on the highest and lowest power frequencies as a single data point in Figure 9. If there is no temporal pattern common to users, the data points should be uniformly, randomly distributed in the figure, which is what the graph shows. Although we see few more dots toward the left side of the graph, it is likely FFT artifacts from users with slightly more than 28 days of data. Notably, we do not observe any concentration of data points for frequency of 1/7, which would have suggested a weekly periodicity.

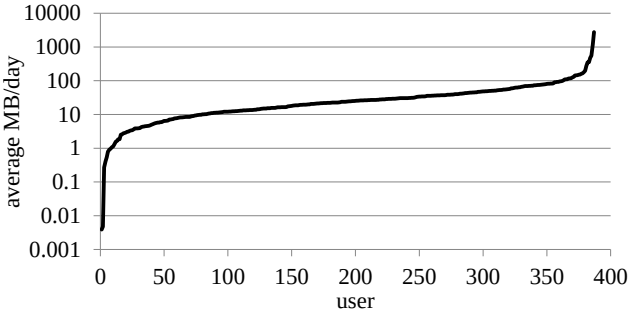

Figure 10: Average daily data use of all users ranked from the least to the most, each data point being $x(i)$ (i is user). The vertical axis is on a log scale.

IV.C. Heavy apps

We now consider $y(j)$, which is the average daily data use by app j. We use this value to identify the top 10 heavy apps by data usage. We also calculate this by time usage instead, and identify the top apps by time consumption.

Table 9 shows the top 10 apps in terms of time use and the top 10 apps in terms of data use. There is a large overlap between the top data and top time use apps. The two groups share six apps. The apps unique to the top time group are local media playback apps and a screen lock app that do not require Internet access for every launch, and a text messaging app that produces only a little data traffic. In contrast, the apps unique to the top data use group are players of online media, i.e., $tubemate$ and $realplayer$, and file download. It is interesting to note that $ibibo.mtt$, a third-party browser, uses more than five times data per hour than $android.browser$, the built-in browser.

IV.D. Heavy users versus light users

Users are very different in their daily data usage. Figure 10 is a histogram of average data consumption per day across all users. The median is around 16 MB.

We now examine the difference between the top and bottom thirds of the users in terms of daily data usage, i.e., $x(i)$ for data use. We call the top third users *heavy* users and the bottom third *light* users. We denote them with sets $\mathbb{H}$ and $\mathbb{L}$, respectively. The average daily data use by the heavy data users, $\sum_{i\in\mathbb{H}} x(i)$, is 84.5 MB while that by the light users, $\sum_{i\in\mathbb{L}} x(i)$, is 5.4 MB, or 16X smaller. We focus on comparing heavy and light data users in order to reveal what factors are correlated to the difference in daily data use.

app	$y(j)$ MB	hours	|MB|	|hours|	popularity # of users	% of users
devian.tubemate.home★	1068.59	6.86	0.61	0.06	45	11.63
com.sec.android.providers.downloads★	329.84	0	0.19	0	147	37.98
com.tencent.ibibo.mtt★	187.27	5.76	0.11	0.05	40	10.34
com.real.RealPlayer★	165.50	1.03	0.09	0.01	26	6.72
com.android.vending†★	1097.37	32.44	0.62	0.29	386	99.74
com.facebook.katana†★	742.32	36.41	0.42	0.32	237	61.24
com.google.android.youtube†★	724.26	13.41	0.41	0.12	309	79.84
com.UCMobile.intl†★	621.76	17.79	0.35	0.16	98	25.32
com.opera.mini.android†★	343.10	35.50	0.19	0.32	151	39.02
com.android.browser†★	332.61	72.15	0.19	0.64	363	93.80
com.whatsapp†	137.02	33.64	0.08	0.30	158	40.83
com.jiubang.goscreenlock†	77.47	30.98	0.04	0.28	37	9.56
com.cooliris.media†	146.18	29.20	0.08	0.26	309	79.84
com.android.music†	35.53	14.71	0.02	0.13	274	70.80

Table 9: Heavy apps measured by total data and time use. ★ top data apps; † top time apps

IV.D.1. Comparing Usage

We employ two techniques to compare usage by heavy and light users. In both techniques, we identify the top 10 apps by the heavy user, $\mathbb{A}_H^m$, and those by the light user, $\mathbb{A}_L^m$, in terms of use m, which can be time or data use. That is, the top 10 apps are the 10 apps with the highest $y(j)$ aggregated over each group. We focus on these two sets of apps while ignoring the rest. The rationale is that the top 10 apps account for over 80% of the usage of m [3, 14].

First, we show how $\mathbb{A}_L$ overlaps with $\mathbb{A}_H$. Next, we define a vector $V = \{v_1, v_2, ..., v_{|\mathbb{A}_L \cup \mathbb{A}_H|}\}$ to indicate the use of each app in $\mathbb{A}_L \cup \mathbb{A}_H$. Let V_L denote the use vector by the light users and V_H that by the heavy users. We compare the usage by the two groups quantitatively with the absolute distance between their vectors, i.e., $||V_L - V_H||$, and the cosine similarity between their vectors, which measures the angle between the two vectors in this space [12].

IV.D.2. Top apps by data use

We first examine the top 10 apps by *data use* for heavy and light data users, shown in Table 10. There are apps where both sets of users spend a lot of data bytes – three browsers, Facebook, YouTube and the app marketplace. However, heavy users tend to spend more bytes on apps that download files and videos – RealPlayer, MyFiles, TubeMate. In contrast, a larger fraction of data bytes for light users are spent in communication apps – email, Skype, WhatsApp.

The table also shows how many bytes and how much time the two sets of users spent in these apps. These two vectors are hard to directly compare between heavy users and light users because in part the magnitude (how much data a user spends overall) is different. Hence we calculate the normalized vectors for data and time consumption. We can compare two normalized vectors by calculating the smallest angle between them – the minimum angle of 0° means that the two vectors are the same, and the maximum angle of 90° means that they are opposites. If we compare the data normalized vectors for heavy users and light users, the angle between them is 54.8°. This indicates that the data they spend across these different top apps is significantly different. The angle between the normalized time vectors is 34.5°, which is much smaller than that between the data normalized vectors. That is, *how heavy and light data users spend their data is relatively more different than how they spend their time among these top data apps.* This difference is not explained by simply different top apps being used or simply different amounts of time spent on top apps.

IV.D.3. Top apps by time use

Similar to the previous analysis, we now examine the top apps where heavy and light data users spend most of their foreground time (as opposed to those apps where they spend most of their data). In Table 11, we again see some apps are common to both sets of users – web browsers, Facebook, and so on. Interestingly, heavy users spend a lot of time in YouTube, while light users spend a lot of time in Deskclock and BubbleShooter – two apps that consume very little data.

For the top time apps, the angle between the normalized vectors for data is 44°, and that for the time vectors is 34.7°. While the angle between the normalized vectors for time is similar to those reported above for top data apps (34.7° vs. 34.5°), the angle between the normalized vectors for data is smaller than that reported above for top data apps (44° vs. 54.8°). This means that the difference between how heavy and light users consume data from the top time

app	heavy users ($\sum_{i\in\mathbb{H}} v(i,j)$) MB	hours	|MB|	|hours|	# users	light users ($\sum_{i\in\mathbb{L}} v(i,j)$) MB	hours	|MB|	|hours|	# users
com.cooliris.media★	124.14	11.87	0.08	0.25	92	4.69	7.51	0.03	0.29	109
com.real.RealPlayer★	164.95	0.22	0.10	0.00	13	0.07	0.33	0.00	0.01	2
com.UCMobile.intl★	549.28	11.76	0.35	0.25	47	5.48	1.03	0.04	0.04	15
devian.tubemate.home★	1034.20	5.18	0.66	0.11	25	7.19	0.46	0.05	0.02	8
com.android.browser†★	166.56	35.87	0.11	0.75	123	52.35	14.41	0.35	0.55	116
com.android.vending†★	522.10	12.84	0.33	0.27	127	111.81	4.82	0.75	0.18	126
com.facebook.katana†★	529.55	14.16	0.34	0.30	82	69.11	9.08	0.46	0.35	64
com.google.android.youtube†★	621.90	10.47	0.39	0.22	110	16.03	0.60	0.11	0.02	90
com.opera.mini.android†★	214.52	11.47	0.14	0.24	54	25.71	7.92	0.17	0.30	41
com.sec.android.providers.downloads†★	240.48	0.00	0.15	0.00	47	17.67	0.00	0.12	0.00	39
com.android.email†	5.31	0.75	0.00	0.02	55	10.19	1.35	0.07	0.05	62
com.google.android.apps.maps†	20.04	2.37	0.01	0.05	115	11.78	1.47	0.08	0.06	120
com.viber.voip†	2.03	0.36	0.00	0.01	18	8.37	1.12	0.06	0.04	23
com.whatsapp†	49.95	7.25	0.03	0.15	41	28.48	15.76	0.19	0.60	57

Table 10: Heavy users and light users, $\mathbb{H}$ and $\mathbb{L}$, in their top apps by data consumption. ★ top data consumer for heavy users; † top data consumer for light users

app	heavy users ($\sum_{i\in\mathbb{H}} v(i,j)$) hours	MB	|hours|	|MB|	# users	light users ($\sum_{i\in\mathbb{L}} v(i,j)$) hours	MB	|hours|	|MB|	# users
com.google.android.youtube★	10.47	621.90	0.34	0.61	22	0.59	13.18	0.03	0.17	90
com.jiubang.goscreenlock★	13.04	31.40	0.42	0.03	113	5.24	4.38	0.23	0.06	4
com.UCMobile.intl★	11.76	549.28	0.38	0.54	60	1.04	5.53	0.05	0.07	16
com.android.browser†★	35.87	166.56	1.15	0.16	123	14.36	51.40	0.63	0.66	116
com.android.music†★	5.37	21.23	0.17	0.02	81	4.17	4.17	0.18	0.05	92
com.android.vending†★	12.84	522.10	0.41	0.51	127	4.85	109.74	0.21	1.40	126
com.cooliris.media†★	11.87	124.14	0.38	0.12	88	7.51	4.69	0.33	0.06	109
com.facebook.katana†★	14.16	529.55	0.45	0.52	86	8.99	65.89	0.40	0.84	64
com.opera.mini.android†★	11.47	214.52	0.37	0.21	17	8.10	26.80	0.36	0.34	42
com.whatsapp†★	7.25	49.95	0.23	0.05	45	15.65	28.21	0.69	0.36	56
com.sec.android.app.myfiles†	5.13	13.06	0.16	0.01	59	2.94	2.55	0.13	0.03	78
com.wssyncmldm†	0.80	0.32	0.03	0.00	44	2.60	1.17	0.11	0.01	22
util.sms†	2.22	19.05	0.07	0.02	28	4.46	7.00	0.20	0.09	15

Table 11: Heavy users and light data users, $\mathbb{H}$ and $\mathbb{L}$, in their top apps by foreground time. ★ top time consumer for heavy users; † top time consumer for light users

apps is smaller than that between how they consume data from the top data apps. This suggests that *the heavy and light users are more different in how they consume data via their top data apps.*

IV.E. Indicators of heavy data use

Based on our findings so far, we identify a candidate set of variables that may be correlated with being a heavy or light user. We now employ regression and correlation analysis to quantify this. Note that we identify correlation between variables, not causal relationships. Knowing such correlations can be useful for cellular operators, who may want to tailor their advertising or data plans to specific customers.

The technique we use here is the *ordered probit analysis*, which is a form of the regression analysis that allows for an ordinal categorical dependent variable. Regression analysis is a technique to estimate the relationship between dependent variables, or responses, and independent variables, or predictors. The dependent variable in our analysis is whether a user is a heavy user, a light user, or an in-between user (3 categories) as defined by cellular data usage (§IV.D). This dependent variable is *ordinal*. That is, there is a simple ordering between the categories, i.e., heavy users are those users who use *more* data than other users. The ordered probit analysis calculates the probability of a user being a heavy or light user or in-between, based on the independent variable(s). These probabilities can then be used for prediction by picking the most probable outcome. The prediction accuracy will be 33% when there is no dependency between independent and dependent variables and the prediction is purely random. As our input to each analysis, we measure and use the value of the independent variables for each user. Therefore, each analysis has 387 input values.

Table 12 summarizes the ordered probit analysis results with different independent variables. Below we summarize the main findings.

First, Wi-Fi usage is a mild indicator of cellular usage. As our independent variable in this analysis, we use a binary variable that indicates whether a user used Wi-Fi at all in our collected dataset. In fact, the

independent variable(s)	prediction accuracy
Wi-Fi	43%
screen size and resolution	40%
average time use	49%
heavy apps	46%
light apps	36%
altogether	58%

Table 12: Accuracy of predicting whether a user is a heavy user, or light user, or in-between user using different independent variables in the ordered probit analysis. Random guesses can do no better than 33%.

users who used Wi-Fi (125 users out of 387), used a median of 6.4 MB of cellular data per day, while the users who did not use Wi-Fi at all used a median 16.3 MB of cellular data per day, which further demonstrates a (negative) correlation between Wi-Fi and cellular usage.

Second, a smartphone's screen size and resolution shows a mild correlation with data usage. The independent variables for the screen size and resolution are the size in inches and the number of pixels, respectively. This finding goes against common intuition that better and larger screens result in higher data usage. The reasons for this intuition are that apps might download content with higher resolution for better screens, and that a larger screen size makes it easier for the user to browse webpages and apps. However, our data includes a limited set of screen sizes and resolutions, so this finding may be premature.

Third, total usage time of a smartphone is a decent indicator of data usage. The longer one uses the smartphone, the higher the data usage is likely to be. The independent variable in this analysis is the average daily time use of a user (the sum across all apps). Usage of heavy apps (Table 10) is also a decent indicator of data usage. However, the converse is not true – light apps are poor indicators. As our independent variables, we use the number of top 3 heavy apps and the number of top 3 light apps that a user has ever used. Heavy apps are a decent indicator because they consume significant amounts of data, and therefore, have a noticeable impact on a user's data usage.

Finally, the combination of these 5 independent variables can achieve a decent prediction accuracy of 58%. Recall that we have three categories of users, so random can do no better than 33%. Cellular operators can obtain estimates of these variables for new users by asking simple and quick questions, without the need for instrumentation of their phones. By knowing in advance what bucket a user will fall into, the cellular operator can promote different data plans.

V. Related work

Wireless traffic characterization: Xu et al. [15] characterized app usage by examining packet traces from a large cellular operator. Their packet traces cover a large number of users in the US and there are many interesting findings on the geographic locality of apps. Our goals are different and require us to do lab experiments and collect traces from phones, and would be difficult to achieve with packet traces. Since they rely on the HTTP User-Agent field to identify apps, they are unable to attribute non-HTTP traffic nor apps that embed browsers. Email, YouTube and browser usage can only be indirectly inferred and are hard to study in depth. Their packet traces do not include Wi-Fi usage, and do not indicate how long a user ran an app for. They do not compare different smartphone platforms in depth nor directly compare individual apps with similar functionality.

Falaki et al. [3] studied the behavior of 255 Android and Windows Mobile users. They found tremendous diversity in total data consumption and interaction time among different users. Similar findings were made by LiveLab [12, 14] which measures the smartphone usage of 34 iPhone users. Unlike Metre, these tools cannot breakdown network traffic by app. As a result, they did not investigate per-app data usage.

There have also been several studies [2,7,10,11] on network performance and how app traffic leads to excessive energy consumption on smartphones. In particular, periodic measurements and transfers, although small in terms of traffic volumes, may consume a disproportionately large amount of energy.

Caching and prefetching: Qian et al. [9] showed that most HTTP caching library implementations on smartphones do not conform to the HTTP specifications, nor do all apps effectively leverage caching opportunities, resulting in redundant transfers. Prefetching may reduce user-perceived latency, but the prefetched content that is not used results in wasted network consumption. Higgins et. al. [6] developed a prefetching library for mobile devices to optimize prefetches and reduce waste.

Data usage charging: Ha et al. [5] proposed time-dependent pricing, as opposed to conventional usage-based pricing, to incent users to shift traffic to off-peak hours. Peng et al. [8] demonstrated that the current charging systems employed by cellular operators can be inaccurate, leading to overcharges or undercharges in specific scenarios. Our work investigates how the wireless data usage is correlated with different users, apps, and platforms, which could help cellular operators improve their charging schemes. Zhang et al [16]

study the additional network traffic cost of free apps compared to their paid versions. They find that due to advertising and telemetry traffic, the paid versions are cheaper in the long term.

Offloading to other networks: Small cells, such as Wi-Fi hotspots and femtocells, can be leveraged to combat limited spectrum availability. Shifting traffic to these networks whenever available could reduce contention at cellular towers. However, it is unclear if that simply allows users to consume more, or if it actually reduces consumption at cellular towers. Rahmati et al. [13] designed a system for smartphones to transparently switch between cellular and Wi-Fi networks without any network support.

VI. Conclusion

Cellular consumption is a pain point for both users and network operators. Especially in countries where post-paid plans are the norm, users pay for the data that their apps consume. There is little clarity on how the choice of app and smartphone impacts data cost.

We have studied the data cost of different apps, platforms, and APIs. From field measurements, we have examined user behavior. We have found that even for a relatively straightforward task of getting a weather forecast, there is as much as 140X difference in data consumption between apps. When comparing apps with the same functionality and written by the same professional authors across different platforms, there is huge diversity in consumption. While no one platform is lighter than another across the board, a user with particular app habits can consume 3.3X higher data or more if they pick the wrong platform.

Operators may want to tailor data plans to users, incent specific behavior via coupons or rebates, or customize their marketing. Operators do not need intrusive monitoring to classify users – knowing what screen size a user prefers, whether they will use Wi-Fi, how long they may use the phone for and for which apps, has the potential for accurately classifying the user. Simply identifying a user's popular apps and pattern of usage can point to which of the three smartphone platforms is lowest in data consumption for them.

Several areas for future work remain open. Market research studies have been warning us of impending overload as a result of video streaming. While we did not see this behavior in our data, we wonder what the impact will be in the future. We did our lab experiments over Wi-Fi, and it is possible that some apps adjusted their behavior automatically based on what network they were on. We do not know if such behavior exists in current apps, and we expect new research and tools to be developed to help app developers automatically adjust their network usage. Finally, we specifically targeted users in India for our study, and it would be interesting to understand how their behavior is different from other parts of the world.

Acknowledgments

The authors thank Zhen Wang for his contributions to Metre.

References

[1] The comScore 2010 Mobile Year in Review. comScore Market Research.

[2] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A first look at traffic on smartphones. In *IMC*, 2010.

[3] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *ACM MobiSys*, 2010.

[4] R. Gilmore. Your life on a smartphone. In *HotMobile keynote*, 2012.

[5] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang. Tube: Time-dependent pricing for mobile data. In *ACM SIGCOMM*, 2012.

[6] B. Higgins, J. Flinn, T. Giuli, B. Noble, C. Peplin, and D. Watson. Informed mobile prefetching. In *ACM MobiSys*, 2012.

[7] J. Huang, F. Qian, Z. Mao, S. Sen, and O. Spatscheck. Screen-Off Traffic Characterization and Optimization in 3G/4G Networks. In *IMC*, 2012.

[8] C. Peng, G. Tu, C. Li, and S. Lu. Can We Pay for What We Get in 3G Data Access? In *ACM MobiCom*, 2012.

[9] F. Qian, K. Quah, J. Huang, J. Erman, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Web caching on smartphones: Ideal vs. reality. In *ACM MobiSys*, 2012.

[10] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. In *ACM WWW*, 2012.

[11] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Profiling resource usage for mobile applications. In *ACM MobiSys*, 2011.

[12] A. Rahmati, C. Shepard, C. Tossell, M. Dong, Z. Wang, L. Zhong, and P. Kortum. Tales of 34 iPhone users: How they change and why they are different. *arXiv preprint arXiv:1106.5100*, 2011.

[13] A. Rahmati, C. Shepard, C. Tossell, A. Nicoara, L. Zhong, P. T. Kortum, and J. P. Singh. Seamless flow migration on smartphones without network support. Technical Report 2010-1214, Rice University, 2010.

[14] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. Livelab: measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS*, 2010.

[15] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *IMC*, 2011.

[16] L. Zhang, D. Gupta, and P. Mohapatra. How expensive are free smartphone apps? In *ACM MC2R*, 2012.