# Meddle: Transparency and Control for Mobile Network Traffic

Ashwin Rao
INRIA

Amy Tang
UC Berkeley

Justine Sherry
UC Berkeley

Arnaud Legout
INRIA

Walid Dabbous
INRIA

Arvind Krishnamurthy
University of Washington

David Choffnes
University of Washington

## ABSTRACT

Today's mobile systems make it difficult to obtain visibility into network traffic and provide few mechanisms for changing how devices and apps use the Internet. As a result, we have a poor understanding of – and limited ability to improve – network performance, battery consumption, data-quota usage and privacy in the mobile environment. In this paper, we address these issues using *Meddle*, a platform for improving transparency for mobile networking traffic and enabling fine-grained control over flows generated by mobile devices.

*Meddle* achieves this by using virtual private networks (supported by nearly all devices) to forward traffic from devices to a server that uses software-middlebox techniques to modify, shape and/or block network traffic before forwarding it to the destination. Unlike previous work, *Meddle* provides the ability to characterize and control mobile-device traffic on user devices regardless of the OS, carrier or access technology – without rooting the device or modifying the default OS. Further, *Meddle* allows us to experiment with new techniques for interposing on mobile traffic in a way that improves performance, resource consumption and privacy.

This paper describes results from our initial deployment of *Meddle*, showing that it imposes reasonable overhead while providing users with services previously unavailable on mobile devices. We further use the network flows traversing *Meddle* to reveal unexpected and alarming network behavior of mobile apps made visible through our approach. Last, we discuss how *Meddle* can address these and other issues in the mobile environment without requiring app or OS modification.

## 1. INTRODUCTION

Mobile systems consist of walled gardens inside gated communities, i.e., locked-down operating systems running devices that interact over a closed and opaque mobile network. Despite a large collection of privacy, policy and performance issues in mobile networks [5, 8, 15, 10], users and researchers are faced with few options to characterize and address them.

In this paper, we demonstrate that we can provide the necessary framework to simultaneously address many of these issues for users and researchers by using middleboxes accessible through VPN tunnels, an approach we call *Meddle*. *Meddle* works for nearly all mobile devices out of the box: Android, iOS, and Blackberry all support VPNs, thus providing a portable mechanism to tunnel traffic to servers outside of the carriers' control regardless of the mobile device's network. Once packets arrive at VPN servers, we use a variety of software-middlebox approaches to transform traffic to and from mobile devices. Together, these features enable new opportunities for characterizing and controlling network traffic in the mobile environment.

First, *Meddle* provides a portable, pervasive, passively-gathered view of mobile network activity from mobile devices. It is portable in that VPNs work on nearly all devices and they capture traffic regardless of which carrier or access technology (*e.g.*, cellular or WiFi) is being used. It is pervasive in that *Meddle* ensures a device's VPN is always connected when devices access the Internet, offered an comprehensive and continuous view of network traffic generated by devices. By tunneling traffic to a server we control, *Meddle* enables low-cost passive monitoring of traffic naturally generated by users, OSes and apps on their own devices.

Second, *Meddle* provides a new point of control over mobile network traffic. *Meddle* enables researchers to investigate what-if scenarios for the impact of new middleboxes as if they were deployed in carrier networks. These include new app-accelerators, mobile-specific security filters and protocol manipulation to improve power consumption and data quota usage. Importantly, service providers and users can take advantage of these features without requiring any support from carriers or new OS-specific apps installed by users.

Last, *Meddle* facilitates participation from a large number of users, allowing us to characterize and evaluate mobile systems in the wild. First, it offers a low barrier to adoption – on modern mobile OSes users need only a few taps to install *Meddle* and once installed there is no maintenance required. Second, *Meddle* explicitly align the goals of researchers and users by offering free and easy-to-use services

that include device-wide ad-blocking, privacy/security filters and parental controls at the network layer – functionality that mobile network providers do not currently make available.

This paper provides the following key contributions. First, we describe the design and implementation of *Meddle*. We demonstrate the scalability and reasonable overheads of tunneling traffic from large numbers of mobile users through our servers. Second, we present measurements results gathered from our initial deployment of *Meddle* comprising XX users lasting XX days. We use data collected from our IRB-approved study to highlight key advantages that our platform offers compared to previous work. Third, we describe several services that we built on top of *Meddle*. Last, we must address the additional security, trust and privacy concerns that arise when tunneling traffic outside of carrier networks into a third-party distributed service. We discuss these issues and others in Section **??**.

## 2. GOALS AND ARCHITECTURE

In this section, we discuss the motivation for *Meddle*, describe our goals for the system and present an architecture to meet those goals.

*Meddle* is motivated by the fact that users and researchers currently have a limited view – if any – into how devices and apps generate network traffic. In addition, there are few mechanisms available for shaping, modifying or blocking this network traffic.

This can be harmful to users, as apps may generate sufficient traffic to cause overage charges on cell phone bills. Even when the volume of traffic is small, high-frequency network activity can quickly deplete battery charge. Further, apps can use the network to leak personal information or track users without consent.

Researchers are in a similar bind. It is common to use testbeds or proprietary ISP datasets to reveal network usage and its impact on data quota, battery life and privacy. However, the coverage and representativeness of these results can by limited by the use of i) a single carrier's network, ii) a small testbed with a single, custom operating system and iii) a single access technology (cellular or WiFi). Once a problem with network traffic has been identified, addressing it can be challenging. OS developers are slow to adopt changes, app developers may be unwilling to modify their code and carriers often have no incentive to deploy new in-network functionality.

### 2.1 System Goals

The two primary goals for *Meddle* are 1) to provide comprehensive visibility into mobile networking traffic and 2) facilitate the development of new solutions and services for mobile networks. We further identify the following sub-goals that address limitations of previous work.

- *Portability.* We want a solution that works regardless of operating system, access technology and provider,

without requiring advanced (and possibly warranty-voiding) techniques such as rooting a phone.

- *Pervasiveness.* For maximum transparency, our system should provide seamless visibility into all network traffic generated by devices. This means continuous monitoring over time and as users move with their devices.

- *Deployability.* Our solution should be easy to use, immediately deployable and incur reasonable costs (or none at all) for users.

- *Isolation.* To prevent existing in-network devices from interfering with these new solutions and services, our system should provide network-traffic isolation.

- *Control.* To enable new solutions and services, our system should enable users and researchers to shape, modify or block network traffic.

### 2.2 Design and Architecture

#### 2.2.1 Design.

To meet the above goals, we propose building a system that redirects all mobile-device network traffic to a server outside the carriers network, thus providing a point of control where one can characterize, modify or block this traffic before sending it to the intended destination. Importantly, we observe that we can do this today without any additional support for devices or carriers: the key idea is to combine software middleboxes (e.g., packet filters, proxies, etc.) with virtual private networks (VPNs).

We use VPNs to achieve our goals of portability, pervasiveness, deployability and isolation. Specifically, major mobile OSes provide built-in VPN functionality for enterprise customers to enable access to resources in the enterprise's private network for employees "on the road". In *Meddle*, we use VPNs as a portable mechanism[1] to tunnel traffic from mobile devices to a machine outside of the carrier's network for the purpose of analysis and interposition.

We use software-middlebox techniques to meet the goal of providing control over network traffic in the mobile environment. Middleboxes are traditionally used in managed networks (e.g., in enterprises and ISPs) to implement policies and enhanced services over IP. In *Meddle*, we use middleboxes as a mechanism not only to implement custom policies and services for users and service providers, but also for measuring networks and experimenting with alternative protocols for the mobile environment without requiring access to mobile carrier networks.

#### 2.2.2 Architecture

---

[1] Andriod, BlackBerry and iOS all support VPNs natively, representing more than 86% of the mobile device market[6].

Figure 1: *Meddle* architecture. Devices use VPN connections to tunnel all traffic to one of potentially many *Meddle* servers dynamically chosen to optimize for performance. Each *Meddle* server uses a device- or user-specific profile to determine the set of services that will operate on the tunnel's network traffic.

Figure 1 depicts an architectural diagram for how we realize this approach via *Meddle*. Devices use VPN connections to tunnel all traffic to one of potentially many *Meddle* servers. We envision that *Meddle* can run as a single-user system in a user's home network or in a hosted data center, or it can run as a shared, distributed and/or virtualized cloud-based service (*e.g.*, on Amazon EC2). The figure depicts the latter, where multiple *Meddle* replicas are available for any given device.

When a device attempts to connect to *Meddle*, we direct it to a nearby *Meddle* server in a similar way to how the Akamai CDN uses DNS to redirect Web clients to nearby content caches [2]. To optimize for performance, the service can send VPN clients to a server that is relatively near the location at which data exits the mobile carrier's network to enter the Internet. This technique can also redirect clients based on server load, reachability and availability.

To meet the goal of providing control over network traffic in the mobile environment, each *Meddle* server uses software-middlebox techniques, or *meddleboxes* to interpose on network traffic. These meddleboxes can (i) implement custom services for users such as packet filtering, caching, and intrusion detection, (ii) monitor network traffic characteristics, and (iii) allow us to experiment with alternative algorithms and protocols for the mobile environment. The system maintains a per-device and per-user profile to determine the set of services that will operate on the tunnel's network traffic.

TODO: Discuss the interior of a meddlebox.

While the *Meddle* architecture is straightforward, there are a number of challenges we must address in building a viable deployment. In the next section, we describe how we achieve continuous monitoring of network traffic across platforms, demonstrate that the power, data-volume and latency overheads are reasonably small.

## 3. IMPLEMENTATION

In this section, we describe our current deployment, we discuss how we addressed several challenges toward achieving the goals listed in the previous section and we present empirical results demonstrating the feasibility of our approach in terms of reasonably low overhead.

### 3.1 Meddle Details

To facilitate widespread adoption, *Meddle* is supported by the vast majority of smart mobile devices (smartphones and tablets) and is easy to deploy on servers. Manually installing a VPN generally requires filling out five fields on a Android phone, and the VPN configuration can be distributed using a single file on iOS.

**iPhone support.** TODO iphone details
**Android support.** TODO android details
**Server-side implementation.** In our current implementation, *Meddle* uses native IPsec, implemented via StrongSwan [16], to establish VPN tunnels and Vyatta middlebox software to shape traffic. Both of these software artifacts are supported on vanilla Linux operating systems, which in turn run on nearly all servers.

### 3.2 Feasibility

Because our approach in part depends on users installing a VPN configuration and tunneling all traffic through *Meddle*, we evaluate whether the cost to the user in terms of performance, power and data quota is sufficiently low.

**Power consumption.** Tunneling traffic to a *Meddle* server requires that all traffic be encrypted by the mobile device. While this is already commonly performed for SSL connections, *Meddle* requires an *additional* layer of encryption. We observed a 10% increase in power consumption when streaming an HD video to Android and iPhone devices using our IPsec tunnel. We believe that this overhead is reasonably low, and we note that this cost for encryption comes with the added benefit of increased privacy from carriers.

An interesting research question is whether it is possible to *reduce* power consumption using *Meddle*. For example, Qian et al [13, 12, 11] found that traffic shaping (a service that *Meddle* provides) can significantly reduce the power consumed by devices when periodic application traffic and radio resource timers are out of sync.

Is it possible to assess the power consumption based on the same experiment as Data consumption? This way we can factorize the explanation. **Data consumption.** 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 keepalive messages,

finding that it ranges from 8–12%. For our measurements we setup *Meddle* as a VPN gateway for an iPhone and Android phone. On each phone we accessed the Internet using the VPN tunnel for about one hour. Our test traffic was generated by activities that we expect to be typical of mobile device usage: Web searches, map searches, online shopping, downloading popular apps, emailing and reading the news. We also uploaded a picture to Facebook and Twitter, streamed a video on YouTube, and played a popular game (Angry Birds).

**Performance.** By forcing user traffic to an intermediate server and interposing on flows, we may add latency both due to additional hops and due to processing time at the *Meddle* server. We envision a DONAR-style deployment where users are dynamically redirected to different *Meddle* servers based on network conditions and server load [19]. Given this model, we evaluate whether we can locate servers near mobile-network egress points using a deployment such as PlanetLab, and found that this is generally the case.

For this experiment, we used data from approximately 10 mobile phones located throughout the US and issued traceroutes from the devices to targets in Google and Facebook's networks. We then used the first non-private IP address seen from the mobile device on the path to a server. We assume that this corresponds to the first router adjacent to the mobile carrier's public Internet egress point. Note that we could not simply ping the device IPs because mobile carriers filter inbound ping requests. Using this set of egress adjacencies, we determined the round-trip time from each PlanetLab site, then took the average of the nearest five sites to represent the case where a host at the nearest site is unavailable due to load or other issues. The average latency to each router was between 3 ms and 13 ms, with a median of 5 ms. Thus, when compared to RTTs of 10s or 100s of milliseconds that exist in mobile networks, the additional latencies from traversing *Meddle* servers is expected to be relatively small or even negligible.

[**TBD: What is the cost in terms of end-to-end performance?**]

[**TBD: Are there cases where we actually improve performance due to detouring? (I've decided not to mention this because it's not a sure thing.)**] In case the carrier is performing traffic differentiation to penalize some bandwidth consuming traffic, using a VPN might significantly improve performance. I don't know whether traffic differentiation is something common on mobile networks. (that was an answer to Dave question. If he discards it, also drop my comment.)

**Scalability.** If wildly successful, we would like to ensure that *Meddle* scales gracefully and that there are sufficient resources to support large numbers of concurrent users worldwide. Based on our initial analysis using StrongSwan on commodity hardware, we found that each connection consumed on average less than 1% of CPU time. Thus, we expect to be able to support up to 100s or small number of

| Protocol | Service | Android | | iOS | |
|---|---|---|---|---|---|
| | | **Flows** | **Bytes** | **Flows** | **Bytes** |
| TCP | HTTP | 14.87 | 74.31 | 16.07 | 82.01 |
| TCP | SSL | 37.19 | 24.35 | 37.64 | 17.31 |
| UDP | - | 40.52 | 0.98 | 42.34 | 0.49 |
| TCP | other | 2.08 | 0.22 | 1.13 | 1.89 |
| Other | - | 5.35 | 0.14 | 2.82 | 0.09 |
| *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.*

thousands of users per server, which is in line with low-end VPN appliances sold by Cisco and Vyatta. A recent study [1] showed that current rates for 3G networks in the US were between 0.59 and 3.84 Mbps; assuming devices are uniformly distributed across carriers, we expect to be able to support 250 simultaneous users (saturating their download capacity) for every 1 Gbps of bandwidth at the server.

## 3.3 Current Deployment

** Dataset description - IRB based study from September 2012 - full packet captured - num of Android, iOS devices - type of device (phone/tablet) - num of service providers - Total volume of traffic

## 4. CHARACTERIZING MOBILE TRAFFIC

### 4.1 Dataset Description

- IRB based study from September 2012. - Full packet capture - Currently serves 20 devices from 14 users, 8 Android, 12 iOS device, 4 iPads, 7 iPhones, and 1 iPod Touch. 15 service providers spread across US and France. - 30.2 GB of Traffic Volume from November 1, 2012

### 4.2 Initial Results

Objective coverage of meddle across access technology and devices

I plan to use these two tables to introduce the results in the following sections.

In Table 1.

We classify the IP flows as either TCP, UDP, or other; all IP flows that cannot be classified as either TCP or UDP are classified as other. We further classify TCP flows as either HTTP, SSL, or other. The SSL flows include HTTPS, IMAP, and other services that rely on SSL to secure the connection between the mobile client and the remote server. TCP flows that are not classified as either HTTP or SSL are classified as other.

In our traces we observe that DNS is responsible for more 93.5% of the UDP flows. The rest of the flows are due to

| Protocol | Service | Wi-Fi | | Cellular | |
|---|---|---|---|---|---|
| | | **Flows** | **Bytes** | **Flows** | **Bytes** |
| TCP | HTTP | 16.91 | 83.52 | 13.03 | 56.14 |
| TCP | SSL | 38.18 | 15.83 | 36.14 | 41.42 |
| UDP | - | 41.43 | 0.49 | 41.56 | 1.67 |
| TCP | other | 1.38 | 0.16 | 1.95 | 0.4 |
| Other | - | 2.12 | 0.01 | 7.2 | 0.3 |
| *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.*

services such as Skype which is not widely used by users in our dataset.

HTTP is the dominant protocol of traffic. SSL flows outnumber the HTTP flows however the bytes per SSL flow is significantly smaller than the bytes per HTTP flow. This is because of the media content from streaming sites such as YouTube and Netflix is streamed over HTTP.

In Table 2.

7.2 of cellular traffic classified as other is an artifact of our user base that includes researchers who tend to troubleshoot connectivity issues using icmp packets.

Though the fraction of flows for HTTP over Wi-Fi and Cellular remain the same, the byte shares are different. This is because Wi-Fi is the prefered medium to transfer media content. The flows that transfer media content typically have more bytes compared to flows that do not contain media content.

SSL traffic is typically compressed - discussion on compression opportunities for HTTP over Cellular networks refer to section ... for more details. We can refer to this section to introduce compression.

The increase in HTTP traffic over Wi-Fi is also because apps such as Google Plus (image backup on Android) allow users to upload their images only over Wi-Fi.

## 4.3 Case Study Device Specific Apps - iOS Push

text from Arnaud on iOS Push. How push works

When a push notification is received, the iOS device connects to port 5223 of a nearby Apple server. For each iOS device in our dataset we logged the timestamp at which these connections were established. From this set of connections we then selected the connections established between midnight at 6 am of the users local time. For each user, we computed the median of time interval between successive connections. We observe this median to be in the range of 190 seconds to 590 seconds. This time interval depends on the different apps for which the user has enabled notifications.

A user can disable notifications for specific time intervals on devices running i0S 6. In our dataset, three devices had notification disabled during different time intervals. During

| Time | Bytes Downloaded | Query String |
|---|---|---|
| 1353489965.97 | 356 | a |
| 1353489966.19 | 321 | aw |
| 1353489966.47 | 300 | awe |
| 1353489966.88 | 301 | awes |
| 1353489967.36 | 302 | aweso |

Table 3: Table

these time intervals we did not observe any traffic serving notifications.

## 4.4 Case Study Longitudinal Measurements - Google Search

We now use Google search as an example of longitudinal studies that are possible using using Meddle.

In Table 3 we present a sample search session on Android 4.0. During this sesssion, we observe that each key stroke by the user resulted in a unique GET query made to a Google server. We observe similar behavior on Android 4.0, Android 4.1, and Android 4.2 [**TBD: confirm**]. In the case of iOS devices we observed search queries in the clear on devices running i0S 5. On devices running iOS 6 we observe that search queries take place over SSL. [**TBD: Privacy similar to desktop sentence.**]

## 4.5 Case Study Filtering Intrusive Traffic

## 4.6 Case Study Traffic Optimization - Compression

## 4.7 Case Study Suspicious Apps

## 4.8 Case Study Interference By Service Providers

## 5. TRANSPARENCY AND CONTROL

Example Apps (highlight things we can do with meddle) - Begin with a sentence/intro-para on using middleboxes to offloading activities and offer device wide services - Middlebox based packet monitoring - cross * possible - passive - real traffic 24x7 - actual users - Network traffic characterization - Longitudinal study of network traffic - undersbehavior of apps - Device wide services - service like packet filtering that is not limited to an app - Ad blocking - Platform for malware detection and blocking - Parental controls - Deployment of new protocols and services - Users can opt in for specific service - Mobile story for services like Free-DOM, CCNs, etc. - service in a VM where users opt in for services - Generic Proxy - Privad - Anti-censorship

## 6. DISCUSSION

Privacy
Acceptable use

ISP objections
Scalability
Pigeonholing traffic for monitoring ISPs
Limitations - only network traffic (nothing on device) - not all traffic is captured - cost?

## 7. RELATED WORK

*Meddle* builds upon two existing technologies: VPNs and middleboxes. In our current implementation, we rely on an IPsec [9] implementation to tunnel traffics to *Meddle* servers. Sherry et al. [14] explore the opportunities enabled by moving middleboxes to the cloud, which includes simplifying management for enterprise network administrators. In contrast, our work focuses on the mobile capabilities software middleboxes enable rather than a redirection architecture for enterprise networks.

*Meddle* provides researchers with a cross-platform, cross-carrier technique for capturing network usage patterns from mobile devices. Previous work used on-demand active measurements to characterize network measurements [18, 15]; however, these measurements are restricted to the point at which users run the tests. Gerber et al. [7] use passive measurements alone to estimate transfer rates in a single carrier's network; *Meddle* will enable such analysis across multiple carriers.

We expect to use *Meddle* to investigate security and privacy issues in the traffic that mobile devices generate. By monitoring and controlling information at lower layers in the software stack, previous work [5, 8, 17] has shown that existing apps leak significant private information. In *Meddle*, we take this downward mobility to an extreme, moving off the end-host entirely and eliminating the restrictions of a lab setting.

The CloneCloud [3] and MAUI [4] projects explored the space of moving functionality from mobile devices into the cloud. By interposing on user traffic, *Meddle* will allow us to explore some of this functionality without requiring device modifications.

** Middleboxes in the cloud - Sekar Hotnets'11, Sekar NSDI'12, Sherry Sigcomm'12 ** Security and Privacy of Apps - Taintdroid, Appfence, Third party tracking NSDI'12, - We can act on it without the need to root the phone - computation is offloaded to the middlebox ** Ads and Analytics for Mobile - Valina-Rodriguez IMC'12, other mobile measurements papers from IMC'12 - Appfence - We not only report activities but also provide an interface that users can use to filter ads and analytics ** User Behavior and Energy consumption - Falling Asleep Angry Birds – Appsensor MobileHCI'12 - Meddle is passive - Who killed my battery - WWW'12 ** Web caching - ATT papers - periodic nature stuff

## 8. ADDITIONAL AUTHORS

## 9. REFERENCES

[1] 3G/4G performance map: Data speeds for AT&T, Sprint, T-Mobile, and Verizon. `www.pcworld.com/article/254888/3g4g_performance_map_data_speeds_for_atandt_sprint_tmobile_and_verizon.html`.

[2] Akamai. Akamai CDN. `www.akamai.com`.

[3] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proc. of Eurosys*, 2011.

[4] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In *Proc. of MobiSys*, 2010.

[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] Gartner smart phone marketshare 2012 Q1. `www.gartner.com/it/page.jsp?id=2017015`.

[7] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman. Speed testing without speed tests: estimating achievable download speed from passive measurements. In *Proc. of IMC*, 2010.

[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] S. Kent and K. Seo. Security architecture for the internet protocol, 2008.

[10] A. Pathak, Y. C. Hu, and M. Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proc. of Eurosys*, 2012.

[11] 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 *Proc. of WWW*, 2012.

[12] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Profiling resource usage for mobile applications: a cross-layer approach. In *Proc. of MobiSys*, 2011.

[13] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing radio resource aloptlocation for 3G networks. In *Proc. of IMC*, 2010.

[14] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making middleboxes someone else's problem: Network processing as a cloud services. In *Proc. of ACM SIGCOMM*, 2012.

[15] Speedtest.net mobile. `www.speedtest.net/mobile.php/`.

[16] Strongswan. `www.strongswan.org`.

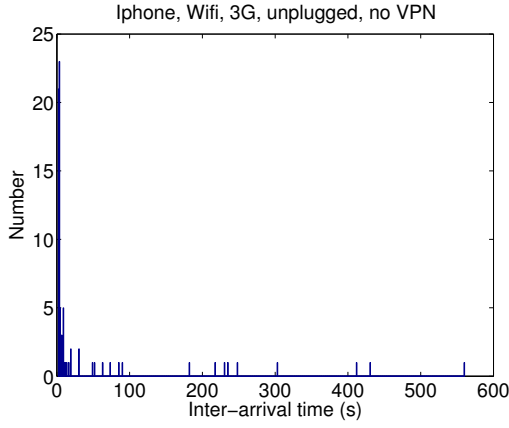[17] S. Thurm and Y. I. Kane. Your apps are watching you. accessed july 19, 2012. `online.wsj.com/article/`
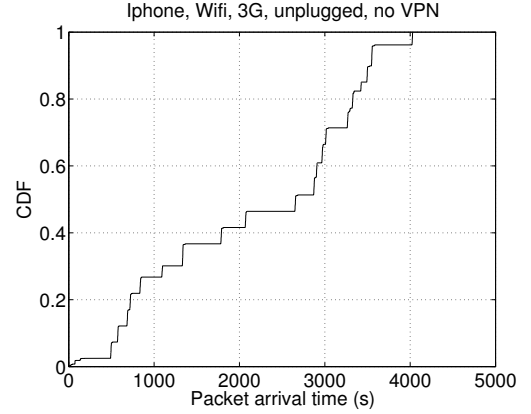
Figure 2: .



Figure 3: .

SB10001424052748704694004576020083703574602.
html.

[18] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *Proc. of ACM SIGCOMM*, 2011.

[19] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford. Donar: decentralized server selection for cloud services. In *Proc. of ACM SIGCOMM*, 2010.

## 10. PUSH NOTIFICATIONS AND POWER MANAGEMENT ON IOS

[**TBD: AL: this section is not intended to be as is in the final report, but just a collection of results that can be incorporated in the final document if needed.**]

All the following experiments are performed on an iphone 4 with iOS 6.0.1. At the beginning of each experiment we close all applications and restart the iphone. The iphone is connected in wifi to a controlled access point on which we perform a tcpdump on the wifi interface and monitor the wifi association between the access point and the iphone. Each experiment lasts for around 1 hour.

In a first set of experiments, we consider a default iphone setting with 3G and wifi enabled, and the iphone unplugged. This corresponds to a typical setting for a user on move.
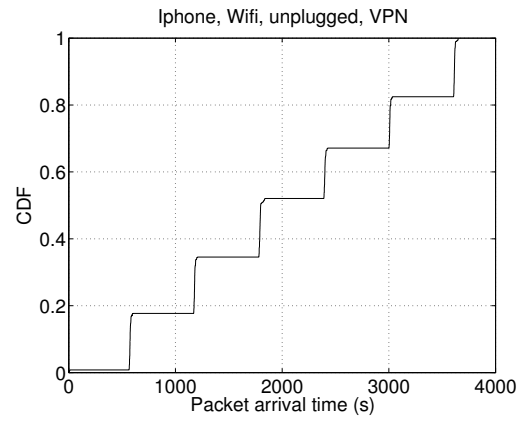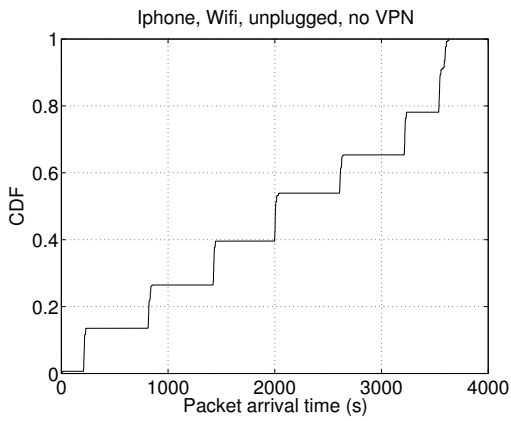


Figure 4: .



Figure 5: .
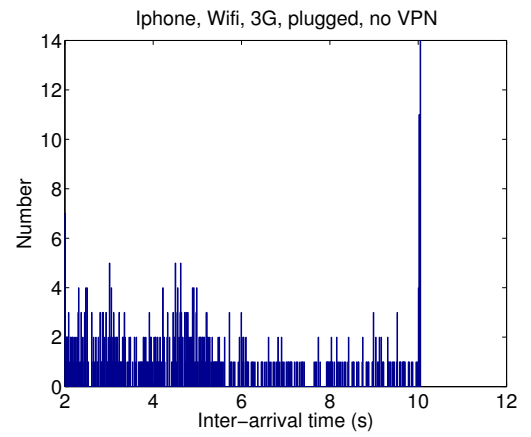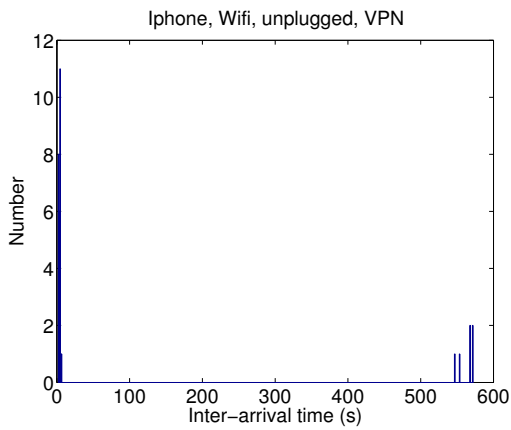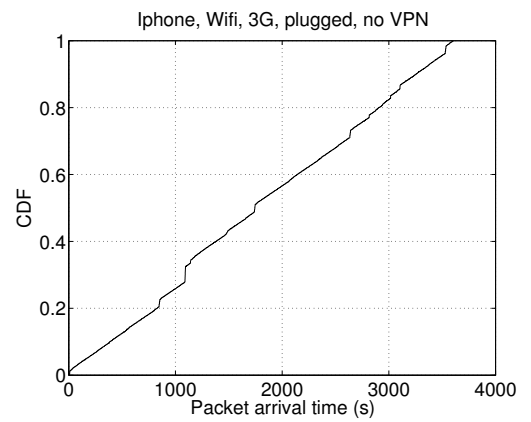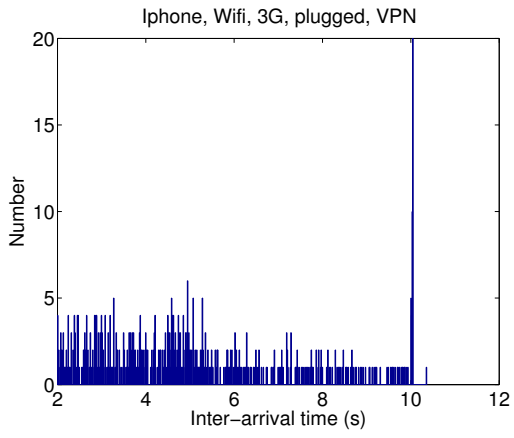
7
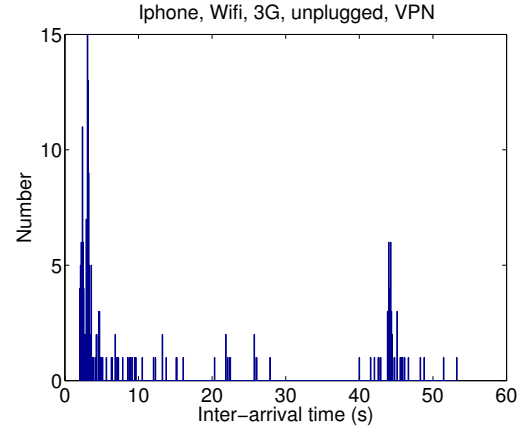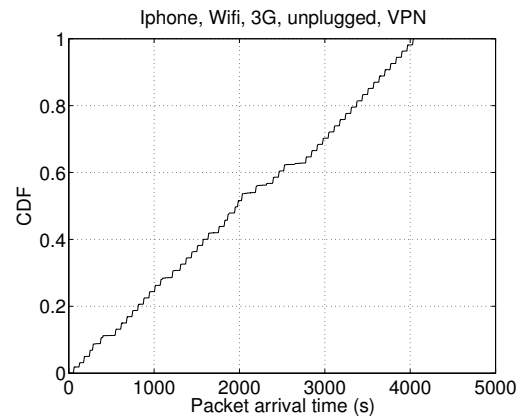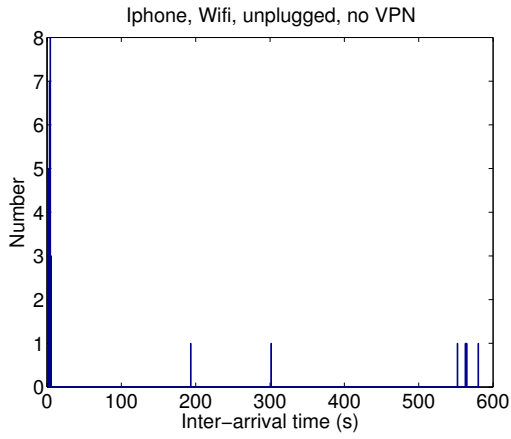
Figure 6: .



Figure 7: .
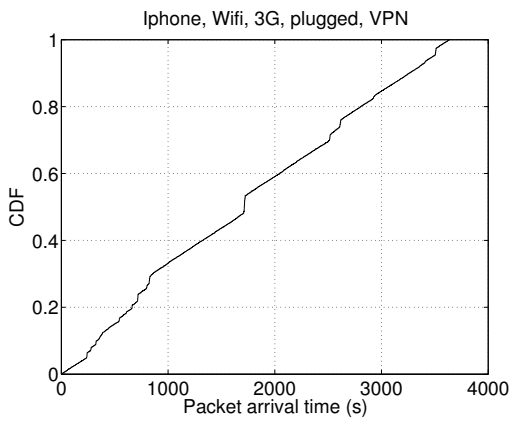


Figure 8: .



Figure 9: .



Figure 10: .



Figure 11: .

Figure 12: .



Figure 13: .