A Framework for Monitoring Net Neutrality

Wilfried Mayer
SBA Research
wmayer@sba-research.org

Thomas Schreiber
TU Wien
thomas.schreiber@gmx.at

Edgar Weippl
SBA Research
eweippl@sba-research.org

ABSTRACT

Internet service providers can discriminate traffic in certain ways, e.g., by the used protocol or application. This leads to advantages for providers, but also harms the freedom and innovation in the Internet. However, ISPs currently use a variety of technical measures. Some can be seen as questionable regarding net neutrality – an important topic in legal and economic discussions. These methods are often neither technically identified nor continuously monitored, which prevents an informed discussion about the legitimacy of such methods. In this paper, we design and implement an open-source framework for monitoring such techniques within a country. Compared to other projects, we are able to fully control the client and server endpoint and therefore analyze network behavior in depth. We implement 17 different metrics that cover a wide range of the network spectrum. We test basic network features on transport layer as well as specific application layer protocols. We then use this framework to monitor five different Internet products in Austria over the time span of more than one year. We evaluate the results from our three measurement periods of three months each and find different questionable methods in place. This includes e.g., middleboxes used for various protocols, monetization of DNS results and different behavior for special DNS queries. However, many metrics show that currently no questionable techniques are used.

ACM Reference format:
Wilfried Mayer, Thomas Schreiber, and Edgar Weippl. 2018. A Framework for Monitoring Net Neutrality. In *Proceedings of International Conference on Availability, Reliability and Security, Hamburg, Germany, August 27–30, 2018 (ARES 2018),* 10 pages.
DOI: 10.1145/3230833.3230846

1 INTRODUCTION

Net Neutrality is a highly discussed topic from a technological, economical and legal viewpoint [3, 7, 11, 13]. In short, it describes that Internet Service Providers (ISP) should not treat traffic differently based on any characteristic [2], e.g., the protocol used. While legislation, e.g., in the EU or the USA, as well as economic discussions often gain attention, technological discussions of used techniques are scarce. Our study focuses more on these technical aspects. Therefore, we create a specialized measurement platform that is capable of detecting the usage and the consequences of miscellaneous network management techniques. We implement a variety of test cases and conduct three measurement periods of three months each. We then analyze the measurement results and find questionable methods in use. In this paper, we solely focus on the technical part and do not discuss the findings from a legal or economic viewpoint.

Techniques used to differentiate traffic and potential net neutrality violations are often hard to detect. Customers sometimes have knowledge of different infringements, but they just communicate it over bulletin boards or directly to network regulators. However, evidence from an integrated and consistent monitoring system is missing. With our work, we introduce a framework that consistently monitors different Internet products with specialized metrics in a transparent way. That is not an easy task, since questionable techniques do not necessarily differentiate from network management methods or security measures. Port blocking techniques are also used in simple network firewalls, changed DNS responses are used in censorship systems and proxying of traffic is often enforced by company policies, but all three techniques are questionable in terms of net neutrality.

Because the topic of net neutrality is rather young, measurement studies and frameworks focusing on it are scarce. Related research in the field of Internet measurements exist, e.g., the RIPE Atlas network. In addition, studies from the field of censorship detection as well as methods from intrusion detection systems are relevant. Still, not all these approaches are suitable for our case. We introduce a methodology that enables us to fully control both endpoints and conduct extensive tests in terms of computation and network bandwidth. This system is scalable in terms of metrics used and in terms of the Internet products checked. One main challenge for a monitoring system is the need for reproducibility and traceability. Simple reports of found techniques have to include a traceback to the original traffic. To facilitate this process, we capture the raw traffic on server- and client-side and link it directly to the generated results. Therefore, each reoccurring test – an instance of a specific metric – has a generated result (in JSON format) and a complete record of the raw network traffic dump (as PCAP file).

We then use this framework to conduct measurements for five different Internet service providers in Austria. We repeat this three-month monitoring process in total three times to validate our results over time. We identify some questionable techniques that are further described in the paper.

The main contributions of this paper are:

- An open-source framework for net neutrality measurements.
- A wide spectrum of defined metrics for different aspects of net transparency.
- Results of three measurement periods (each three months) in Austria, revealing questionable techniques of Internet service providers and their changes made over time.

The remainder of this paper is structured as follows. Section 2 gives an overview of the technologies relevant to our work and the related scientific work. Afterwards, Section 3 describes the measurement framework, the defined metrics and how the measurement was conducted. An evaluation of the results follows in Section 4. We discuss these results in Section 5 before we conclude in Section 6.

2 RELATED WORK

Studies were also conducted in the field of censorship detection. Anderson et al. [1] made use of the RIPE Atlas network, and measured blocking events in Turkey and Russia. Although the RIPE Atlas framework is suited for large-scale network measurements, the different testing methods are rather limited to non-computational tasks, e.g., pings or traceroutes. Filastò and Applebaum [5] created the Open Observatory for Network Interference (OONI), a censorship analysis tool. It is a framework that measures data tampering for various protocols, e.g., HTTP and DNS, while using a Tor-secured connection as a reference. The project uses open methodologies in a free and open source software environment. OONI does not support further network quality measurements, so it is not a solution for our use case. Since some websites in Austria are blocked by ISPs and because of the open nature of the project, we adopted some OONI test decks in our framework. Content manipulation and blocking was also studied by Khattak et al. [10]. They studied differences of HTTP content served over Tor, compared to non-Tor users. They found that around 6.8% of HTTP requests are blocked when using Tor while being served normally without using a Tor connection. They only measured the homepage of each respective site, therefore not taking content manipulation or blocking into account that occurs on subpages. Dischinger et al. [4] also introduced a measurement tool: "Glasnost", which is capable of detecting traffic differentiation for end users. With 350,000 users testing over 5,800 ISPs, they found that P2P protocols, e.g., BitTorrent, are treated differently. This traffic shaping is only affecting a limited number of users and some ISPs only employ these measures at certain times of a day. As a basis for discrimination, most providers inspect the TCP layers and slow down traffic that flows between ports that are associated with certain protocols, e.g., TCP port 6881 for BitTorrent traffic. Based on their results, this paper also measures traffic shaping in regular intervals. Flach et al. [6] surveyed Internet-wide traffic policing. In comparison to shaping were data is buffered, traffic policing drops data above a certain traffic rate. They collected traces from Google servers and found out that depending on regions 7% of the connections were affected. In comparison, our study concentrates on a wider spectrum of metrics.

Various cases of net neutrality violations are mentioned in media or on various blogs. For example, the case of ISPs that remove their customers' email encryption [8]. Although this are only single reports and not scientific studies, these reports give good insights in the practices of Internet service providers. We try to implement automated tests and integrate these metrics in our framework.

Kakhi et al. [9] focused their research on techniques that were used in T-Mobile's BingeOn program. They discovered that T-Mobile uses policing to reduce bandwidth. Even though they claim to "optimize" video, no transcoding is taking place. Therefore, content is not modified. For detecting eligible video streams, they are using deep packet inspection, as the selection criteria are matched with a simple string comparison. They showed that by randomizing the request headers evading detection is possible. Although their study gives very good insights into the practices of a large ISP, it is limited to only this one ISP. In comparison, our framework is not limited, and our study includes results of five different Internet products. Nakibly et al. [12] show that some non-edge network operators are manipulating HTTP traffic. These operators use out-of-band packet injection, i.e., forged packets are inserted into the network stream without touching the original packet. Traffic from three universities and one company over multiple weeks was used, detecting that Asian network operators often inject advertisements. In contrast, we are focusing on edge providers' in-band HTTP manipulation in this work. Since we use a client/server architecture – with both endpoints located in Austria serving non-public content – we do not expect such alterations. It is relevant to our work, as it shows that not all content manipulation is caused directly by the ISP, but could also be attributed to core network operators. Xu et al. [16] also found that some cellular network operators manipulate HTTP traffic and include ad content for monetization. They also cache objects for performance increases. The same goal – generating revenue streams by delivering advertisements – can also be reached by other measures. In their research based on 259,000 measurements by 193,000 users, Weaver et al. [15] showed that some providers use failed DNS lookups for monetization. They showed that some ISPs don't return the legitimate NXDOMAIN error code for non-existing web pages but instead return a custom page to users. Aside from ISPs, third-party DNS providers use DNS redirection to custom error pages as their primary source of income. As a consequence, we implement a DNS lookup metric, where a request for non-existing hostnames is performed.

3 METHODOLOGY

We decided to create a client/server-based framework to conduct our measurements. The architecture, as shown in Figure 1, utilizes one highly capable server and a customizable number of configured clients. These clients are connected to different Internet products of different Internet Service Providers. Although the architecture of this framework seems rather simple, other approaches were just not suitable, as we want to test technical measures within one country. While crowd-sourced and highly distributed approaches like RIPE Atlas[1] or software measurement tools have their clear advantages, our approach enables a fully controlled, fully configurable environment for both endpoints. This allows us to conduct more complicated tests in regards to bandwidth usage and processing complexity. We conduct all measurements on an hourly basis. That enables measuring changes happening during a day, e.g., different treatment of data at peak times or capturing the exact moment of changes in the use of different techniques.

To fulfill these requirements, we decided to develop our own measurement framework. Therefore, clients request the planned tests via a REST API from the server. The tests are specific to different, exchangeable and extensible metrics, which are described in Section 3.2. The server schedules the tests of the different metrics in a non-overlapping way and returns the upcoming test configurations as JSON objects. Then, the test clients start the measurement at the given time interval. In this time interval, the needed functionality is loaded, the firewall permits only access for the needed ports and the actual test is performed. The results of each test are summarized to a single JSON object (which contains basic information about the test and additional metric-specific results), transferred to the server and stored in a MongoDB database for later analysis. To keep the results traceable, a packet capture (PCAP file) of every test is recorded on the clients and the server.

The measurement framework was realized with Python 3, MongoDB, dumpcap and Bottle. Several other dependencies (e.g., Scapy, dnspython) are required for some metrics. The source code is openly available[2].

3.1 Experimental Deployment

We decided to support five different connections of major Austrian Internet service providers to test current technical measures occurring in Austria. We included ADSL, cable and three stationary LTE products into our study. Each connection used its own client hardware and modem. Therefore, we deployed our measurement framework with five different clients. For the client hardware, we used Intel NUCs (2x1.8GHz CPU, 16GB RAM). In comparison of test clients of other measurement architectures that use single-board computers, these rather strong clients did not limit computational intense tests. We were also able to rule out any result noise, caused by undersized hardware (happened, e.g., with the first generation of RIPE Atlas probes). The server was configured with a static IPv4 address and connected via Gigabit LAN from the university network of TU Wien. As modems we mostly used operator provided modems (Huawei E5180 and E5170, TG588v, CNB CH7465CE) and one other device (TP-Link TL-MR6400). All contracts were concluded by private customers, to preclude any privilege based on company-based contracts and reproduce the experience of private customers.

3.2 Metrics

In this setup, we defined 17 different metrics. Each of these metrics is designed to measure a single aspect of possible technical measures. We tried to implement a broad spectrum of tests in different ISO/OSI layers, which is denoted in the number after the abbreviated name. Because our framework is extensible, new metrics can be easily adopted.

3.2.1 Voice over IP. Since VoIP could be seen harmful to the traditional business models of mobile ISPs, we measure the quality of VoIP streams. Blocking of VoIP traffic should be detected by other metrics, so we additionally measure different QoS metrics. For this, we simulate a call using a replay of a pre-recorded RTP stream given in a PCAP file. The individual packets are loaded, parsed and filtered using scapy's `scapy.all.readpcap` functionality. The payload itself is sent using the standard Python UDP socket implementation. The recorded traffic is finally analyzed with tshark to generate QoS metrics, e.g., jitter, packet loss and delta metrics provided by tshark. (Metric: Voice over IP test – VOIP7)

3.2.2 SYN flooding attack test. This metric tests if the ISP is blocking or altering SYN flooding attacks with a specific amount of SYN packets sent. Normally, this kind of attack can lead to DoS. To test it, the number of client-side transmitted and server-side received TCP segments with the SYN flag set is measured and compared to see any inconsistency. This happens on predefined TCP ports. (Metric: SYN Flooding attack test – SYN4)

3.2.3 E-mail related metrics. We implement two metrics to detect the use of middleboxes in e-mail related protocols. This is done by transmitting commands containing syntactical errors. For POP3 we use malformed e-mail addresses making the authentication process invalid. We suspect middleboxes to show an irregular behavior after receiving these syntax errors. This test is then executed on the standard POP3 port TCP/110 and one control port TCP/8110. (Metric: Invalid POP3 syntax test – POP37) For SMTP, we also use malformed SMTP commands and check the integrity of the received data of server and client. (Metric: Invalid SMTP syntax text – SMTP7) A sequence diagram of these metrics is given in Figure 2. SMTP, an unauthenticated, unencrypted protocol, is upgraded to a TLS-secured connection with the

[1] https://atlas.ripe.net/
[2] https://github.com/sbaresearch/monitoring-net-neutrality

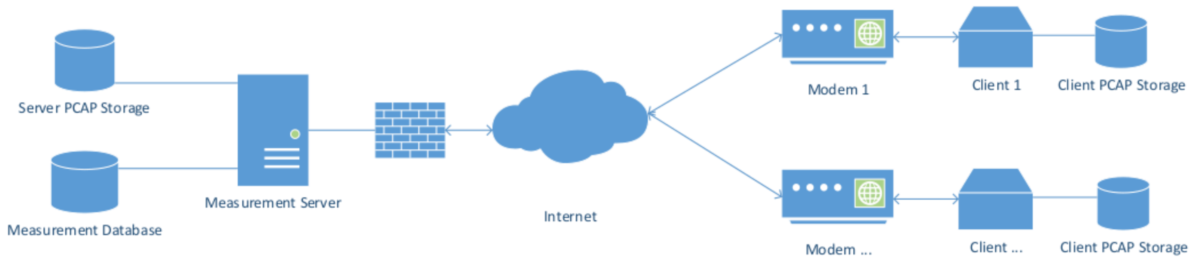


Figure 1: Basic measurement framework architecture.

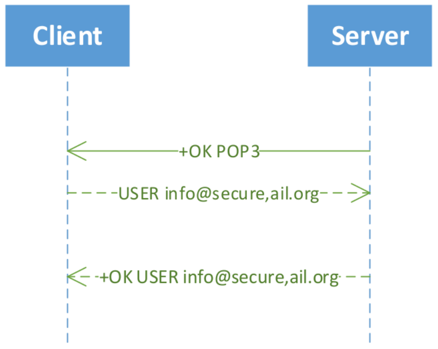


Figure 2: Sequence diagrams of the POP37 metrics.

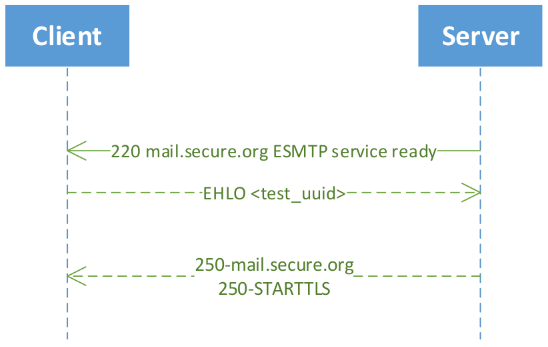


Figure 3: Sequence diagrams of the SMTP7 metrics.

STARTTLS command. In the past, some ISPs have been known for not transmitting this command to the client. After establishing a SMTP connection, one metric tests if the STARTTLS commands are transmitted and received at the client without interference. A sequence diagram of these metrics is given in Figure 3. (Metric: StartTLS stripping test – STLS7)

3.2.4 HTTP manipulation tests. With the first test (Metric: HTTP Caching and Manipulation – CM7), which is based on Nakibly et al. [12] resp. Xu et al. [16], we want to find out, if network operators manipulate content for monetization or cache objects for performance reasons. It works by simulating HTTP traffic and measures two possible techniques:

- Caching, by sending the same HTTP request multiple times and returning the same response with large-sized image data and HTTP headers set for caching (`Cache-Control: max-age=600, public`). In theory, the ISP could cache this resource and deliver it directly to customers.
- Manipulations, by sending HTTP traffic and testing for changed parts of the transmitted traffic. Hence, we calculate checksums of the header and the body part of request resp. response.

We then introduce an additional metric to test modifications with traffic that contains virus-like data. Therefore, we transfer the EICAR test file (A file recognized as a virus by anti-virus software[3]), and measure if any change has happened. ISPs could argue that this is necessary from a security viewpoint, but it clearly indicates content inspection. (Metric: HTTP Antivirus test – VS7). Lastly, another metric (Metric: Invalid HTTP syntax test – HTTP7) is testing if any modification occurs when invalid HTTP syntax (e.g., incorrect request codes) is used. This is similar to the e-mail-related metrics and can lead to middlebox detection.

3.2.5 Basic network measurements. For basic network measurements like port blocking and different treatment of packets, we try to connect to various ports and measure the network connection parameters. We do this for a variety of common and uncommon TCP ports (20 FTP, 80 HTTP, 443 HTTPS, 554 RTSP, 1214 P2P Kazaa, 1725 Valve Steam Client for Online Gaming, 6881 BitTorrent, ...) and UDP ports (e.g., 1725, 5060, 6881, ...). The test follows a simple request ("ping message") and response ("pong message") approach. We then measure multiple connection characteristics:

- If no segments are received, the provider seems to block that port.

[3]https://www.eicar.org/86-0-Intended-use.html

- The round trip time (RTT) from client side and server side, to check traffic differentiation.
- The TTL of all packets received, which detects middleboxes and different routing strategies for different traffic.

This is all implemented in two metrics. (Metric: Basic TCP measurements – TCP4 and Basic UDP measurements – UDP4) We additionally execute *traceroute* commands and check the results for any inconsistency. This helps us to draw a detailed picture of the routing strategies of the different providers (Metric: Traceroute test – TRAC3).

3.2.6 TLS. We also check for modifications in TLS handshakes by setting invalid TLS length fields in the header. Client and server are ignoring the invalid fields, middleboxes may throw an error or do not allow to establish connections at all. We then compare, if the checksums of the TLS handshakes match. (Metric: Malformed TLS handshake test – TLS4)

3.2.7 Bandwidth tests. To measure traffic shaping based on specific ports, we conduct a speed test measurement based on the RTR Multithreaded Broadband Test [14], which was developed by the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR) and implemented in their `netztest.at` website to check connection quality. For our framework, we implement a stripped-down version of the RMBT test in Python. The communication sequence is identical to other RMBT-based tests and the data for the transmission is randomly generated. It is divided into multiple phases. After the test initialization, a download resp. upload pre-test for getting a rough estimate of the connection speed is performed. It works by sending data chunks that double in size (first iteration: 4k) with every iteration in a fixed time interval (2 seconds). After that, the download resp. upload test phase is conducted. It sends data chunks of fixed size (we used 4096 bytes) in a fixed time interval and measures the number of packets received. With this metric, we are able to measure connection speed, and any differences over time. A sequence diagram of these metrics is given in Figure 4. (Metric: TCP bandwidth test – TCPS4).

We then tried to test for techniques that are similar to BingeOn used by T-Mobile USA [9], that uses the HTTP `Host` header to shape traffic. We test the speed similar to the former metric, with no upload test and the test configuration reduced in a single HTTP GET request that mimics traffic from a video provider. The server then directly responds with the download phase of the test. (Metric: Multimedia test – MM7)

3.2.8 DNS-based metrics. We defined two metrics for testing DNS-based techniques. First, we test which return code the default DNS servers of ISPs are returning for non-existing domains. We use one randomly (manually) chosen domain – that does not exist – and analyze the returned DNS result. For the domain `www.123hjaf9hu32iufhuihoafine.com` we would expect a "Not found" return code `NXDOMAIN` and nothing else. (Metric: Non-existing Hosts DNS test – NDNS7). Second, we looked at the results for in some way censored domains. One example is `www.kinox.to` or `www.thepiratebay.se`, domains that ISPs are forced to block by the Austrian Supreme Court (OGH) judgment 4 Ob 71/14s following judgment C-314/12 of the European Court of Justice (ECJ). We compared the results to Googles DNS resolver 8.8.8.8[4]. We are interested in how and if ISPs block these domains. (Metric: Blocked Hosts DNS test – BDNS7).

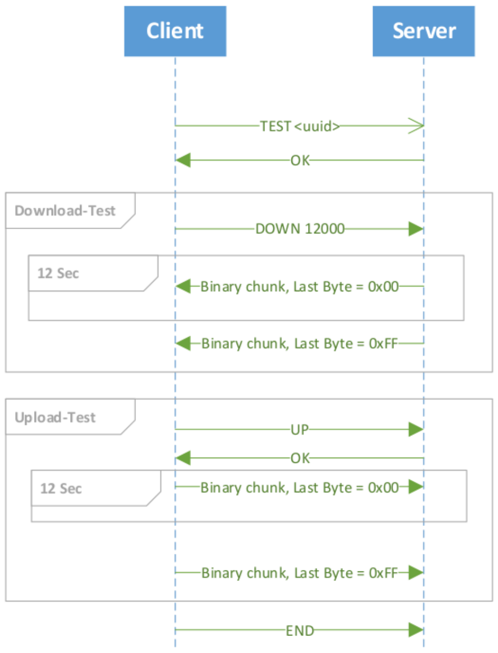

Client
Server
TEST <uuid>
OK
Download-Test
DOWN 12000
12 Sec
Binary chunk, Last Byte = 0x00
Binary chunk, Last Byte = 0xFF
Upload-Test
UP
OK
12 Sec
Binary chunk, Last Byte = 0x00
Binary chunk, Last Byte = 0xFF
END

Figure 4: Sequence diagram of the TCPS4 metric.

3.2.9 OONI. Although this metric tests similar modifications than other metrics, it utilizes the Tor OONI framework (Metric: Tor OONI test – OONI7). This censorship analysis tool detects differences between the direct transmission and a transmission done via the Tor low-latency anonymity network, that routes traffic over three relay nodes. The final exit relay then accesses the destination server. This framework is mainly used for censorship detection, but we use three different OONI testdecks: (i) manipulation/http_ invalid_ request_ line, (ii) manipulation/http_ header_ field_ manipulation, (iii) blocking/web_ connectivity) to verify the results of our other metrics (i) HTTP7, (ii) HTTP7, CM7 and VS7, (iii) BDNS7.

4 EVALUATION

In this section, we describe the results of our measurements. First the datasets generated, second special results where we found questionable techniques and third a detailed overview of the results from all 17 defined metrics. Last, we explain

[4]https://developers.google.com/speed/public-dns/

change over time, as the first and last measurement are more than one year apart.

4.1 Dataset

The dataset contains test results of three measurement periods over the time span of three months each.

(1) August 2016 to October 2016, with 140,000 measurement points
(2) February 2017 to May 2017, with 140,000 measurement points
(3) September 2017 to December 2017, with 87,000 measurement points

We conducted these tests for five Internet products from different Austrian Internet Service Providers. We tested one ADSL, one cable and three stationary LTE products. All test metrics were conducted on an hourly basis with a manually defined scheduling. There are short time frames were no measurements have taken place, due to different hardware failures, maintenance windows and software updates.

4.2 Summarized Results

Although we conducted tests for seventeen different metrics, we only found some questionable techniques. In this section we are going to describe the most interesting findings and summarize the results. We found following techniques:

4.2.1 HTTP middlebox. We found one Internet product (ISP4) that uses a middlebox for traffic over port 80. We found evidence in more than one test and more than one metric. We measure higher TTL values for exactly one port and do not see this behavior for other Internet products. This leads us to the conclusion that only traffic on port 80 is treated differently. This can be seen in Table 1. Also new HTTP headers are added. We see this in our content manipulation metric for the `Host` as well as the `Connection: Keep-Alive` header. We also see that HTTP syntax errors sometimes prevent transmission to the server and commands written in irregular upper and lowercase (e.g., `CONtenT-LeNgtH`) were exchanged with the correct version (e.g., `Content-Length`). Therefore we assume the complete header is exchanged. We illustrate this behavior in Figure 5. In the last measurement period, we were not able to detect any irregularity and no middlebox in use, so we assume the provider stopped this technique.

4.2.2 SMTP middlebox. We found that ISP3 uses a kind of middlebox for SMTP traffic. We observe that SMTP traffic is changed in transmission. Wrong server response codes (Code: 136 instead of 250) appear to cause the middlebox to terminate the connection. Therefore, the last (correct) command, `220 2.0.0 Ready to start TLS` is not transmitted. Instead, the message `421 syntax error (wait for server reponse)` is received at client side. Although a middlebox is in place, we have not found any evidence of STARTTLS-Stripping. The changed communication can be seen in Table 4. In the last measurement period we were not able to detect any SMTP middlebox, but SMTP traffic over TCP/25 was completely blocked.

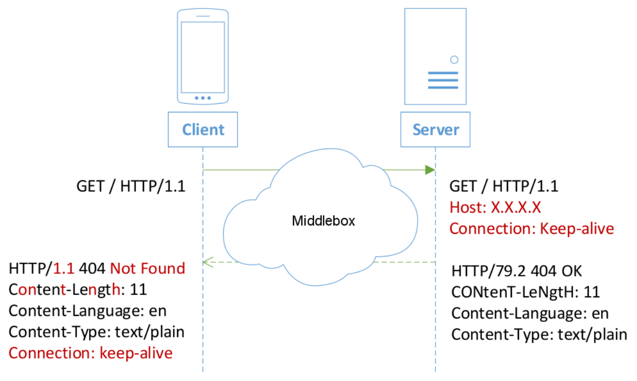
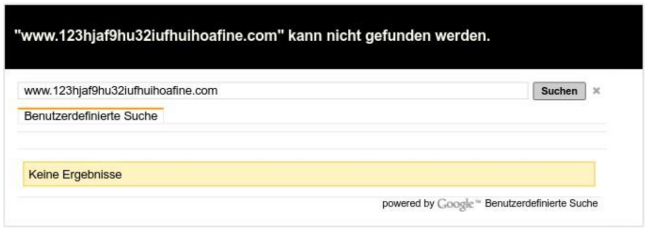


Figure 5: Detected Manipulation for Port 80.


"www.123hjaf9hu32iufhuihoafine.com" kann nicht gefunden werden.

www.123hjaf9hu32iufhuihoafine.com

Keine Ergebnisse

Figure 6: Custom search page for non-existing domains.

4.2.3 DNS blockades. We tested DNS blockades for various domains and found different behavior of the ISP's default DNS servers. We found that all providers blocked the domains www.kinox.to and www.movie4k.to. Four providers redirected to a webpage on a webserver in their address space showing a legal explanation of the domain blockade. One provider (ISP5) resolved the domains to `0.0.0.0`. For the domain www.thepiratebay.se we found that one default DNS server (ISP4) was not preventing access and was resolving the domain name correctly in the first two measurement periods. Table 2 gives an overview of the different results.

4.2.4 Google Custom Search for Not Existing Domain. For non-existing domains, ISP3 (resp. the default DNS server) redirects to an existing address in the address space of this provider. It shows a Google Custom Search Page, which is preselected with the requested domain. This custom search can be seen as a way to monetize requests as described by Weaver et al. [15]. We show a screenshot of the custom search page in Figure 6. One DNS server produced a timeout after five seconds, all other providers were returning the NXDOMAIN DNS status code correctly.

4.2.5 Blocked Port 5060. ISP1 blocks UDP port 5060 completely. After manual tests, we identified that this port – used for the Session Initiation Protocol for VoIP – is blocked directly on the provided router. With sufficient knowledge,

Port	80	554	220, 443, 1725, 5060, 6881, 8333, 48123
ISP1	51	51	51
ISP2	55	55	55
ISP3	52	52	52
ISP4	**61**	53	53
ISP5	55	**54**	55

Table 1: Median TTL Source Value for Different TCP Ports (measurement period: 2)

it is possible to deactivate the blockade easily in the router settings.

4.3 Detailed Results of Metrics

In this subsection, we present the detailed results of all metrics.

4.3.1 Basic TCP measurements (TCP4). First, Port 554 was occasionally differently treated for ISP5. We have no explanation for this behavior. Second, and more noticeable, the TTL on port 80 had a significant difference for ISP4. We also noticed that fixed line providers stayed consistent with their TTL values, whereas mobile providers varied to a greater degree. We could not determine any traffic shaping based on our RTT measurements. Neither depending on protocol nor on hour of the day.

4.3.2 Basic UDP measurements (UDP4). The TTL values we measured varied more than for the TCP measurements but we were not able to identify any irregularity. Except for one provider, were UDP port 5060 was blocked as described above. After deactivation of the blockade in the provided modem, we were not able to detect a port block in the second and third measurement period.

4.3.3 Blocked Hosts DNS test (BDNS7). As described, we discovered inconsistencies regarding executed domain blocking. The default DNS server of one provider did return `0.0.0.0`, others returned IPs from their address space, showing legal information about the blockade. Over time, one provider harmonized the IPs for the legal information, but principally the results did not change over time. The detailed values are listed in Table 2. We noticed that one provider was not blocking www.thepiratebay.se in the first and second measurement period, although other domains were already redirected to their information page. In the third and last measurement period, we see the start of the blockade.

4.3.4 Non-existing Hosts DNS test (NDNS7). We discovered different behaviour regarding non-existing domains. One provider resolved the domain to a server with a Google Custom Search in place. Another request stopped with a timeout after five seconds. The detailed values are also listed in Table 2.

[5]www.kinox.to
[6]www.123hjaf9hu32iufhuihoafine.com

Domain	Blocked[5]	Non-existing[6]
ISP1	Provider-specific	Timeout (5s)
ISP2	Provider-specific	Not found
ISP3	Provider-specific	Provider-specific
ISP4	Provider-specific	Not found
ISP5	0.0.0.0	Not found

Table 2: Different DNS results for choosen Austrian ISPs

Client request	Received by Server

`GET /ID/image2.jpg HTTP/1.1`	`GET /ID/image2.jpg HTTP/1.1`
	`Host: XXX.XXX.XXX.34`
	`Connection: keep-alive`

Server response	**Received by Client**

`HTTP/1.1 200 OK`
`Content-Length: 15717310`
`Content-Language: en`
`Content-Type: image/jpeg`
	`HTTP/1.1 200 OK`
	`Content-Length: 15717310`
	`Content-Language: en`
	`Content-Type: image/jpeg`
	`Connection: keep-alive`

Table 3: HTTP alterations made by ISP4

4.3.5 SYN Flooding attack test (SYN4). With our test configuration of 200 SYN packets sent per measurement, we were not able to discover any filtering of SYN packets.

4.3.6 Voice over IP test (VOIP7). When testing Voice over IP connection quality, none of the ISPs had worse connection quality than the others. Since the number of successful measurements was rather low, statistical evaluation is unrewarding.

4.3.7 Malformed TLS handshake test (TLS4). All TLS handshakes, regardless of syntax errors, were received by both client and server without interference for all tested ISPs. We tested interference with invalid length fields, other variants are still possible.

4.3.8 Invalid POP3 syntax test (POP37). All POP3 communication, regardless of syntax errors, was received by both client and server without interference and manipulation for all tested ISPs.

4.3.9 HTTP Antivirus test (VS7). The EICAR testfile was transmitted without any alteration for all tested ISPs for all measurement periods. No in-band anti-virus inspection was detected by this metric. We found the HTTP header changed by one provider as also seen in the results of metric CM7, but the HTTP body was untouched.

4.3.10 HTTP Caching and Manipulation (CM7). We were not able to detect any form of caching for any of the tested providers. For one provider we were able to detect a HTTP Header insertion. A `Connection: keep-alive` header and a `Host` header were inserted for client and server, as shown in Table 3.

4.3.11 Invalid SMTP syntax text (SMTP7). For four tested providers, the complete SMTP communication (including syntax errors) was transmitted without any alteration. One provider changes the traffic going over TCP port 25. `220 2.0.0 Ready to start TLS` is not transmitted, instead it is `421 syntax error (wait for server reponse)`. That would be correct behavior, because we transmitted an invalid status code (136), but it shows the use of a middlebox modifying SMTP data in transmission. This can also be seen in Table 4.

StartTLS stripping test (STLS7). As for SMTP7, four tested providers did not interfere with the STARTTLS command which is used to start an encrypted communication channel within the SMTP communication. One provider – the same as for invalid SMTP syntax – did not perform any STARTTLS stripping, but changed the transmitted data. The first command was transmitted twice, while the `250-PIPELINING` command was suppressed. The detailed traffic is presented in Table 4. This also shows the use of a middlebox modifying SMTP data.

4.3.12 Invalid HTTP syntax test (HTTP7). The results are very similar to metric CM7 as shown in Table 3. Four providers do not alter any traffic, ISP3 adds the HTTP headers *Connection: keep-alive* and *Host*. Syntax errors (e.g., wrong code *HTTP/79.2*) are replaced (e.g., with *HTTP/1.1*).

4.3.13 TCP bandwidth test (TCPS4). We were not able to measure any traffic shaping during the measurement periods depending on the used TCP port. We observed slight changes in bandwidth depending on the time of day, however throughout the day the connection speeds of the different ports stayed the same.

4.3.14 Multimedia test (MM7). With this metric, we were not able to measure any traffic shaping based on the HTTP header fields for any ISP.

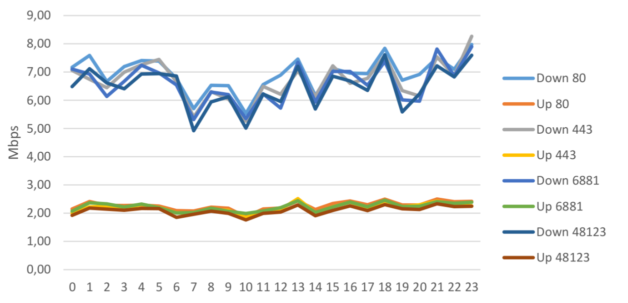
4.3.15 Traceroute test (TRAC3). We were not able to discover any inconsistency regarding the traceroutes we measured for any ISP.

4.3.16 Tor OONI test (OONI7). Consistent with the results of CM7, we were able to detect HTTP header manipulation for one provider with the supplementary OONI metric. We tested four different kinds of modifications: Request Line Capitalization (changes to capitalization (e.g., `GET / HtTP/1.1`, Header Field Number (number of different header fields), Header Name Capitalization or Header field value (value changes).

4.4 Compared measurement periods

We conducted three different measurement periods to validate our results and continuously monitor applied techniques. When we compare the different results, we come to following conclusions:

(1) The majority of metrics did not reveal new results. The chosen techniques we identified tend to be rather stable. Single, reoccurring scans seem to be sufficient to detect ongoing usage.



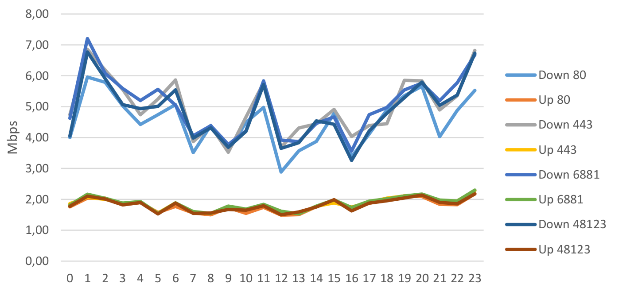


Figure 7: Mean bandwidth of various TCP ports for ISP4 over the day, measurement periods: 2 – top, 3 – bottom, measurement points: 2 – n=1894, 3 – n=1342

(2) Two providers changed their used techniques and replaced their middleboxes. No metric indicated the use of the middlebox on TCP/80. The middlebox on port TCP/25 for SMTP was also removed, but instead a complete blockade was introduced.

5 DISCUSSION

Our paper and framework only considers the technical aspects, as it is the only one we can empirically test. We can only speculate about the reasons specific measures are in place. There are several explanations for different measures. If you take a middlebox for e-mail related ports, you may first see privacy violations but Internet Service Providers will maybe argue differently. Security measures or spam prevention are possible explanations. Our framework helps identifying, measuring and continuously monitoring network anomalies, but the consequences still have to be discussed in the concerned field.

The effort of ongoing continuous scanning vs the insights gained can be discussed. We implemented our system as an ongoing monitoring framework for net transparency, but we found out that most techniques are rather stable. For some metrics, scanning on an hourly basis is not necessary, e.g., DNS behavior, were a reduction to daily scanning is still sufficient to monitor changes in a timely manner. Other metrics, e.g., bandwidth tests could be performed more often

Server	Client
`220 mail.secure.org ESMTP service ready`	`220 mail.secure.org ESMTP service ready`
`EHLO 4684acbf-eec0-4f12-b74f-f6fd9e81c1cc`	`EHLO 4684acbf-eec0-4f12-b74f-f6fd9e81c1cc`
`136-mail.secure.org`	`136-mail.secure.org`
`...`	`...`
`136-8BITMIME`	`136-8BITMIME`
`136 DSN`	`136 DSN`
`220 2.0.0 Ready to start TLS`	`421 syntax error (wait for server response)`
Server	Client
`220 mail.secure.org ESMTP service ready`	`220 mail.secure.org ESMTP service ready`
`EHLO 9cd076a8-7aba-4409-b293-818c53beab28`	`EHLO 9cd076a8-7aba-4409-b293-818c53beab28`
`250-mail.secure.org`	`250-mail.secure.org`
`250-PIPELINING`	
`250-SIZE 15728640`	`250-SIZE 15728640`
`250-VRFY`	`250-VRFY`
`...`	`...`
`220 2.0.0 Ready to start TLS`	`220 2.0.0 Ready to start TLS`

Table 4: SMTP alterations made by ISP3; Above: Syntax Errors (SMTP7); Below: Pipelining (STLS7)

to gain a deeper insight. We decided to remain our test configuration unchanged throughout the study to increase comparability of the results. You could further argue that more complex metrics could be executed to detect clearly questionable techniques like middleboxes or proxies, while more often executed ones give us a more detailed picture.

With the 17 implemented metrics, we tried to cover a wide spectrum of possible technique detection. Still, our work is limited by the number of possible tests. With releasing the source code, we encourage other researchers to implement their own metrics for continuous net transparency monitoring. In addition, metrics could be improved in various ways, e.g., full protocol simulations instead of simple port-based traffic shaping, a greater port range instead of hand-selected ports, or more variance for hostnames used for DNS tests. Still, we believe that our metrics uncover a variety of techniques. Our work is also limited, as we concentrated on five major Internet products of large Austrian-based providers. However, our methodology is applicable for tests with more providers. Also, some manual work had to be done to evaluate the large number of results. An automated detection of questionable techniques could improve this situation.

6 CONCLUSION

In this study, we introduced a Net Neutrality monitoring system. This extensible system – which we published as open-source software – works with a client/server architecture and makes use of fully controlling both endpoints. This enables a wide variety of possible metrics to measure different network management techniques. We introduced 17 different metrics that measure and test different protocols, e.g., basic network measurements, modification and caching tests, e-mail related tests, Voice over IP quality measurements and bandwidth tests. We also utilized state-of-the-art measurements like OONI. The system was then used to monitor techniques in place from five different Austrian Internet products. We found and described several questionable methods. This includes the use of middleboxes for SMTP or HTTP. Also, methods used for blocked and non-existing DNS requests. We conducted our measurements in three periods of three months each.

We communicated all inconsistencies to the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR) after each measurement period.

ACKNOWLEDGMENTS

We would like to thank Dietmar Zlabinger from the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR) for his valuable feedback.

The competence center SBA Research (SBA-K1) is funded within the framework of COMET – Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG. This research was further funded by the Austrian Science Fund (FWF): P30637-N31.

REFERENCES

[1] Collin Anderson, Philipp Winter, and Roya. Global Network Interference Detection Over the RIPE Atlas Network. In *Workshop on Free and Open Communications on the Internet (FOCI)*. USENIX Association, August 2014.

[2] Hsing Kenneth Cheng, Subhajyoti Bandyopadhyay, and Hong Guo. The Debate on Net Neutrality: A Policy Perspective. *Information systems research*, 22(1):60–82, 2011.

[3] Jay Pil Choi, Doh-Shin Jeon, and Byung-Cheol Kim. Net Neutrality, Business Models, and Internet Interconnection. *American Economic Journal: Microeconomics*, 7(3):104–141, 2015.

[4] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, P Krishna Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling End Users to Detect Traffic Differentiation. In *NSDI*, pages 405–418, 2010.

[5] Arturo Filasto and Jacob Appelbaum. OONI: Open Observatory of Network Interference. In *Workshop on Free and Open Communications on the Internet (FOCI)*. USENIX Association, 2012.

[6] Tobias Flach, Pavlos Papageorge, Andreas Terzis, Luis Pedrosa, Yuchung Cheng, Tayeb Karim, Ethan Katz-Bassett, and Ramesh Govindan. An Internet-Wide Analysis of Traffic Policing. In *ACM SIGCOMM 2016 Conference*, pages 468–482. ACM, 2016.

[7] Electronic Frontier Foundation. Net Neutrality. https://www.eff.org/issues/net-neutrality.

[8] Jacob Hoffman-Andrews. ISPs Removing Their Customers' Email Encryption. https://www.eff.org/de/deeplinks/2014/11/starttls-downgrade-attacks.

[9] Arash Molavi Kakhki, Fangfan Li, David Choffnes, Ethan Katz-Bassett, and Alan Mislove. BingeOn Under the Microscope: Understanding T-Mobiles Zero-Rating Implementation. In *Proceedings of the 2016 workshop on QoE-based Analysis and Management of Data Communication Networks*, pages 43–48. ACM, 2016.

[10] Sheharbano Khattak, David Fifield, Sadia Afroz, Mobin Javed, Srikanth Sundaresan, Damon McCoy, Vern Paxson, and Steven J Murdoch. Do You See What I See? Differential Treatment of Anonymous Users. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2016.

[11] Robin S Lee and Tim Wu. Subsidizing Creativity through Network Design: Zero-Pricing and Net Neutrality. *The Journal of Economic Perspectives*, 23(3):61–76, 2009.

[12] Gabi Nakibly, Jaime Schcolnik, and Yossi Rubin. Website-Targeted False Content Injection by Network Operators. In *USENIX Security Symposium*, pages 227–244, 2016.

[13] Body of European Regulators for Electronic Communications (BEREC). All you need to know about Net Neutrality rules in the EU. http://berec.europa.eu/eng/net/.

[14] Christoph Sölder, Leonhard Wimmer, Dietmar Zlabinger, Ulrich Latzenhofer, Ursula Prinzl, Philipp Sandner, Lukasz Budryk, and Ulrich Liener. RTR Multithreaded Broadband Test (RMBT) Specification. Oct 2015. https://www.netztest.at/doc/.

[15] Nicholas Weaver, Christian Kreibich, and Vern Paxson. Redirecting DNS for Ads and Profit. In *Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[16] Xing Xu, Yurong Jiang, Tobias Flach, Ethan Katz-Bassett, David Choffnes, and Ramesh Govindan. Investigating Transparent Web Proxies in Cellular Networks. In *Conference on Passive and Active Network Measurement (PAM)*, pages 262–276. Springer, 2015.