Closing the Loop: Feedback at Your Fingertips

Ionut Trestian, Rahul Potharaju, Aleksandar Kuzmanovic
ionut@northwestern.edu, rpo219@eecs.northwestern.edu, akuzma@northwestern.edu

ABSTRACT

The global network we call the Internet today was started a few decades ago following a very simple model. The entities involved in the beginning had very simple (stated) goals and interaction between them was consequently simple. Technological advances in the recent years have brought exciting new capabilities for Internet users - they can access the Internet from almost anywhere using mobile devices. The entities involved in the functioning of the Internet are however much more complex in current times and their goals are increasingly fragmented and often times hidden. In this paper we elaborate on two such entities: (i) mobile network providers, and (ii) application or service providers. Considering recent signs of tussle and debate between the two entities we argue in this paper that given the obviously more complex nature of the Internet of today, feedback from the end-user is useful and might help decision makers, policy makers settle the network neutrality debate. We further elaborate on our efforts of building an application aimed at several popular mobile platforms that captures feedback about the availability and performance (both measured and judged by the user) of the network, and of Internet services and applications.

1. INTRODUCTION

In the networking research community, the network neutrality debate has been focused mainly on personal computers as affected devices and p2p as the main affected application [12, 14, 9]. Several systems have consequently been developed so far, as a natural response of the research community, that try to capture ISP bias towards p2p flows. The main techniques used to capture such bias consist of conducting either passive measurements [12] or actively sending application probes [14, 9]. The traffic involved being that of p2p applications there is no clear moral line that would distinguish between the two parties (p2p users and ISPs). Indeed, heavy p2p flows can negatively affect available ISP resources and decrease performance for other users. Also the content that users download often times infringes copyright laws. On the other hand the Internet should be regarded by all involved parties as a free network that does not discriminate against individual applications and consequently ISPs should optimize their networks to meet the increase in resource demand of customers. Considering all of the above, systems that passively or actively collect evidences of ISP bias towards p2p applications seem the best solution for this problem as their output mainly points out that such bias exists without effectively taking sides. Sadly, bias shown by Internet providers has proven to be much broader and extensive than pictured above for the p2p case. This bias, as we will elaborate later in this paper, covers different aspects of the wired and mobile Internet. As the mobile environment has, to the best of our knowledge, been overlooked by the research community so far we take on the task of building a system that addresses network neutrality violations for mobile networks and devices.

Designing a system that manages to capture network neutrality violations is a challenging task as witnessed from previous work. Doing so for small power constrained devices such as mobile phones can prove even more challenging. One observation that motivates this work is that computer systems inherently become increasingly complex because of the constant struggle between system designers and malevolent entities. As such, often system designers turn to the end-user as an obvious solution to problems that ultimately affect him. Some examples of problems where the solution can involve also user feedback include: spam (users marking e-mails as spam or legitimate), search (users bumping up or down search results), content rating (users rating content relevance or quality) etc. In this paper we argue that as it is clear that the end-user is also one of the parties affected by mobile operator bias, the users are the ones that can ultimately help solve the network neutrality debate by closing this loop that links users, ISPs, and application or service providers in a natural way, by offering feedback.

Our main contributions in this paper are the following. (i) We bring into the focus of the research community the recent mobile network neutrality debate and argue that user input in the form of explicit feedback or network performance tests might help shed light onto it. (ii) Following on the previous point we describe our development of a mobile application (developed for several popular mobile platforms) that aims at capturing user feedback with regard to the availability and performance of the network, and of Internet services and applications. In order to validate the explicit feedback provided by the user the application performs also a series of lightweight measurements between our servers

and the mobile device.

This paper is further structured as follows. In Section 2 we discuss on recent cases of mobile operator bias as captured by the public opinion and the media. In Section 3 we present current views and arguments for using user input in computer systems. In Section 4 we present the architecture of our system. Related work is presented in Section 5 and we present our conclusions and future work in Section 6.

2. MOBILE OPERATOR BIAS

One of the clearest cases of mobile operator bias can be seen in the recent blocking of the Skype streaming application by T-Mobile in Germany [1]. Skype is one of the most popular VoIP applications in the world. It is available on various platforms including the Apple iPhone. T-Mobile is the exclusive reseller for the iPhone in Germany. T-Mobile publicly declared that it will block Skype both contractually (clients who use the application will be cut off from the service) and physically (Skype use will be detected and blocked by network methods) on both iPhone and Blackberry devices. The reason that they have cited is that the high amount of traffic generated by Skype users will thwart network performance. Skype has publicly responded to this [3]. They have conjectured other reasons for the blocking (Skype usage would drop T-Mobile revenues).

Also as of March 31st, several amendments have been made to the Telecommunications Package by the European Parliament [7] that give discretionary powers to providers to prioritize and block certain applications and further legitimize abuses of the type previously mentioned. These amendments have been met with disbelief and various agencies have since filed petitions such as the one previously cited.

Mobile operator bias is not limited to a few disparate cases. It is extensive and it comprehensively covers multiple aspects of user activity on the mobile Internet: search [4], social networking [6], VoIP [1], streaming [5]. The motivations behind operator bias can be classified into two categories:

Aggressive resource use. Applications make use of various network resources for their functioning (*i.e.*, bandwidth, messaging services etc). Popular applications with a consistent userbase and network operators often find themselves on conflicting positions as the goal of the network operator is to offer satisfactory service to all network users, a service which sometimes degrades because of the users of the popular application. One example of this type of bias is given by the recent T-Mobile blocking of the Twitter social networking application. T-Mobile has blocked Twitter mainly because Twitter users send SMS messages to the service [6]. As SMS uses a mobile network control channel a big inflow of such messages will also affect the functioning of the mobile network. If one is to accept the reasons mentioned by T-Mobile for the blocking of Skype, the blocking also falls under this category however one cannot ignore the obvious economic ramifications that this action has either.

Rival application or service support. Operators can have conflicting interests with the applications that carry traffic on their networks. They sometimes favor a different service [5] or they regard the application as a competitor to their own services [1]. In all cases it is the end-user that suffers since he cannot use a more favored application or a cheaper alternative to the services offered by the operator.

3. INVOLVING THE END USER

Involving users in computer system design is not new to this paper. In fact a variety of systems (even networked) incorporate user characteristics into their design in order to enable new applications or meet scaling demands. Most such systems either try to model the user or predict him. Some examples include: capturing predictable actions in order to prefetch content, operation scheduling that minimizes user wait time etc. At most, systems either try to model the user by using utility functions and considering all users canonical or try to guess individual user actions by looking at bursts of traffic or proceeding action patterns.

More recent work surprisingly shows that user feedback can also be used to reduce computer power consumption [11] or reduce latency in remote display systems [10].

4. SYSTEM ARCHITECTURE

Systems which address network neutrality in wired networks are inherently different from systems that are constrained to work on small power constrained devices such as mobile phones. In this section we outline the trade-offs involved in designing such a system and the design choices we consequently made.

We first describe what a generic implementation on a regular computer would consist of and after that detail limitations of mobile platforms and the corresponding trade offs we made.

4.1 Underlying Architecture

While the scope of developing a feedback driven application is more for desktop systems, it is pretty narrow for mobile platforms. For instance, whereas a desktop application can run on a similar chain of wide operating systems with little or no modifications, the same is not true for mobile phones due to their differences in the development tool chain. The underlying architecture irrespective of the platform at which the application is being targeted, however remains similar. Data is collected according to two mechanisms:

- Periodic: The activity is logged every n seconds of elapsed time.
- Events: The activity is logged whenever the user performs a pre-determined set of actions.

The **Application Level Logger** is mainly intended to record any application specific data or system wide data that might be of use. We collect data such as time spent in execution, virtual memory size, page faults per minute and various other metrics depending on the capabilities provided by the underlying operating system. Using the two collection mechanisms, we enumerate and record all processes that are running every 30 seconds. Measuring this at the end-users side is attractive for a couple of reasons: It is simple and provides access to measurements only accessible at the host side. We also leave to the user an option for providing feedback in case the current networked applications don't perform at the expected level.

One of the ways of collecting data is by employing a **Browser Plugin**. The browser plugin has a wider variety of uses depending on the browser in question and the mobile platform on which it is running. Because we focus mainly on user feedback, we intended to keep the application front-end simple so it provides a facility to submit the reason for the user's annoyance. In the background, we measure a lot of other parameters unrelated to the network to help us characterize any false-positives that we might encounter.

Because we target mobile users, we needed a mechanism to allow us to handle any potential bugs that we might encounter in the future. Thus, one of the primary design decisions we made was to include an **Automatic Updater Component** in our client-side core architecture. This component periodically contacts a stable server hosted on Google's M-Lab to check if an update is present. The updates are divided into two types: critical updates and normal updates. If the updater finds that a critical update is available but fails to update the local application, it immediately stops all logging activity until the application is updated. For a normal update, however, the updater follows the conventional way of applying the update whenever it senses idle activity. The task of notifying the automatic updater is taken care by the Update Manager component on the server-side.

It is common with systems that passively or actively collect evidences of ISP bias towards an application consume a considerable amount of bandwidth. In fact, there is a trade off between the level of application monitoring one can get and the network bandwidth the user is willing to sacrifice in order to support such high levels of monitoring. Our current system periodically checks the log file size and once it reaches a specific size, an **Uploader** component attempts to upload it to the Google M-Lab server. It has a retry timer that attempts to resend the file in case it fails in its initial attempts.

One of the most important component of such a system is the **Network Profiler** which captures the network level activity both as a timer based event (periodic capture of inbound and outbound flows) and an activity based event (whenever the user is annoyed about something). Depending on whether or not a libpcap port is available to the specific platform, we change our implementation strategy. For instance, as will be further discussed, there is no stable port of libpcap available to Windows Mobile so we had to resort to data could be captured through the browser plugin. We log the connection start time, connection establishment time, connection finish time and the inbound and outbound bytes in a compact netflow format. Finally, we perform simple network measurements using the operating system's implementation of ping and traceroute.

Mobile device implementation considerations. On the server-side, we differentiate each mobile user by a unique global identifier known as an *IMEI* (International Mobile Equipment Identity) number which is analogous to an Open ID and is guaranteed to be globally unique i.e. no two mobile phones in the world can have the same IMEI number. Each log file that is uploaded to the server is named after the IMEI number of the device so that they can be separated at the time of inserting into the database on the server-side.

4.2 Implementation on Windows Mobile

In this section we detail our current effort of implementing the application on Windows Mobile devices.

4.2.1 Description

Windows Mobile is an operating system targeting at devices that are based on the Microsoft Windows API. We developed our application for Windows Mobile 5 devices. We chose this operating system because of its *Persistent Storage* capability which stores data in flash memory. The main advantage with this is that no data is lost (which is apparently very crucial for us to achieve our goal) even when the device loses power. Though there are several development options to develop applications for these devices, we use Microsoft's .NET Compact Framework to develop our application mainly because:

- It is a subset of the .NET framework and hence shares many components with software development on desktop systems
- It provides an emulator to test the application with minimal difficulty. We used a combination of Visual C# (for designing the UI Front-end, Application Level Activity Logger, and Log File Uploader)

and Visual C++ (for implementing the browser plugin using Native C++ code). At startup time, the application starts in the background and continues running until the user interrupts it with an annoyance event.

In addition to the above mentioned points, because Windows CE uses Win32 API, some Windows code can be recomplied for Windows CE. However, the display and memory properties are significantly different. The use of multithreading/multitasking significantly increases the potential for the applications being developed.

4.2.2 Limitations

The Windows mobile device by itself did not have have any serious limitations as far as our application development was concerned. However we did observe that the color screens and the powerful CPUs drained significant power from the battery thus making the device exhibit a shorter battery life.

4.2.3 Design Decisions

Though it is likely that most Windows code will work on Windows CE without modifications, we did run through some issues. For instance, there was no easy way to write the application startup code in Visual C++ so we had to resort to using native code. *libpcap* is a popular library for sniffing packets at the network level. At the time of writing this paper, there was no proper port available that worked with the Windows CE TCP/IP stack so we had to omit the Network profiler module from our application. Instead we resorted to writing down most functionality as an Internet Explorer plugin. Because Windows CE supports multithreading, we managed to run our application as a background task, thus not interfering with the user operations unless it is really needed.

4.3 Implementation on the iPhone OS

In this section we are describing our future work regarding a port to the iPhone OS.

4.3.1 Description

The iPhone OS was built as a derivative of the MAC OS X and has four abstraction layers: Core OS layer, Core Service Layer, Media Layer and the Cocoa Touch Layer and runs on devices that are based on an ARM processor and uses OpenGL ES 1.1. The development of applications for the iPhone was made popular with the release of the device's SDK in the early 2008. It allows developers to make applications for the iPhone and iPod Touch, as well as test them in an iPhone simulator. XCode is the preferred tool chain of development. Most of the development code is written in Objective C which is a simple programming language that enables complex object oriented programming.

4.3.2 Limitations

There are a number of advantages and disadvantages with using iPhone to achieve our goal:

- Application development involves a hefty development fee (and involves various catches like penalties etc.).
- Applications should be approved by Apple before they can be released into the AppStore which is the default application repository from which applications are downloaded and installed on the iPhone.
- iPhone does not support backgrounding of applications. Because of this, it is not possible to design any type of monitoring applications. Thus, our application will monitor activities and network level flows only when installed on a Jailbroken iPhone. On a normal iPhone, the monitoring application is pretty much limited to collecting user feedback.
- There is no official development tool chain available for Windows or Linux. One needs to have a MAC OS X to develop applications for iPhone.

4.3.3 Design Decisions

Other than Apple's application deployment limitations described above, we do not foresee any design limitations for this device. Though the learning curve is considered pretty steep for the iPhone, we will try to port most of the components to the iPhone.

4.4 Implementation on the gPhone OS - Android

In this section we are describing our future work regarding a port to the Google Android.

4.4.1 Description

Android is an operating system that runs on gPhones and is based on the linux kernel. Most of the development is done by writing managed code in Java, controlling the device via Google-developed Java libraries. The Android SDK includes a comprehensive set of development tools which include a debugger, libraries and a handset emulator.

4.4.2 Limitations

While we initially thought of Android to be possessing similar capabilities as that of Windows Mobile, there were some basic differences:

- Though it uses a Linux kernel as its operating system, it is not a conventional Linux distribution and thus does not have a native X Window System, nor does it support the full set of standard

GNU libraries like its system libraries like GNU C Library). This specific modification makes it difficult to reuse existing Linux applications or libraries on Android.

- Android does not use established Java standards, i.e. Java SE and ME which prevents compatibility among Java applications written for those platforms and those for the Android platform. Android only reuses the Java language syntax, but does not provide the full-class libraries and APIs bundled with Java SE or ME.

4.4.3 Design Decisions

Like the iPhone, because the gPhone uses a Linux kernel, we consider it feasible to port our application. We are still unsure of any issues that might turn out with the Network Profiler component.

5. RELATED WORK

Recently networked systems have started incorporating user feedback into their functioning for tasks such as determining spam e-mails, improving search relevance, content rating etc.

A variety of other systems rely on user input to optimize their functioning. HomeMaestro is such an example [13]. It aims at providing application fairness for a household based on a set of application level weights. The likely input for these weights is the end-user as the authors note.

OneClick [8] has been designed as a system which uses user feedback (by pressing a key when the user is dissatisfied) when using network applications. The system uses qualitative measures to quantify the Quality of Experience of a single user. As such it is orthogonal to our design since we aim at capturing repeated bad experiences for multiple users of a mobile network.

HerdictWeb [2] uses user feedback to give verdicts of website availability in given regions. It functions as a Browser extension which allows users to report unavailable websites. The extension doesn't record response times and browsing events such as errors so it is hard to validate the feedback given by users to actual availability events. Such a feedback model (browser toolbar) would not function on a mobile phone where browsers are minimal. Our solution, a standalone application which also records events in a lightweight manner is a better fit.

6. CONCLUSIONS AND FUTURE WORK

This project is aimed at addressing mobile network neutrality violations from the part of ISPs. As we detailed in the paper, our personal belief is that the device users should actively participate in the network neutrality debate by providing feedback concerning their preferred applications and services.

Our future work includes the port of our system to other mobile platforms and addressing the security aspects of the functioning of the system. As we expect our system to become popular we are dealing with harnessing the M-lab infrastructure to meet system scaling demands.

7. REFERENCES

[1] German carrier t-mobile blocking skype. `http://www.washingtonpost.com/wp-dyn/content/article/2009/04/01/AR2009040101124.html`.

[2] Herdictweb. `http://www.herdict.org/web/`.

[3] Is deutsche telekom playing an april's fool joke at the expense of skype users in germany? `http://share.skype.com/sites/en/2009/04/is_deutsche_telekom_playing_an.html`.

[4] Mobile giants plot secret rival to google. `http://www.telegraph.co.uk/finance/migrationtemp/2803788/Mobile-giants-plot-secret-rival-to-Google.html`.

[5] Mobile video, net neutrality, and verizon wireless. `http://www.dslreports.com/shownews/Mobile-Video-Net-Neutrality-and-Verizon-Wireless-79989`.

[6] Net neutrality outrage: reports of t-mobile blocking twitter. `http://blogs.zdnet.com/ip-telephony/?p=2877`.

[7] Opennet coalition open letter to the european parliment. `http://www.assoprovider.it/index.php?option=com_content&task=view&id=213&Itemid=46`.

[8] K.-T. Chen, C. C. Tu, and W.-C. Xiao. Oneclick: A framework for measuring network quality of experience. In *Proceedings of IEEE INFOCOM 2009*, 2009.

[9] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi. Detecting BitTorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC'08)*, Vouliagmeni, Greece, October 2008.

[10] J. R. Lange, P. A. Dinda, and S. Rossoff. Experiences with client-based speculative remote display. In *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, Berkeley, CA, USA, 2008.

[11] A. Shye, Y. Pan, B. Scholbrock, J. S. Miller, G. Memik, P. A. Dinda, and R. P. Dick. Power to the people: Leveraging human physiological traits to control microprocessor frequency. In *MICRO '08: Proceedings of the 2008 41st IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, 2008.

[12] M. Tariq, M. Motiwala, and N. Feamster. NANO: Network Access Neutrality Observatory. In *Seventh ACM Workshop on Hot Topics in Networks (HotNets-VII)*, Calgary, Canada, October 2008.

[13] Thomas Karagiannis and Elias Athanasopoulos and Christos Gkantsidis, and Peter Key. HomeMaestro: Order from Chaos in Home Networks. In *Microsoft Research Tech. Rep. MSR-TR-2008-84*, 2008.

[14] Y. Zhang, M. Mao, and M. Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *Seventh ACM Workshop on Hot Topics in Networks (HotNets-VII)*, Calgary, Canada, October 2008.