# MobiScope: Practical Mobile Internet Traffic Monitoring


Ashwin Rao
INRIA

Amy Tang
UC Berkeley

Justine Sherry
UC Berkeley

Adrian Sham
University of Washington

Sam Wilson
University of Washington

Shen Wang
University of Washington

Arnaud Legout
INRIA

Walid Dabbous
INRIA

Arvind Krishnamurthy
University of Washington

David Choffnes
University of Washington



## ABSTRACT

Existing platforms to monitor mobile Internet traffic fall short of being practical because they fail to be either portable, pervasive, passive, or deployable. In this paper, we present *MobiScope*, a practical VPN based platform solutions to monitor mobile Internet traffic. We posit that *MobiScope* can be used to monitor the mobile Internet traffic regardless of the mobile operating system, source of the mobile application, access technologies, and ISPs serving the mobile device. *MobiScope*'s practicality is powered by some novel functionality provided by Mobile OSes to manage VPN tunnels, namely, *VPN On-Demand* by iOS and [**TBD: ...**] by Android.

We use *MobiScope* to detail the characteristics of iOS Push Notifications, compare the behavior of popular social networking apps across Android and iOS, and study the [**TBD:** ] populars in the Apps in the Android and [**TBD:** ] number popular apps in the iOS market. In our experiments and measurement studies we observe [**TBD:** ] ...


## 1. INTRODUCTION

Our contributions are as follows

- We posit that VPNs can be used build a practical platform to monitor mobile Internet traffic regardless of the mobile operating system, device type, access technologies, and service provider.
- We present *MobiScope*, a practical platform for comprehensive monitoring of mobile Internet traffic. *MobiScope* builds on existing functionality provided by Mobile OSes to manage VPN tunnels.
- We use *MobiScope* to compare the behavior of popular mobile applications over Android and iOS. We observe [**TBD: values come here**]. We also compare the behavior of these application over Wifi and 3G.
- [**TBD: Results based on Amy work**].
- [**TBD: Results from an on going IRB based study of 30 users. We use these results to compare our observations from exisiting studies. The key take home is that these measurements were did not require custom OSes, ISP support, or support from marketplaces, warranty voiding of devices.**]

[**TBD: Things to highlight in Intro
Tools
Techniques
Methodology
Insights**]

## 2. MOTIVATION

Despite the increasing popularity of mobile devices the current mobile ecosystem offers researchers a limited view into the Internet traffic generated by the mobile devices and the installed apps. Further, we have absolutely no idea if ISPs interfere with mobile Internet traffic. This opaqueness of mobile Internet traffic can be reduced only if comprehensive traffic traces from end users is available to detail the intricacies of mobile Internet traffic.

The activity of detailing these intricacies includes detailing the characteristics of mobile Internet traffic and zooming into the causes for the observed behavior. Intuitively, the causes of the observed behavior include the design decisions of underlying operating system and the apps, the APIs used by the apps, the access technology used to connect to the Internet, and the way ISPs manage mobile Internet traffic.

Current approaches that can be used to detail these intricacies of mobile Internet traffic include instrumenting the mobile operating system (OS), instrumenting app binaries, static analysis of app binaries, or relying on ISP traces. However, due to the various reasons we present in the rest of this section, these approaches are unsuitable to obtain a comprehensive perspective of mobile Internet traffic in a practical manner from end users.

Instrumenting a mobile OS system using tools such as Taintdroid [5] and AppFence [8] provides researchers a fine grained view of the apps and OS in action. However, Instrumenting an OS also results in a high barrier to entry for practical measurement studies and experiments on mobile Internet traffic that require participation of end users who may be unwilling to modify the underlying OS and void the warranty of their devices. Furthermore, longitudinal studies that detail the impact of OS code changes and app code changes cannot be performed by instrumenting OSes.

Instrumenting app binaries at predefined code points using tools such as AppInsight [11] can be used to detail the behavior of a specific set of apps. Indeed, AppInsight provides a detail analysis of apps, however, in terms of the network footprint of the app, the scope of AppInsight is limited to the instrumented apps, the marketplaces from where the apps are downloaded, and the OS version for which the app was instrumented. Furthermore, each new version of the app needs to be instrumented to details the impact of the changes made for that version.

Static analysis of the app code can be used to study apps whose code cannot be instrumented. For example, PiOS [4] was used to perform static analysis of 1400 IOS apps by static analysis. A shortcoming of PiOS is that access to the app binary is possible only if the device is jail-broken, thus voiding the warranty of the device. Furthermore, like AppInsight [11], the results of PiOS are limited to the iOS operating system. [**TBD: Dave Text for SPARTA project at UW.**]

ISP traces are useful to study mobile devices in the wild. Viallina-Rodriguez *et al.* [17] use an ISP trace of 3 million subscribers to detail the impact of ads and analytics on the mobile data and energy consumption. Similarly, Maier *et al.* [10] study the mobile traffic by looking at the DSL traces from a popular European ISP. However, these studies cannot provide a comprehensive view of the traffic from mobile devices because users can access the Internet using different ISPs depending on their location and the access technology used to connect to the Internet. For example, the home Wi-Fi and office Wi-Fi may be served from ISPs that are different from the ISP used for cellular data traffic.

The problem of ISP interference was highlighted by Reis *et al.* [12]. The authors demonstrated that inflight changes made by ISPs tend to introduce vulnerabilities such as overflows and cross-site scripting (XSS) attacks. They proposed and deployed *Web Tripwires*, Javascript code that detects in-flight page changes. The main limitation of their study is that the approach requires each Web site to modify their content to include a tripwire that can detect ISP interference. *Web Tripwires* is therefore not deployable because it requires support from the Web site maintainers.

In summary, existing solutions to measure the network characteristics of mobile Internet traffic fall short of being either portable, pervasive, passive, or deployable. In the next section we present *MobiScope* a practical approach that uses traffic redirection to monitor mobile Internet traffic. [**TBD: Discuss new OSes coming out. .. Ubuntu, Firefox OS, etc**].

# 3. TRAFFIC REDIRECTION TO MONITOR MOBILE INTERNET TRAFFIC

In this section, we begin by enumerating the goals for practical mobile Internet traffic monitoring. We then show how *MobiScope* extends the existing functionality provided by mobiles OSes and uses traffic redirection to achieve the described goals in a feasible manner.

## 3.1 Goals

Our primary goal was to obtain comprehensive visibility into the mobile Internet traffic. To meet this goal, we further identify the following sub-goals we believe are important to ensure that our approach is practical and feasible.

1. *Portable.* We want our monitor mobile Internet traffic regardless of operating system, access technology, and service provider of the mobile devices. Portability ensures that the measurement studies are realisitic and the data collected can be used to compare different OSes, access technologies, and service providers in action.
2. *Pervasive.* Seamless visibility to all the Internet traffic generated by mobile device is essential to ensure that the data collected is comprehensive. Pervasiveness ensures that the measurement results are realistic and can be used to study real users including those that use mobile devices "on the move."
3. *Passive.* The data collected will not comprehensive if traffic monitoring requires explicit triggers from the end users. Passiveness also ensures that the system is capable of capture the network traffic even when the devices are *idle*.
4. *Deployable.* An easy to use and deployable solution that has a low barrier to entry is essential to ensure participation from end users.

In summary, we use portability, pervasiveness, passiveness, and deployablity as the building blocks to define comprehensive and practical monitoring of mobile Internet traffic.

## 3.2 Description

We posit that traffic redirection can be used to comprehensively monitor mobile Internet traffic. For our analysis we use two forms of redirection: a VPN based redirection to detail the network traffic characteristics of mobile device, and a transparent proxy based redirection to detail ISP interference of mobile Internet traffic.

### 3.2.1 VPN based Traffic Redirection

The use of VPNs by corporate executives "on the move" to securely connect to corporate servers with their mobile device was our motivation to consider VPNs. VPN usage by corporate clients gave us hints towards the portability and deployability of VPNs. Further investigation showed us that Mobile OSes expose features that can make them pervasive and suitable for passive monitoring of mobile Internet traffic.

VPNs are deployable and portable because Android, BlackBerry, Bada, and iOS all support VPNs tunnels over Wi-Fi and the cellular interface. We now provide an overview of the features that make them pervasive. All iOS devices (version 3.0 and above) come with a feature called "VPN On-Demand". *VPN On-Demand* forces the iOS device to use VPN tunnels when connecting to a specified set of domains. Using trial-and-error, we discovered that VPN On-Demand uses suffix matching to determine which domains require a VPN connection. We extend this feature to ensure that a VPN tunnel is used when an iOS device connects to the Internet. Android version 4.0 and above comes with native VPN support. Unlike iOS, Android does not offer an equivalent of *VPN On-Demand*; however, Android provides an API that allows an user space app to manage VPN connections. We modify the open source StrongSwan VPN client [14] to ensure that the VPN reconnects each time the preferred network changes (*e.g.*, when a device switches from cellular to Wi-Fi). As of Android 4.2, Android supports "Always On" VPN connections that uses VPNs to tunnel all the data traffic. [**TBD: Dave:text on Always ON from inputs from Adrian**].

We believe that any practical platform should be based on *off-the-shelf* hardware using open source software. Open source VPN solutions to manage VPN tunnels include Strongswan, Openswan, and OpenVPN. *MobiScope* uses Strongswan [13] because it is the only open source solution that can use the IPsec services of the Linux kernel *without any kernel modifications*. We emphasize on VPN tunnels created using IPsec, though PPTP and L2TP can be used to create VPNs tunnels, because the *VPN On-Demand* feature of iOS is supported only for VPN tunnels that use IPsec. Furthermore, Strongswan also supports IKEv2 [9] protocol used by Android clients and the IKEv1 [7] protocol used by iOS devices. Thus Strongswan, which is used the IPsec services of the Linux kernel ensures that VPN tunnels can be managed using *off-the-shelf* hardware and open source software.

### 3.2.2 Redirection using a Transparent Proxy

[**TBD: Dave add text here**]

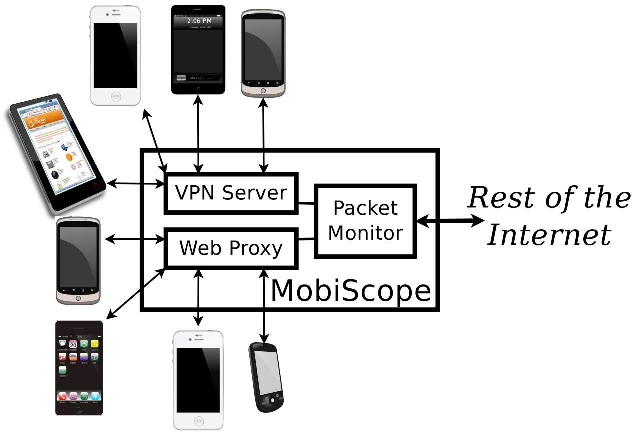


Figure 1: *MobiScope* uses traffic redirection to monitor mobile Internet traffic. *MobiScope requires mobile clients to redirect their traffic through a server that monitors Internet traffic. VPN based redirection is used to characterize mobile traffic sans ISP interference. To detect ISP interference* MobiScope *relies on a single hop transparent Web Proxy.*

### 3.2.3 Traffic Monitoring Setup

As shown in Figure 1 mobile devices redirect their Internet traffic through a *MobiScope* server where all their Internet traffic is monitored. *MobiScope* uses VPNs for comprehensive monitoring of mobile Internet traffic sans ISP interference. *MobiScope* relies on a a transparent Web proxy to detail interference by mobile ISPs.

Deploying VPNs on mobile clients is simple for end users primarily VPNs are natively supported by popular mobile OSes. The Android users need to install a certificate and fill our five fields while iOS users need to install a configuration file. Once the configurations are stored, all Internet traffic from the mobile device flows through *MobiScope*. *MobiScope* uses Strongswan to manage VPN tunnels of *MobiScope*. This simplicity is important for practical and realistic measurement studies with end users.

We use tcpdump to monitor the packets that flowing through *MobiScope*. We now present the technique we used to segregate traffic from mobile devices. *MobiScope* uses NAT to divert the packets encapsulated in the VPN tunnels to Internet. In the ideal scenario, the encapsulated packets from the mobile device would be tunneled to *MobiScope*. On *MobiScope*, these packets would be decapsulated and the packets would then undergo NAT before leaving for their intended destination. The packets from the mobile clients to the Internet could be monitored before they undergo NAT. However, due to the existing network stack implementation, the packets from the Internet that are destined to the mobile clients undergo NAT and IPSec encapsulation take place in one step. The inter-dependencies

[**TBD: Dave: New figure for Tripnet comes here.**]

Figure 2: Overview of the Web tripnet

between these various modules responsible for routing, NAT, and IPSec make it difficult segregate and monitor the packets. We address this issue by looping the packets through a virtual (*tun/tap*) device. The looping of packets through a virtual device allows us to monitor the packets when they are outside the VPN tunnels and have not undergone NAT.

[**TBD: Dave: Figure 2 text for tripnet should come here**]

In summary, mobile VPNs are portable and deployable because they are natively supported by popular mobile operating systems. We build on existing features provided by iOS and Android to make sure that the VPN tunnels are pervasive and are created passively. We rely on the open source Strongswan VPN daemon to manage VPN tunnels which makes *MobiScope* deployable. We plan to release the source code to configure mobile Internet traffic using *MobiScope*.

## 3.3 Feasibility

Traffic redirection when using *MobiScope* implies that all services that offer features based on IP address of their clients shall react according to the IP address of the server rather than the IP address of the mobile client. Furthermore, some ISPs are known to block VPNs. During our measurement we observed one such ISP that blocked VPN tunnel creation requests from one of our clients. We now show that the cost to redirect traffic in terms of latency, data consumption, and power is sufficiently low.

### 3.3.1 VPN Latency Overheads

The iOS devices use IKEv1 to manage the VPN tunnels while Android devices support both IKEv1 and IKEv2. To establish the VPN tunnel, IKEv1 requires a total 16 packets to be exchanged between the mobile client and the VPN server while IKEv2 requires 4 packets. We use IKEv2 for our Android devices while IKEv1 is used for the iOS devices.

We performed controlled experiments using one Android device and an iPhone 5 to measure the time required to establish a VPN tunnel. We performed this test from two different locations and performed !**number**$_{verify}$! of connections over !$_{verify}$! hours. These two locations were based in the same city in which the server was deployed. For the Android device, we observe a median connection establishment time of !**0.62**$_{verify}$! seconds from both locations when using Wi-Fi with a maximum of !**0.81**$_{verify}$! seconds. The median connection establishment time was !**0.81**$_{verify}$! seconds with a maximum of !**1.59**$_{verify}$! seconds from both locations when the Android device used cellular networks to establish the tunnel. Compared to the Android device, the iOS devices required a larger amount of time to establish the connection. We observed a median connection establishment time of !**1.60**$_{verify}$! seconds and !**1.34**$_{verify}$! seconds with a maximum of !**2.0**$_{verify}$! seconds and !**1.48**$_{verify}$! seconds respectively from the two Wi-Fi networks; in the case of cellular networks we observed a median of !**1.80**$_{verify}$! seconds and !**1.65**$_{verify}$! seconds with a maximum of !**2.18**$_{verify}$! seconds and !**1.87**$_{verify}$! seconds respectively.

In summary, we observe that because iOS devices take up to twice as much time as Android devices because iOS devices use an older key management algorithm (IKEv1). [**TBD: Any more insights .. The tunnel establishment times in the order of 2 seconds implies that *MobiScope* can have a significant latency overhead if VPN tunnels are established periodically for short tests.**]

### 3.3.2 Data Consumption Increase when using VPN

IPSec encapsulation slightly inflates packet sizes, in addition to preventing carrier middleboxes from applying their own compression. We measured the overhead of the tunnel in terms of data overhead from IPsec headers and keep-alive messages, finding that it ranges from !**8–12.8**$_{verify}$!%.

For our measurements, we capture the encrypted packets exchanged by our *MobiScope* servers and the mobile clients that use *MobiScope*. We performed the packet capture for !**30**$_{verify}$! days during which !**25**$_{verify}$! devices tunneled their traffic via *MobiScope*. During this time interval we also capture the packets that were encapsulated in the IPsec packets. We use these samples to compute the increase in the amount of bytes transferred due to encapsulation and the keep-alive messages. During the !**30**$_{verify}$! day period we observe that the median of the increase to be !**8.31%**$_{verify}$!, with a maximum increase of !**12.8%**$_{verify}$!.

[**TBD: In summary, we observe a maximum overhead of 12.8% increase in data consumption. We believe the costs of this overhead are minimal compared to the cost of warrant voiding the device.**]

### 3.3.3 Power Overheads when using VPNs

[**TBD: Daves results**]

[**TBD: In summary, we show *MobiScope* is feasible to build and deploy**]

### 3.3.4 Impact of using Transparent Proxy

[**TBD: Dave: Text comes here**]

## 3.4 Discussion

In this section we show that traffic redirection can be used for comprehensive monitoring of mobile Internet traffic. *MobiScope* relies on the low barrier to entry offered by VPNs that can be served using *off-the-shelf* hardware and open source software. [**TBD: some more text here on the pros and cons of this approach**] We now show how we used *MobiScope* for measurements using real users and also for controlled experiments to detail the characteristics of mobile Internet traffic.

# 4. CONCLUSION

Placeholder

# 5. ALL PLACHOLDER SECTIONS COME HERE

# 6. DATASET DESCRIPTION

In this section, we describe the three datasets: *mobAll*, *mobCompare*, and *mobileExpt*. We use these datasets in our studies that we present in the subsequent sections.

[**TBD: Run genDataSetDescription.R to get these numbers**] The *mobAll* dataset consists of mobile data traffic traces from !**25**$_{verify}$! devices that belong to !**19**$_{verify}$! users who are volunteers of an IRB approved study. This dataset consists of !**9**$_{verify}$! iPhones, !**4**$_{verify}$! iPads, !**1**$_{verify}$! iPodTouch, !**10**$_{verify}$! of Android phones, and !**1**$_{verify}$! of Android tablet. Though *tablets* can access the Internet via a cellular data connections, we consider tablets to be devices that only use Wi-Fi to access the Internet. The Android devices in this dataset include the Nexus, Sony, Samsung, and Gsmart brands. The users of the 25 devices are spread across France and USA. This dataset consists of !**176**$_{verify}$! days of data that flowed through our VPN servers; the number days for each user varies from !**5**$_{verify}$! to !**176**$_{verify}$! with a median of !**32**$_{verify}$! days.

[**TBD: we need some wording and consitency for the usage of ISP – for example ATT can provide cellular and DSL. Also mobile data cannot be used and we need some word for cellular data and wifi data and this must be defined in the dataset description.**] The *mobAll* dataset consists of data traffic from !**54**$_{verify}$! distinct ISPs, of which !**10**$_{verify}$! provided cellular services. Of the 19 devices that used cellular data, we observed that 16 devices restricted their cellular data traffic to one ISP each; the other three users used four, two, and two ISPs respectively. The number of Wi-Fi ISPs per device was larger, the median number of ISPs observed was 4 with a maximum of 24 for one user. The user who contributed 24 distinct ISPs used *MobiScope* when traveling across 6 different countries. This implies *MobiScope* was able to capture traffic of users "on the move." Campus wide studies such as [**TBD:** ], studies on DSL networks [10], and studies limited to traces from one specific ISP [17] are not able to capture this behavior.

The *mobCompare* dataset consists of data from two users who had three and two devices respectively. The three devices that belong to one of the two users consists of a two smart phones and a tablet, while one smartphone and a tablet belong to the second user. We use this dataset to compare the behavior of popular apps and to detail the behavior of devices when the device is kept idle. The data set consists of !**number**$_{verify}$! of days of data from each device and [**TBD: number**] of days for which data traffic was seen for all the 24 hours.

The *mobileExpt* dataset contains the traffic traces from an Android device and an iOS device that were used to perform a controlled experiment on popular application. We tested !**number**$_{verify}$! of Android applications and !**number**$_{verify}$! of iOS application for this study. [**TBD: How we decided this list**]. [**TBD: How we performed the test**] [**TBD: other attributes of this dataset**].

# 7. MEASUREMENT RESULTS

[**TBD: In this section we ...** ]

## 7.1 Descriptive Statistics

We now use several descriptive statistics to summarize our *mobAll* dataset. We use these descriptions to support our claim that *MobiScope* can be used as a portable, pervasive, and deployable platform for passive measurements of mobile network traffic.

### 7.1.1 Comparison of Devices

One advantage of using *MobiScope* is that we can detail and compare the behavior of mobile devices regardless of the underlying operating system. We now use Table 1 to directly compare Android and iOS, focusing on the key protocols used by the devices.

Our first key observation is that HTTP is responsible for the majority TCP traffic volume in bytes, !**63.13%**$_{verify}$! for Android and !**80.28%**$_{verify}$! for iOS. This observation is inline with previous results that report HTTP to be the dominant protocol used by mobile devices [6, 10]. We also observe that the majority of TCP flows are sent over SSL, !**43.77%**$_{verify}$! of total flows for Android and !**32.18%**$_{verify}$! of total flows for iOS . We believe this occurs across platforms because the majority of flows come from e-mail, messaging and social networking, all of which use secure channels regardless of the OS. We discuss some of these popular application in [**TBD: section**].

For the Android devices in the *mobAll* dataset, !**96.45%**$_{verify}$! of the UDP flows that account for !**70.92%**$_{verify}$! of the UDP bytes are due to DNS request. The rest of

| Protocol | Service | Android | | iOS | |
|---|---|---|---|---|---|
| | | Flows | Bytes | Flows | Bytes |
| TCP | HTTP | 13.55 | 63.13 | 14.64 | 80.28 |
| TCP | SSL | 43.77 | 31.36 | 32.18 | 18.98 |
| UDP | - | 37.23 | 1.09 | 47.32 | 0.55 |
| TCP | other | 2.24 | 4.30 | 1.55 | 0.15 |
| Other | - | 3.19 | 0.10 | 4.03 | 0.01 |
| *total* | | 100.00 | 100.00 | 100.00 | 100.00 |

Table 1: Percentage of flows and bytes from iOS and Android devices. [**TBD: Verify total 100 for final results**] *SSL is responsible for the majority of TCP flows from iOS and Android devices.*

the traffic is due to other applications such as Skype. Similarly, **!80.98%**$_{verify}$! of UDP flows that account for **!66.29%**$_{verify}$! of UDP bytes from iOS devices are due to DNS requests.

We observe 0.1% of the traffic volume was classified as *other* for Android device which is larger than the 0.01% observed for iOS. This is because the Android users in our dataset tend troubleshoot network connectivity issues using applications that perform pings and traceroutes.

On balance the number of bytes transferred per unit time in Android, and the number of flows, is larger than the same for iOS. We found that iOS devices generated about **!number**$_{verify}$! of traffic per hour while Android devices generated about[**TBD: number**] MB/hr, and increase of **!number%**$_{verify}$!. Further, Android devices contributed more flows (**!number**$_{verify}$! per hour vs. **!number**$_{verify}$! per hour), an increase of **!40%**$_{verify}$!. It is difficult to account for the impact of user behavior on device-generated traffic; that aside, one explanation is that Android devices generate more bytes and flows due to the relatively permissive API for running code in the background on Android. In contrast, iOS quickly kills processes that are in the background, preventing them from generating network traffic after only a few seconds of losing the foreground. We discuss this [**TBD: section**]

[**TBD: In summary,** ]

### 7.1.2 *Comparison of Access Technology*

*MobiScope* allows us to passively monitor traffic regardless of the access technology used by the mobile device. We use this feature of *MobiScope* to provide a direct comparison of how the different access technologies are used by our users.

In Figure 3 we present the share of Wi-Fi as a fraction of the total traffic generated by the user. The devices are sorted according to the share of Wi-Fi traffic as a fraction of the total traffic from the device. The devices with device IDs from **!18**$_{verify}$! to **!23**$_{verify}$! are devices that do not have a cellular data plan associated with them device: **!3**$_{verify}$! iPads, **!1**$_{verify}$! iPodTouch,

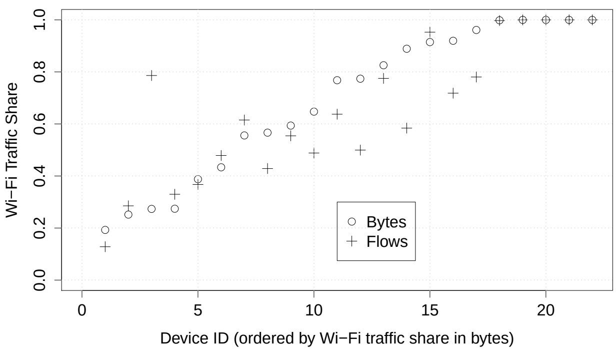


Figure 3: Traffic share of Wi-Fi as a fraction of total traffic from a device.

| Protocol | Service | Wi-Fi | | Cellular | |
|---|---|---|---|---|---|
| | | Flows | Bytes | Flows | Bytes |
| TCP | HTTP | 16.14 | 81.59 | 10.99 | 57.08 |
| TCP | SSL | 32.56 | 16.97 | 43.99 | 39.59 |
| UDP | - | 46.29 | 0.52 | 38.56 | 1.25 |
| TCP | other | 1.43 | 0.89 | 2.46 | 1.97 |
| Other | - | 3.55 | 0.02 | 3.96 | 0.09 |
| *total* | | 100.00 | 100.00 | 100.00 | 100.00 |

Table 2: Percentage of flows and bytes using Wi-Fi and Cellular as access technology. [**TBD: Verify total 100 for final results**] *The share of SSL traffic over Cellular networks is larger than its share over Wi-Fi in terms of bytes and flows*

and **!1**$_{verify}$! Android tablet. The diversity in Wi-Fi traffic share in terms of flows and bytes indicates the diversity of mobile device usage. For example, device ID **!3**$_{verify}$! is used by a user who prefers to use cellular network to listen to music, watch videos, and troubleshoot the cellular network connectivity issues using tools such as speedtest; the primary use of the device is used to read emails and access social networks when on Wi-Fi. Media content such as music and videos have a higher share of traffic volume per flow, therefore the share of cellular traffic in higher for device ID **!3**$_{verify}$!. Similarly, device ID **!1**$_{verify}$! is used by a user who prefers to use cellular networks.

In Table 2 we further classify the protocols and services that are responsible for Wi-Fi and cellular data traffic with the help of the techniques used to generate Table 1. The first key observation is that the portion of HTTP bytes sent over Wi-Fi and cellular are significantly different. Upon further inspection, we found that Wi-Fi is the preferred medium to transfer media content, which generates relatively large flows. For example, apps such as Google Plus image backup on Android allow users to upload their images only over Wi-Fi. In line with the difference in HTTP traffic ratios, we note that SSL flows are dominant type in cellular connec-

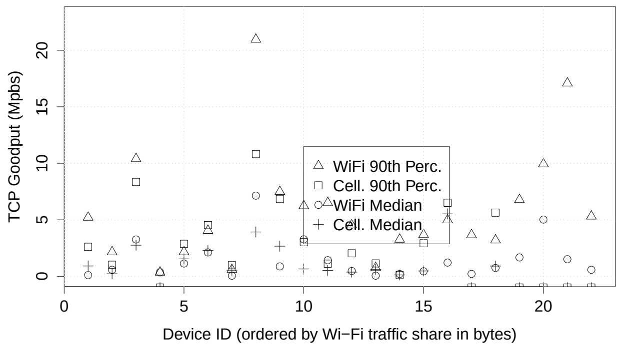


Figure 4: Goodput of TCP flows over Wi-Fi and cellular networks.

tions. We also observe a smaller number of UDP flows for cellular networks. [**TBD: Why do we observe less??**].

In Figure 4, we present the 90th percentile and median TCP goodput of flows that exchanged more than 512 kB of data. [**TBD: cite [2] if needed for 256 kB**]. We use the goodput because it a good estimate of how the applications perceive the quality of the network. The devices are sorted according to the same technique used to sort the devices in 3. For device 16 we observe higher goodput for cellular compared to Wi-Fi while for device 6 we observe similar values for the TCP goodput; this highlights opportunities for proposals such as 3G onloading [16]. Across all users we observe that ratio of median goodput over Wi-Fi to the median goodput over cellular varies from ! $_{verify}$! to ! $_{verify}$! with a median of !$_{verify}$!.

[**TBD: In summary,** ]

### 7.2 Case Studies

### 7.3 Controlled Experiments

### 7.4 Discussion

## 8. RELATED WORK

Placeholder for the papers. [15]

[3]

[17]

[6]

Bro port based classification of HTTP, SSL, other, and so on.

## 9. CLASSIFICATION OF APPLICATION

*MobiScope* enables us to monitor the data being exchanged by the mobile device. However, it does not provide any details of the source of the data. We use the following technique to estimate the source of the mobile data traffic. In 1 and 2 we observe that TCP is responsible for the majority of the traffic volume from iOS and Android devices regardless of the access technology used. We therefore focus on associating TCP flows to the applications for our analysis. From our analysis we were able to classify !x%$_{verify}$! of the TCP traffic volume and !y%$_{verify}$! of the TCP flows to the applications.

### 9.1 Methodology

### 9.2 Results

User Agent based classification flows per user with a blank user agent. flows per user with a default user agent flows with application in user agent

SSL certificate based classification flows to dedicated hosts flows to cdns

### 9.3 Discussion

## 10. SPECIFIC APPLICATION BEHAVIOR

## 11. ADS AND ANALYTICS

Ads and Analytic sites have received considerable attention. The reason for the attention being the intrusiveness exhibited by the ads and analytics sites in tracking personal information. [17] show that ! $_{verify}$! of data traffic volume observed from a Cellular ISP is because of mobile ads. In our traces we use the classification of [17] and [1] to classify flows as ads flows.

## 12. DISCUSSION ON PLATFORM

IPv6 support of VPNs

ISP support of VPNs. One ISP blocked VPN access.

Open source ??

Releasing datasets ??

## 13. REFERENCES


[1] Ad blocking with ad server hostnames and ip addresses. `http://pgl.yoyo.org/adservers/`.

[2] HTTP Archive. `http://httparchive.org`.

[3] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones. In *Proc. of IMC*, page 280, New York, New York, USA, Nov. 2009. ACM Press.

[4] M. Egele and C. Kruegel. PiOS: Detecting privacy leaks in iOS applications. *Proc. of the NDSS Symposium*, 2011.

[5] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *Proc. of USENIX OSDI*, 2010.

[6] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin. A First Look at Traffic on Smartphones. *Proc. of IMC*, page 281, 2010.

[7] P. Hoffman. Rfc 4109: Algorithms for internet key exchange version 1 (ikev1). *IETF Request For Comments, http://www.ietf.org/rfc/rfc5996.txt*, 2005.

[8] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proc. of CCS*, 2011.

[9] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Rfc 5996: Internet key exchange protocol version 2 (ikev2). *IETF Request For Comments, http://www.ietf.org/rfc/rfc5996.txt*, 2010.

[10] G. Maier, F. Schneider, and A. Feldmann. A First Look at Mobile Hand-held Device Traffic. *Proc. PAM*, 2010.

[11] L. Ravindranath and J. Padhye. AppInsight: Mobile App Performance Monitoring in the Wild. *Proc. of USENIX OSDI*, 2012.

[12] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver. Detecting in-flight page changes with web tripwires. In *Proc. of USENIX NSDI*, 2008.

[13] Strongswan. www.strongswan.org.

[14] strongswan VPN client. http://play.google.com/store/apps/details?id=org.strongswan.android.

[15] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh. Who Killed My Battery: Analyzing Mobile Browser Energy Consumption. In *Proc. of WWW*, pages 41–50, 2012.

[16] N. Vallina-Rodriguez, V. Erramilli, Y. Grunenberger, L. Gyarmati, N. Laoutaris, R. Stanojevic, and K. Papagiannaki. When david helps goliath: The case for 3g onloading. In *Proc. of HotNets*, 2012.

[17] N. Vallina-Rodriguez, J. Shah, A. Finamore, H. Haddadi, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft. Breaking for commercials: Characterizing mobile advertising. In *Proc. of IMC*, 2012.