BonaFide: A Traffic Shaping Detection Tool for Mobile Networks

Vitali Bashko, Nikolay Melnikov, Anuj Sehgal, Jürgen Schönwälder
Computer Science, Jacobs University Bremen
Campus Ring 1, 28759 Bremen, Germany
{v.bashko, n.melnikov, s.anuj, j.schoenwaelder}@jacobs-university.de

***Abstract*—With the growth of the mobile Internet customer base we expect that mobile operators may feel tempted to apply certain traffic shaping techniques. Mobile operators may view such traffic shaping techniques as a quick and a cost-effective way of ensuring more "equalized" Quality of Service (QoS). A situation when a more bandwidth hungry application is tamed for the sake of other protocols and applications. As such, customers Voice over IP (VoIP), YouTube or BitTorrent traffic can be shaped or blocked by certain operators. We developed a tool that can detect such traffic shaping manipulations on a mobile network. We also demonstrate that certain traffic manipulations occur both on 3G as well as EDGE networks.**

I. Introduction

In recent years, many studies have been conducted [1][2][3] and tools developed [4][5][6] to measure network performance under different traffic shaping policies deployed by Internet service providers (ISPs). The target of all these studies, and tools, have been desktop devices connected to the Internet, but ever since the advent of smartphones, mobile Internet consumption has been steadily increasing. Some studies show that the global share of mobile Internet traffic has risen from just about 1% in August 2009 to over 11% in August 2012 [7] and has been showing over 450% year-on growth. This coupled with other studies that predict the number of mobile Internet users surpassing that of desktop Internet users by the end of 2014 [8] means that mobile network operators (MNOs) have had to deal with a huge growth in demand in a very short time.

Since smartphones can typically be used to perform data-intensive tasks like watching videos online and sharing pictures, it is quite likely that heavy users within a cell could degrade the experience of others located within the same cell. Furthermore, the convenience of using instant messaging services and voice over IP (VoIP) on a smartphone to reduce expenditure can quickly reduce revenue of operators as well. As such, it is possible that faced with a large increase in demand and reduction of traditional revenue sources, within a network channel plagued with low-reliability, lack of bandwidth and high saturation in cities, MNOs would deploy traffic shaping policies in order to improve overall customer experience and protect revenue.

With such an active community of mobile Internet users, it is important to perform network measurements from smartphones as well. However, tools designed for desktop devices are not suitable in their current form since they either use too much bandwidth and mobile Internet access is usually metered. Some tools are also not suitable for the smartphone OSes being utilized. Furthermore, the existing network measurement tools for smartphones only report basic network information (e.g., local and global IP addresses of the mobile device) and performance (e.g., downlink/uplink throughput, latency, round trip time). These factors have left the traffic shaping situation in mobile networks virtually unexplored.

In this paper we present BonaFide, a traffic shaping detection tool that is an adaptation of Glasnost [1] , designed to run on the Android OS. This tool can be used to discover instances of traffic shaping in mobile networks and answer questions such as:

1) Do mobile network operators apply traffic shaping techniques based on the deep-packet inspection in order to restrict the performance of "unwanted" applications? For example, an operator might restrict VoIP applications on the network.

2) Does the performance of certain application protocols in mobile networks depend on the time of the day? Can we observe the difference in applications' performances between day time and night time? Do mobile operators perform application-dependent traffic shaping policies during peak hours, when the number of concurrent users is larger than usually?

3) Is network access throttled for heavy data consumers on the network?

Detecting the above traffic shaping situations is interesting to end-users and regulators. It is important for both these parties to evaluate ground-truth on network neutrality issues, and whether a provider is delivering the agreed upon services. Furthermore, it is also important to end-users to be able to evaluate which providers in their area might deliver better service than others.

Section 2 of this paper presents related work in the area of Internet measurements and traffic shaping detection. This is followed by an overview of the traffic shaping detection methodology employed by BonaFide in Section 3. Information regarding testing and verification of the tool is provided in Section 4 and some interesting results obtained from experiments carried out in mobile networks are presented in Section 5, followed by a conclusion.

II. Related Work

Since network performance metrics are of interest to network operators, end users and regulators, there are a few

978-3-901882-50-0

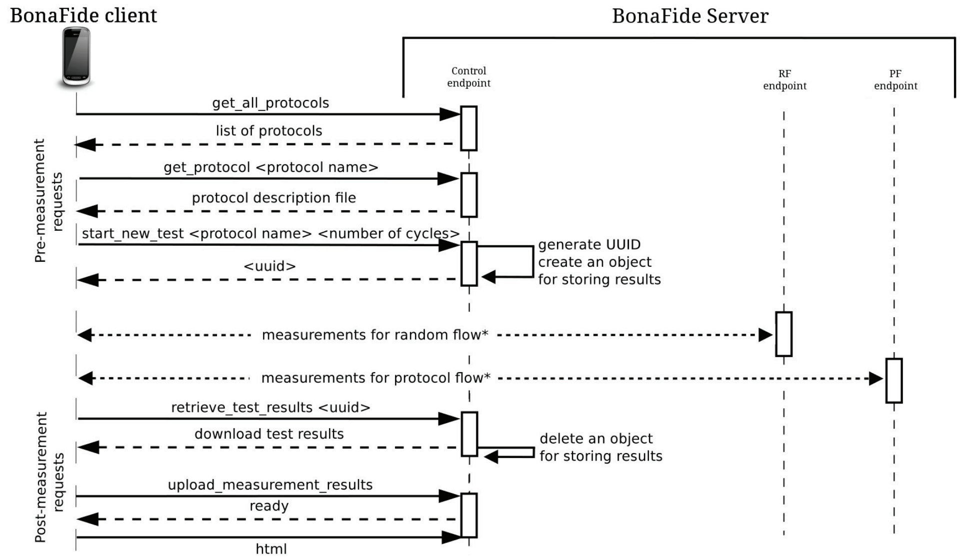

Figure 1. BonaFide operation ; RF - random flow; PF - protocol flow

large-scale infrastructures that have been designed to gather network performance statistics. Measurement Lab (M-Lab) [9] is an open, distributed server platform for researchers to develop, test, and deploy new active measurement tools. The goal of M-Lab is to advance network research and empower the public with useful information about their broadband connections. Currently, instead of focusing on the Internet core, M-Lab focuses on measuring the end-to-end performance and on the characteristics of broadband access links. Measurements capture basic operational characteristics (e.g., TCP throughput, available bandwidth), advanced host diagnostics (e.g., misconfiguration, small socket buffer sizes), and ISP traffic management practices (e.g., BitTorrent blocking, traffic shaping).

Another interesting infrastructure for gathering network performance metrics is SamKnows [10]. Their platform is built around specialized hardware called Whiteboxes, which are consumer grade, home Wi-Fi routers with additional testing software integrated. These can be deployed onto the home network in order to collect a range of metrics. One of the focuses of SamKnows is to characterize end-user experience, rather than just collect metrics about latencies and throughput. As such, unlike many other measurement tools and infrastructures, SamKnows also keeps track of the performance of web browsing (total time taken to fetch a page and all of its resources), video streams (the initial time to buffer, the number of buffer under-runs and the total time for buffer delays) and VoIP (upstream packet loss, downstream packet loss, upstream jitter, downstream jitter, round trip latency). These are, of course, in addition to standard metrics like packet loss, DNS resolution time, latency and throughput.

Similar to SamKnows in some ways, the RIPE Atlas project [11] is a distributed Internet measurement network consisting of thousands measurements nodes, called probes, placed all around the world and connected to a controlling framework. The main goal of RIPE Atlas is to take active measurements in a coordinated fashion, thereby supplying more measurement data for the benefit of the ISPs and research community. Currently the RIPE Atlas measurement system executes built-in measurements, such as ICMP ping to predefined destinations (measuring round-trip time), traceroutes, uptime and DNS (anycast) measurements.

While all these projects provide decent measurement infrastructure and tools for traditional computing devices, none of them are designed to function with mobile networks or on mobile devices. On the other hand, there are a few tools available for mobile devices that can measure network performance from a user's perspective. The Network Diagnostic Tool (NDT) [12] is an application for Android-based smartphones for running network speed and diagnostic tests. An NDT test reports the upload and download speeds, in addition it also attempts to determine what, if any, problems limited these speeds, differentiating between computer configuration and network infrastructure problems. Fing [13] is a multiplatform toolkit, which performs service scans (TCP port scan), hosts availability detection, traceroutes, TCP connection testing and DNS lookups. MobiPerf [14] is a handy mobile network measurement tool designed to collect network performance information (e.g., downlink/uplink throughput in kbps) and network policies (e.g., testing which ports are blocked by the cellular ISPs). A few other tools also exist [15], [16], [17], [18], but none of them are designed to perform traffic shaping detection, which, given the limited resources in mobile networks and large number of users, is quite interesting to study.

Glasnost [1] is one of the most popular tools for traffic shaping detection. It can be used to test ISP implemented throttling or blocking of different application-layer protocol traffic, like P2P protocols (including BitTorrent, eMule), HTTP traffic, SSH transfer, Flash video and others. Glasnost is based on a client-server architecture, where the client connects to a Glasnost-server to download and run various tests. Each

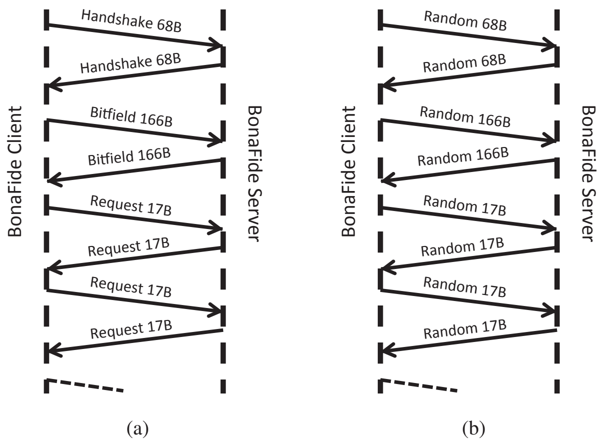

Figure 2. A pair of flows used to detect BitTorrent traffic differentiation. Subfigure (a) Shows the protocol flow and (b) is the random flow.

test measures the path between the client and the server by generating flows that carry application-level data, which are constructed to detect traffic differentiation along the path. However, Glasnost cannot be used in mobile networks via smartphones because the Glasnost-client is implemented using Java Applets, which find limited support amongst smartphones. Furthermore, since Glasnost was not designed to work with mobile networks, a single test can consume more than 100 MB of bandwidth, which is too high for mobile Internet users.

III. Shaping Detection Methodology

BonaFide has a client-server architecture. A client connects to the measurement server (see Fig. 1) and retrieves (`get_all_protocols`) the list of available application protocols to test. The client then selects a single protocol and downloads the corresponding application protocol description file from the server side (`get_protocol`). Each protocol description file provides a set of rules (see Section III-B), which define the client's and the server's behavior during the measurement test. Once the client downloads a particular protocol description file, it can start a measurement test that checks whether the current ISP deploys traffic differentiation for the selected application protocol or not.

The core idea behind the traffic shaping detection measurement test is the emulation of a pair of flows that are identical, except in one respect that should trigger traffic differentiation along the path. In the context of this paper, when we talk about the flow, we mean the sequence of packets that are exchanged between the server and the client sides in both directions within the same TCP connection. The performance of a flow implies the application goodput during the lifetime of that flow. Goodput is the number of useful information bits, delivered by the network to a certain destination per unit of time. We distinguish two components: downlink performance and uplink performance. The first component denotes the application goodput in download (from the server to the client) direction, and the second component denotes the application goodput in upload (from the client to the server) direction respectively. Measuring and comparing the performance of those two flows helps to determine whether content-based traffic shaping methods were applied or not (see Sections III-C and III-E). Since BonaFide injects custom packets into the network during the measurements it can be classified as an active measurement tool.

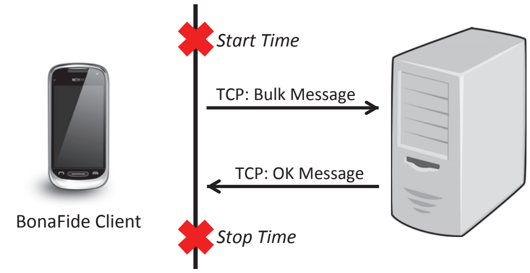

Figure 3. Scheme of uplink bandwidth estimation

Let's consider an example of constructing a pair of flows that can reveal the presence of traffic shaping of a BitTorrent application protocol along the path (see Fig. 2). The left figure corresponds to the tested protocol flow. The client opens a TCP connection to the measurement server (Fig. 2 represents only the application layer protocol messages, hence the TCP handshake is not shown) and starts sending packets that implement the BitTorrent protocol. In this case, the payload of the packets carries BitTorrent protocol headers and content. The server in its turn responds with packets that conform to the BitTorrent specification. Once this initial handshake is completed, the goodput measurements are carried out by sending TCP bulk messages in the direction currently being tested.

The packet exchange on the right side in Fig. 2 corresponds to the random flow. The client opens another TCP connection and sends packets of the same size, but in this case the payload contains randomly generated data. Once again, after the depicted handshake is completed, goodput measurements are carried out using TCP bulk messages.

Two flows traverse the same network path and have the same network-level characteristics. As a result an ISP that differentiates BitTorrent traffic would cause an impact only on the first flow, and keep the performance of the random flow untouched. Thus, significant differences in those two flows' performances are likely to be caused by traffic manipulation along the path. If the ISP completely blocks BitTorrent traffic, it also would be noticed because one of the participating sides (client, server or both) will receive a timeout. And the flow with random data will successfully finish a measurement cycle.

The presented traffic shaping detection technique is similar to that used in the Glasnost [1] project. The benefit of this technique is that we are running an active measurement test, thereby having complete control of the measurement test lifecycle (we can repeat flows with different properties like payloads or port numbers). As a drawback, we need to generate and inject additional data into the network, which in case of mobile networks (EDGE, HSPA) can still be relatively expensive.

A. Protocol Definition Files

Since only flows carrying packets that conform to an application protocol of interest trigger traffic shaping, protocol definition files are used to define a set of rules that specify protocol headers and payloads. The proposed protocol description file format is very similar to the format used by *Glasnost* [1]. However, the *Glasnost* format was not used in BonaFide since the measurement test lifecycle in our case is different, owing to the peculiarities of mobile networks.

Each protocol description file defines the name of the application protocol, port numbers that should be used during the measurement tests, well-known port numbers that might be officially or unofficially assigned to a particular application protocol, and a sequence of commands that tells how to construct a payload which conforms to a protocol specification. A sample of the HTTP protocol description file is presented below:

```
protocol HTTP
PFport 30008
RFport 31008

request string("GET /wiki/Jacobs_University_Bremen
HTTP/ 1.1") byte(13) byte(10) string("Host:
en.wikipedia.org") byte(13) byte(10)
string("User-Agent: Mozilla/5.0 (Linux; U; Android
2.3.5; en-de; HTC Desire S Build/GRJ90) AppleWebKit/
533.1 (KHTML, like Gecko) Version/4.0 Mobile
Safari/533.1") byte(13) byte(10) string("Accept:
text/html") byte(13) byte(10) string("Connection:
close")
byte(13) byte(10) byte(13) byte(10)

response string("HTTP/1.1 200 OK") byte(13) byte(10)
string("Server: Apache") byte(13) byte(10)
string("Content -Language: en") byte(13) byte(10)
string("Content-type: text/html; charset=utf-8")
byte(13) byte(10) string("Content -Length: 20")
byte(13) byte(10) byte(13) byte(10)
string("12345678901234567890")
```

The first line defines an application protocol name which is visible to an end user. The two following properties (*PFport* and *RFport*) define the TCP port numbers which are used to communicate with a server during the measurement tests. The destination port number defined by the *PFport* field is used by a client to establish a TCP connection for sending a flow that carries application protocol data. The *RFport* field defines the destination port number used to send the flow with randomly generated data. Finally, the protocol description file contains a set of *request* and *response* instructions that define how to build the application protocol headers and payloads. In the example above, the *request* command tells a sender (might be the client or server component depending on which direction we are measuring at the moment) to send an HTTP request to retrieve the page about the Jacobs University Bremen from Wikipedia. The responder side sends back an HTTP 200 OK response that contains 20 bytes of payload.

The following functions can be used in a protocol description file to construct the application protocol header and payload:

1) `string(argument)` function appends to a message buffer a sequence of bytes that encode a given string argument;
2) `byte(value)` function appends to a message buffer the given byte value (this argument must be in range from 0 to 255);
3) `repbyte(value, N)` function (is not presented in the example above) appends given byte value to a message buffer exactly N times (N must be a positive integer value).

The protocol description file might contain more than one pair of *request* – *response* commands. The only constraint is that for every *request*, exactly one *response* must be defined immediately after the *request* definition.

B. List of supported protocols

At present BonaFide supports six different application protocols: HTTP, FlashVideo (YouTube), Session Initiation Protocol (SIP), Real Time Streaming Protocol (RTSP), BitTorrent and VoIP H323.

The *VoIP H.323* [19] standard addresses call signalling and control, multimedia transport and control, and bandwidth control for point-to-point and multi-point conferences. It is widely deployed worldwide by service providers and enterprises for both voice and video services over IP networks.

The Session Initiation Protocol (SIP) [20] is a signaling protocol for controlling voice and video streams over IP networks. There are a number of popular SIP clients for Android-powered smartphones that enable VoIP calls over the Internet (*SIPDroid*, *Linphone*, *Fritz!App*, *CsipSimple*, etc.).

According to the Google Play statistics BitTorrent applications are more popular than other Peer-to-Peer (P2P) applications (like eMule or Gnutella). It is highly unlikely that end-users will use the P2P clients on mobile networks, because mobile networks are generally more expensive. Several mobile Internet providers (e.g. in Germany: *NettoKOM*, *O2*, *Congstar*) claim that they do not support P2P traffic on their networks. It would therefore be interesting to find out whether they perform deep packet based traffic shaping for the BitTorrent protocol.

The Real Time Streaming Protocol (RTSP) [21] is a network control protocol designed for use in entertainment and communications systems to control streaming media servers. The protocol is used for establishing and controlling media sessions between end points. Like HTTP, RTSP uses TCP to maintain an end-to-end connection. Some popular applications (Winamp, Spotify, VLC media player) use the RTSP for controlling the audio streaming.

Finally, we created protocol description files for the HTTP and the Flashvideo protocols. In our opinion, it is interesting to run an experiment for these application protocols, since according to the Cisco Visual Networking Index (VNI) global mobile data traffic forecast [22] the mobile video traffic and the web traffic represent more than 90% of the overall mobile traffic.

We can speculate that some of these application protocols can be viewed as deteriorative to the overall experience of other mobile network users (in cases when videos are viewed or files are shared), or even competitive with the services provided by the mobile service operators (in case of VoIP applications). Therefore, MNOs can feel tempted to manipulate or even restrict certain types of data flows.

Algorithm 1: measurement-cycle(server, proto) runs a measurement cycle using a specific protocol description (this pseudo code only shows the download direction)

```
con ← connect (server, rfport (proto));
send-random-signature (con, proto);
rf-goodput ← measure-goodput (con);
close (con);
con ← connect (server, pfport (proto));
send-protocol-signature (con, proto);
pf-goodput ← measure-goodput (con);
close (con);
```

C. Measurement test lifecycle

As mentioned before, to detect the traffic differentiation, we need to emulate a pair of flows that are identical except in one respect, in order to trigger traffic differentiation along the path. By measuring and comparing performance of these two flows, we can make a conclusion about the presence of a content-driven traffic manipulation from the ISP's side. So the core task during the measurement test is to determine the application goodput in both, upload and download, directions for "protocol" and "random" flows.

To obtain this goodput estimation, BonaFide follows the steps shown in Algorithm 1. In order to get a baseline goodput estimation, BonaFide first obtains measurements for the random flow and after completing this, proceeds to the protocol flow. When measuring performance for the protocol flow, the protocol signatures are exchanged as shown in Figure 2. Following this, the goodput calculation is performed using the function measure-goodput(), shown in Algorithm 2. A random message payload, of appropriate size (e.g. 2kB, 4kB, etc.), is created and a starting time stamp is recorded before sending this data to the other end as a TCP bulk message. Once an "OK" notification is received back from the BonaFide server, the end time is recorded (see Figure 3) and these values are used to calculate the goodput for this bulk message size. Once goodput values for all possible message sizes are obtained or a maximum round trip time is reached, the results are recorded.

The size of bulk messages used during the goodput measurements have an impact upon the results obtained. Prokkola et. al. [16] showed that the near-nominal application throughput (and hence the goodput) in mobile networks is realizable for large payloads only. So, the size of the message used for probing the network (i.e. the bulk message in case of BonaFide) should be large enough to fully utilize the allocated bandwidth. This value is different for various mobile networks, and it depends on the network type and configuration. In order to be able to measure the goodput on different mobile networks, we send a train of bulk messages. We start with a relatively small 2kB message and gradually increase the bulk message size. if necessary. At each step the bulk message round-trip-time (RTT) value is measured.

It was determined via empirical testing that a 2 second RTT value is enough to fully utilize the allocated network bandwidth regardless the mobile network type. Increasing the bulk message size after this value does not make sense anymore, since the estimated goodput value remains the same. While the maximum bulk message size is set to 8MB, in

Algorithm 2: measure-goodput(con) obtains goodput measurements on the connection identified by the parameter

```
i ← 0;
rtt ← 0;
msgsize ← 1kB;
maxrtt ← 2 sec;
while rtt < maxrtt ∧ msgsize < 8 MB do
    i ← i + 1;
    msgsize ← msgsize · 2;
    msg ← random (msgsize);
    start ← time ();
    send (con, msg);
    recv (con, "OK");
    end ← time ();
    rtt ← (end − start);
    goodput [i ] ← (msgsize/rtt);
end
return goodput;
```

practice, the goodput estimation round usually terminates when the bulk message size reaches the 1MB or 2MB value (in case of HSPA mobile networks). The bulk message size of 4MB is used in rare cases when the mobile network bandwidth value is higher than 5 Mbps. The 8MB message size is never used in practice (modern HSPA mobile networks have a limit up to 7.2 Mbps) and it is reserved for future use.

When a user wants to start a new test, a TCP control connection is established with the BonaFide server. The BonaFide client then sends the start_new_test <protocol name> <number of cycles> command (see Fig. 1). Once the server receives this command it creates an object to store measurement results, and generates the UUID for the measurement test session. This UUID value later is used to identify the test instance on a server side while handing client's requests. For every application protocol that is supported, a random and protocol flow connection is also created. A single measurement test execution usually is not enough to make an unambiguous conclusion about the presence of traffic shaping along the path. So it is highly recommended to repeat the same test several times, <number of cycles>, to improve the quality of measurements. Each set of goodput measurements for a single flow in both directions is called a measurement cycle. We inject the protocol messages at the beginning of a new TCP connection, since some deep packet inspection tools (e.g. L7-filter [23]) make a decision about the type of the flow by looking at the first several packets only. In the case when we run a random flow, the server and the client also inject protocol messages, but in this case, the protocol message content is generated randomly, while the size of the messages defined by the protocol description file is preserved.

D. Results retrieval and sharing

After the entire measurement test has been completed, the upload goodput results are stored on the client side, and the measured download goodput results – on the server side. To retrieve the measurement results from the server, the client sends a retrieve_test_results command to the main endpoint (see Fig. 1).

By default, BonaFide server does not collect any information about measurement tests. All measurement tests are stored

Table I. THROTTLING POLICY FOR BITTORRENT FLOWS DEPLOYED ON THE ROUTER IN EXPERIMENTAL ENVIRONMENT

Destination Port Number	Maximum allowed bandwidth
45000	no limit
45001	1024 kbps
45002	512 kbps
45003	256 kbps
45004	128 kbps
45005	64 kbps
45006	drop packets (100% drop rate)
45007	drop packets (50% drop rate)

locally by the BonaFide client. However, we consider the tool to be of most value if there is a possibility for distributed collection of user measurement results. For that reason, we have set up an infrastructure for collecting and analyzing measurement results, and each BonaFide user has an option of submitting his/her measurement results from any of the measurements (Fig. 1, `upload_measurement_results`). In fact, we encourage any reader to try out the tool and share their results with us. The measurement test results are stored in an HTML file. They reflect the measured link goodput values in the upload and download direction for the random and protocol flows. HTML format was chosen to store the measurement test results since it can be viewed by any browser installed on an Android-based smartphone.

E. Data analysis

ISPs can shape the application protocol performance not only by limiting the application goodput. They could also terminate TCP connections by sending TCP FIN or TCP RST packets to the client (or server), or drop the packets with a 100% drop rate. The traffic shaping detection tool is able to detect these types of traffic differentiations. In the first scenario, the detection tool would notice that the socket was unexpectedly closed during the test, and it would be shown in the output table as a `connection_reset` record in a corresponding measurement cycle row. In the second case, the traffic detection tool would receive the `SocketTimeoutException` during the test execution. This would be displayed in the output table as a `timeout` record. Thus, if the output tables for protocol flows have many `connection_reset` or/and `timeout` records and the corresponding tables for random flows do not have them (or have only few of them), then it is fairly likely that the provider does traffic shaping for an application protocol that we tested.

BonaFide automatically makes a decision about the presence of traffic shaping along the path using statistical methods. A measurement test may therefore return any of the following five decisions: *traffic shaping is observed*, *traffic shaping most probably exists*, *traffic shaping most probably does not exist*, *traffic shaping is not observed*, and *measured data is not reliable*.

This decision is a two-step process. At first, the measurement results are analyzed by the completeness criterion. A `fail_ratio` which determines the ratio of measurement cycles that exited with a `connection_reset` or a `timeout` exception to the total number of cycles is calculated. This ratio is calculated for both protocol and random flows. If the `fail_ratio` for a random flow is less than 20% and more than 70% for a protocol flow. BonaFide concludes that

Table III. LIST OF MOBILE OPERATORS USED IN OUR TRAFFIC SHAPING DETECTION EXPERIMENTS ON MOBILE NETWORKS.

MVNO	MNO
ALDI Talk (MEDIONMobile)	*E-Plus*
NettoKOM	*E-Plus*
O_2	O_2
Congstar	*Telekom*

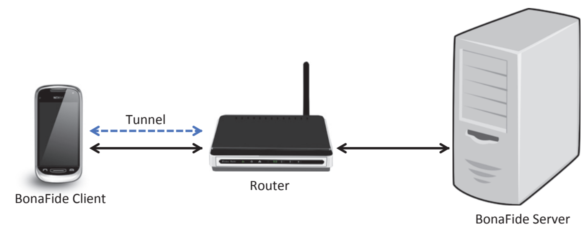

Figure 4. Experimental Setup Overview

the application protocol was affected by a traffic shaping mechanism along the path. Otherwise, BonaFide proceeds to the second step. The second step is a combination of the *Mann-Whitney U* significance test and the confidence interval defined using *Student's t* distribution coefficient. According to the Mann-Whitney U test goodput values for the two flows are combined into a single set, sorted in an ascending order, and assigned a rank corresponding to its position in a sorted set. Next, the sum of ranks for each flow type is calculated and the greater of two sums is assigned to T_x. The *Mann–Whitney U* value is therefore calculated using the following formula:

$$U = n_1 \cdot n_2 + \frac{n_x \cdot (n_x + 1)}{2} - T_x; \tag{1}$$

where the n_1, n_2 are numbers of measured goodput values for protocol flow and random flow respectively, and the n_x is a number of measured goodput values in a flow with a higher rank T_x. Finally the calculated U value is compared with a corresponding $U_{critical}$ value from the predefined *Mann-Whitney* critical values table (with a level of confidence 95%) [24]. The case when $U > U_{critical}$ means that there is no traffic shaping along the path.

The $U \leq U_{critical}$ does not mean that the tested application protocol has been shaped. For example, the *Mann-Whitney U* test would return the false positive decision about the presence of traffic shaping on the following sets of goodput values:

```
Protocol flow: 1000, 1001, 1002, 1003, 1004 (kbps)
Random flow: 1005, 1006, 1007, 1008, 1009 (kbps)
```

The maximum goodput value for the protocol flow is less than minimum goodput value for the random flow (i.e., $U = 0$). However, all goodput values stay at the same level, and the traffic shaping methods definitely were not applied in this case. Hence, it is necessary to take the relative distance of the mean values into account while making a decision.

IV. TESTING AND VERIFICATION

Before running experiments on mobile networks, it is important to verify that the tool is able to effectively identify traffic shaping for the protocols that it supports. As such, we ran a series of measurement tests on an experimental

Table II. THE RESULTS OF MEASUREMENTS MADE ON THE (A) ALDITALK AND (B) NETTOKOM EDGE NETWORKS DURING NIGHT-TIME (1AM). THE UPLOAD AND DOWNLOAD DATA-RATES ARE IN KBPS.

Protocol	Cycle	Download		Upload	
		RF	PF	RF	PF
BitTorrent	1	135	120	93	90
	2	135	129	**97**	**93**
	3	**136**	**134**	91	89
	4	122	123	93	91
	5	120	121	93	92
SIP	1	12	133	5	90
	2	130	122	94	94
	3	**135**	133	**97**	**95**
	4	133	134	92	94
	5	120	**135**	94	93

(a)

Protocol	Cycle	Download		Upload	
		RF	PF	RF	PF
BitTorrent	1	**187**	**190**	41	**46**
	2	186	163	**46**	**46**
	3	147	148	**46**	42
	4	180	189	**46**	42
	5	110	186	**46**	**46**
SIP	1	164	11	16	32
	2	**187**	170	**46**	44
	3	185	**183**	44	16
	4	170	180	**46**	**46**
	5	182	**183**	**46**	**46**

(b)

infrastructure shown in Figure 4. In this experimental setup all generated traffic goes through a Linux machine, which plays the role of a router. The router is configured to perform deep packet inspection and throttling flows' performance according to predefined set of rules. Since we know how the performance of flows that carry different types of data should look like, we can analyze results obtained during the measurement tests, and verify whether traffic shaping detection actually works properly or not.

To apply traffic shaping rules on the router the L7-filter [23] [25] was used. L7-filter is a software package providing a classifier for Linux's Netfilter subsystem, which can categorize flows based on application layer data. Unlike most other classifiers, it does not just look at simple values such as port numbers, but instead, it does regular expression matching on the application layer data to determine what protocols are being used. L7-filter can be used when protocols using unpredictable ports (e.g., peer-to peer file sharing) or non-standard ports (e.g., web traffic on port 1111) are used. It can also be used to distinguish between protocols which share a port (e.g., peer-to-peer file sharing that uses port 80). A benefit of using the L7-filter is that it provides a large number of well tested regular expressions for different protocols, including those that might be interesting from the smartphone users perspective, like Flash Video, SIP, RTSP, H323, different P2P protocols and HTTP.

In order to test whether traffic shaping detection works for different traffic shaping configurations, a number of traffic differentiation rules for the BitTorrent protocol were deployed on the router (see Table I). The router was configured to limit BitTorrent performance on port numbers 45000-45005 and to drop BitTorrent packets on ports 45006 and 45007 in both (upload and download) directions. The performance of flows that carry other protocol data is not affected and stays the same on all destination port numbers.

When BonaFide was used within this test environment to detect traffic shaping, it was successfully able to identify instances of traffic shaping that matched the rules setup on the router.

Furthermore, we ran tests to verify that BonaFide does not consume large amounts of data on mobile networks. Also, in order to compare the amount of data consumed by BonaFide to Glasnost, so as to establish BonaFide's suitability for mobile networks, Glasnost was used on a laptop while using mobile phones as modems. The data consumption values for both, Glasnost and Bonafide, are presented in Table IV. From this table it becomes clear that not only does Bonafide consume relatively small amounts of data on mobile networks, but also that Glasnost is unsuitable for mobile networks since it consumes large quantities of data that can quickly exhaust quotas on mobile networks.

V. MOBILE NETWORK TESTING

A series of traffic shaping detection tests on mobile networks were executed in order to learn how and to what extent content-based traffic differentiation policies are deployed on mobile networks. However, mobile service is not only provided to users via MNOs, but even some providers that do not own radio spectrum or wireless network infrastructure. Such mobile virtual network operators (MVNOs) obtain access to network services at wholesale rates and then resell this independently. Since such MVNOs are quite popular amongst budget conscious users, testing whether there was traffic-differentiation applied between the MVNO and its related operator is also interesting. Furthermore, since multiple MVNOs might use the same basic access infrastructure, it is also interesting to see whether a MNO applies traffic differentiation between the MVNOs using its infrastructure in order to provide better service to a selected few. To obtain a mixture of MNOs and MVNOs, the mobile operators listed in Table III were chosen to perform multiple tests upon. The tests presented here were all performed on the HSPA and EDGE networks of the selected operators.

During our experiments on the chosen operators, there were some instances of traffic shaping that were observed. On the Congstar network, when a test for SIP traffic shaping was run during the evening (between 7-9 PM), it was discovered that this MVNO does indeed shape SIP traffic on their network. Furthermore, while the general maximum for HSPA networks is observed to be around 7.2 Mbps, the Congstar network appears to top out at about 360 kbps. This is quite low and indicative of traffic-shaping applied by the network operator. However, it was more interesting to note that when this test was repeated during morning hours (9 AM), the traffic shaping was no longer present. As such, Congstar would appear to be an example of a mobile network that shapes traffic based on time of day and protocol.

Furthermore, measurements carried out on the service provided by ALDITalk and NettoKOM were also quite interesting. Normally, due to the reduced load, EDGE networks are more stable during night-time and measurement results for this time, for both operators, can be seen in Table II. The

Table IV. A COMPARISON OF AVERAGE DATA CONSUMPTION BETWEEN GLASNOST AND BONAFIDE (RESULTS IN MB).

	Flash Video	HTTP	BitTorrent
Glasnost	4911.80	72.70	1741.40
BonaFide (EDGE)	1.46	1.37	1.47
BonaFide (HSPA)	17.22	17.84	18.21

goodput achieved during download and upload tests on both these operators is quite different. Considering that both these operators are MVNOs operating on the E-Plus network, this result makes it clear that even though the access infrastructure for them is the same, different internal configuration is used. It also appears as though NettoKOM applies protocol-blind rate limiting to upload traffic.

VI. CONCLUSION

We demonstrated that certain traffic manipulations exist both on the EDGE and 3G networks. Our evaluation of BonaFide was carried out in a rather geospatially constrained environment, as most of the measurements were done in the city of Bremen (Germany) and its vicinities. These measurements were done at different times of day, in order to establish a potential dependency of mobile traffic shaping to the network usage peak hours. However we are curious to find out what happens on a country or even a global scale. As one can imagine carrying out such measurement tests by a single individual is not a scalable or even a feasible task. Therefore, we made BonaFide available on Google Play market, such that users around the world can install it, test it and share their measurement results with us. Our ultimate goal is to establish a common quality index for mobile experience around the world. The index to reflect the variation between operators' service promises, and the actual satisfaction on behalf of the individual end customers.

ACKNOWLEDGEMENT

This work was partly funded by Flamingo, a Network of Excellence project (ICT-318488) supported by the European Commission under its Seventh Framework Programme.

REFERENCES

[1] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: Enabling End Users to Detect Traffic Differentiation," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2010.

[2] 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.

[3] M. B. Tariq, M. Motiwala, N. Feamster, and M. Ammar, "Detecting network neutrality violations with causal inference," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 289–300.

[4] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting traffic differentiation in backbone ISPs with NetPolice," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 103–115.

[5] P. Kanuparthy and C. Dovrolis, "ShaperProbe: end-to-end detection of ISP traffic shaping using active methods," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 473–482.

[6] ——, "DiffProbe: Detecting ISP Service Discrimination." in *INFOCOM*. IEEE, 2010, pp. 1649–1657.

[7] StatCounter. (2012, August) Mobile vs. Desktop from Jan 2009 to Aug 2012. StatCounter Web Analytics. Dublin, Ireland.

[8] M. Meeker, "Internet trends," in *D10 Conference*, Rancho Palos Verdes, California, May 2012.

[9] C. Dovrolis, P. K. Gummadi, A. Kuzmanovic, and S. D. Meinrath, "Measurement lab: overview and an invitation to the research community," *Computer Communication Review*, vol. 40, no. 3, pp. 53–56, 2010.

[10] "SamKnows - Accurate broadband performance information for consumers, governments and ISPs," available at http://www.samknows.com/broadband/index.php.

[11] "RIPE Atlas Project," available at http://atlas.ripe.net/.

[12] "Network Diagnostic Tool (NDT)," available at http://www.internet2.edu/performance/ndt/.

[13] "Fing - the Internet measurement tool," available at http://www.over-look.com/site/.

[14] "MobiPerf Official Website," available at http://mobiperf.com.

[15] "Google Play: Traffic Monitor application," available at https://play.google.com/store/apps/details?id=com.radioopt.widget.

[16] J. Prokkola, P. Perala, M. Hanski, and E. Piri, "3G/HSPA Performance in Live Networks from the End User Perspective," in *Communications, 2009. ICC '09. IEEE International Conference on*, June 2009, pp. 1–6.

[17] K. Pentikousis, M. Palola, M. Jurvansuu, and P. Perala, "Active goodput measurements from a public 3G/UMTS network," *Communications Letters, IEEE*, vol. 9, no. 9, pp. 802 – 804, sep 2005.

[18] P. Benko, G. Malicsko, and A. Veres, "A large-scale, passive analysis of end-to-end TCP performance over GPRS," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 3, march 2004, pp. 1882 –1892 vol.3.

[19] P. Calyam, W. M, M. Sridharan, A. Khan, and P. Schopis, "H.323 Beacon: An H.323 application related end-to-end performance troubleshooting tool," in *ACM SIGCOMM NetTs*, 2004.

[20] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, *SIP: Session Initiation Protocol. RFC 2543*, March 1999.

[21] H. Schulzrinne, A. Rao, and R. Lanphier, *Real Time Streaming Protocol (RTSP). RFC 2326*, April 1998.

[22] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011-2016," available at http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html.

[23] "Application layer packet classifier for Linux (L7-filter)," available at http://l7-filter.clearfoundation.com/.

[24] "Critical Values for the Mann-Whitney U-Test," available at http://www.saburchill.com/IBbiology/downloads/002.pdf.

[25] D. Guo, L. N. Bhuyan, and B. Liu, "An Efficient Parallelized L7-Filter Design for Multicore Servers," *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, p. 1, 2011.