NeutMon: Studying Neutrality in European Mobile Networks

Enrico Gregori, Valerio Luconi
IIT-CNR, Pisa, Italy
Email: enrico.gregori@iit.cnr.it, valerio.luconi@iit.cnr.it

Alessio Vecchio
Dip. di Ingegneria dell'Informazione, University of Pisa, Italy
Email: alessio.vecchio@unipi.it

***Abstract*—Net neutrality is the principle that all data on the Internet should be treated in the same way, without discrimination by content, application, or service. Research about net neutrality mostly focused on the wired Internet, and little effort has been devoted to wireless scenarios. However, mobile devices are now the main access medium to the Internet for a large fraction of users and this trend is expected to continue in the next years.**

In this paper, we study net neutrality in a European mobile broadband scenario using NeutMon, a tool specifically designed for being executed in the MONROE testbed. MONROE's nodes are connected to 13 mobile broadband providers spread in four European countries. Preliminary results show that in this set of operators differentiation is enforced, in terms of bandwidth, for a commonly used peer-to-peer application, in contrast with recent EU regulations.

***Index Terms*—Net neutrality, network measurement.**

I. Introduction

According to the Internet neutrality principle a network should treat all traffic in the same way, without intentional degradation of performance depending on source/destination of packets and application type. Several examples of non-neutral behavior have been reported in the last years, and this fueled the debate at different levels, from the merely technical perspective to the economic, legal, and moral ones [1], [2]. The recent repealing of net neutrality rules in the USA fueled again the discussion on this important issue, and highlighted the necessity of tools able to detect possible traffic differentiation performed by network operators. Notable examples of discriminatory policies include blocking/degrading of bandwidth hungry applications (such as peer-to-peer or video streaming) and of Internet-based services that may commercially compete with ISPs (such as voice-over-IP).

EU-wide rules concerning net neutrality [3] are considered as one of the major achievements towards the Digital Single Market. According to these rules, blocking, throttling, and discrimination of traffic by ISPs is not allowed. All traffic has to be treated equally, and no form of traffic prioritization can be enforced. Only few exceptions are allowed: preserving the integrity of the network, managing temporary congestions, and compliance with legal obligations.

The degree of interference applied by non-neutral ISPs may vary greatly, from complete blocking to a degradation of performance that is so soft to be almost invisible to the end users. For instance BitTorrent traffic has been both blocked using forged TCP reset packets and limited, in terms of bandwidth, by means of traffic shaping mechanisms [4], [5]. In some cases the discrimination policies are applied only during specific time periods, e.g. corresponding to peak hours. These factors, together with the inherent variability of network conditions, make the detection of neutrality violations not always straightforward.

So far, research on net neutrality focused on the wired part of the Internet. However, in recent years, smartphones and tablets have become the preferred choice for accessing a large number of networked services and applications, from social networks to video streaming. This paper presents a study on net neutrality in mobile operators' networks. The study has been conducted using MONROE, a testbed with hundreds of nodes connected to 13 operators and distributed in four European countries (Italy, Norway, Spain and Sweden) [6]. To this purpose we designed and implemented NeutMon, a system aimed at studying net neutrality in the MONROE context. The tool can be used for collecting network metrics related to net neutrality (in particular throughput). This is done by producing traffic belonging to different classes and then comparing the observed network characteristics. To the best of our knowledge, this is the first study performed on a significant portion of European mobile network operators, especially after the emission of EU rules.

The rest of the paper is structured as follows. Section II shows the state of the art in net neutrality measurements. Section III introduces NeutMon and the measurement methodology. In Section IV we describe NeutMon's architecture, and in Section V we show how NeutMon is implemented in MONROE. Sections VI and VII show experimental results. The former shows the validation of the NeutMon tool in a controlled environment, whereas the latter shows results of measurements carried out in the MONROE testbed. Finally, Section VIII concludes the paper.

II. Related work

Tools for detecting violations of net neutrality explored approaches based on both passive and active measurements.

Glasnost is a system that allows ordinary Internet users to detect differentiation in ISPs [7]. To make the system easy to use, the client is implemented as a Java applet. Thus, to start a test, the user is just required to point the browser to a Web page. The client communicates with one of the server replicas, hosted on the M-Lab infrastructure [8]. The test is based on the experienced difference of throughput between the application suspected to be differentiated and a random flow with similar characteristics. Glasnost initially focused on BitTorrent, but subsequently the possibility of adding tests related to new applications has been introduced. In particular, a new application can be incorporated by first recording its trace at packet level and then submitting the trace to the system. Glasnost services have been shut down since February 2017.

Neubot is a collaborative approach to measure the neutrality of the Internet [9]. Once installed, the Neubot client periodically monitors the quality of service by running active measurements in the background. The system includes a set of servers used as endpoints during measurements and a central database where results are collected.

Tariq et al. [10] explored the possibility of detecting network neutrality violations by passively observing network metrics on clients. The system, called NANO, is composed of client agents, which collect data on users machines, and a server, where information produced by agents is aggregated and analyzed. The technique includes statistical inference detect network neutrality violations even in the presence of confounding factors.

NetPolice is a tool conceived to detect content- and routing-based differentiation in backbone ISPs [11]. This is done by measuring the loss rate when generating traffic belonging to different applications (HTTP, BitTorrent, SMTP, PPLive, and VoIP). HTTP is used as the baseline to detect the presence of differentiation for the other applications. The TTL of probing packets is varied to evaluate the loss rate at the desired hop. Detection relies on the Kolmogorov-Smirnov test, which operates on the empirical distributions of the applications to be compared. Resampling is used to increase robustness against noise. The method includes mechanisms for parsimonious selection of probing targets.

Other works are aimed at detecting the presence of a traffic shaper on a network path. ShaperProbe can not on only infer the presence of traffic shaping mechanisms, but it is also able to estimate some of the shaping characteristics [12]. ShaperProbe relies on a client that sends probes at a constant bit-rate equal to the capacity of the narrowest link. The server measures and observes the received rate at the other endpoint. A level shift in the received rate is used to infer the presence of traffic shaping mechanisms. Extensive experiments, carried out using the M-Lab infrastructure, highlighted a significant presence of such mechanisms in major ISPs.

DiffProbe is a tool for detecting ISP service discrimination [13]. The flow of an application that is supposed to be discriminated is compared with a probing flow. The probing flow is derived from the application flow using a combination of size, payload, and port randomization. The application flow is generated from a previously recorded application trace (Skype and Vonage are considered in the experiments). Then the two flows are compared to detect statistically significant variations in terms of loss rate and/or delay (using the two-proportion z-test and Kullback-Leibler divergence respectively). The method is able to identify the mechanism used by the ISP for discrimination (Strict Priority, Weighted Fair Queuing).

Few works tackled the problem of detecting differentiation in mobile networks. A method tailored to wireless environments is described in [14]. The method is based on the idea of using a VPN proxy to record a trace of a generic networked application. The trace is subsequently replayed and network metrics are compared to the ones collected when using the encrypted tunnel (and thus in general not subject to differentiation). The method has been verified using a testbed which included two commercial products commonly used for traffic shaping. An implementation of the method is available for Android-based smartphones. Experimental results showed the presence of traffic differentiation policies in some mobile network operators.

BonaFide is a tool for mobile environments able to detect differentiation on a set of application protocols: BitTorrent, HTTP, Flash video, Real Time Streaming Protocol, VoIP H323, Session Initiation Protocol (SIP) [15]. Experiments on some mobile network operators identified a transient presence of traffic shaping for SIP.

This paper contributes to existing literature by studying the neutrality of a large set of European mobile network operators using the MONROE testbed. To the best of our knowledge, this is the first study carried out after the introduction of EU rules about net neutrality [3]. The Internet is more and more accessed via high speed mobile technologies, and we strongly believe that a monitoring system useful to assess if rules are respected is of paramount importance for both experts and end users.

III. Method

NeutMon is aimed at detecting violations of net neutrality in the path between two endpoints (a client and a server), in terms of throughput experienced by different applications. To this purpose, we implemented a speed test aimed at measuring the application-level throughput of the connection between the client and the server. The test is performed in both uplink and downlink directions (with respect to the client) for two classes of traffic: BitTorrent traffic (BT) and Random traffic. Random traffic is used as a baseline, thus hereafter we will use the term Control traffic (CT).

The BT speed test implements a data exchange between the client and the server. In the uplink phase the client transfers data to the server; in the downlink phase data goes from the server to the client. The BT exchange is regulated by the BitTorrent Protocol Specification [16], which is an application-level protocol (such as HTTP, FTP, etc.). To obtain comparable results, CT follows the same pattern of BT in terms of size and sequence of messages, but the payload of CT is completely random (i.e. the application-level content is a string of randomly generated bytes).

In the following we describe the messages exchanged with BT in the uplink direction (downlink is symmetrical). The test is made of two phases: i) a preliminary phase, and ii) the data transfer phase, in which measurements are taken. In the preliminary phase, client and server communicate to set up the actual data transfer. First, client and server exchange BitTorrent handshake messages. This phase is initiated by the server. Handshake messages are needed to identify the two endpoints of the connection as BitTorrent speakers and to specify which file one endpoint (the server in this case) is willing to obtain. Then the client sends to the server an *unchoke* message, which informs the server that it is allowed to send data requests. The server sends back an *interested* message (which means that the server is interested to the client content) that terminates the preliminary phase. The data transfer phase is started by the server with requests for data chunks. For each data request the client sends a data chunk. The payload (i.e. the data portion of a BitTorrent packet) is randomly generated, as this part of the packet is not used to identify traffic and possibly differentiate it. The test duration is configurable and by default is set to 10 seconds. After this time the client completes pending data requests and then sends a *choke* message, which means that the server is no more allowed to send data requests. This message terminates the speed test. The throughput is measured by the server. Starting from the first data request, each time the server receives data, it stores the time of arrival and the amount of data received. This is done for being able to compute instantaneous and average throughput, and the throughput distribution. As specified above, the downlink phase is identical to the uplink phase, but the roles are swapped.

Both BT and CT speed tests are implemented on top of TCP. For each traffic direction (uplink or downlink) and for each class of traffic, the speed test is performed on a separate connection, i.e. the connection is opened right before the test for a direction/traffic, and closed right after. Besides the throughput of the two classes of traffic, the speed test is able to detect also if a port is blocked or if a given traffic flow is blocked upon recognition. The complete sequence of tests is the following. First, the BT uplink (with respect to the client) traffic is tested on port 6881, which is the default port for BitTorrent. If port 6881 is blocked, a random high port is chosen and the test is repeated (some works highlighted that such port numbers are frequently used by peer-to-peer applications and thus adopted as classification features [14]). Then, the CT uplink speed test starts on the same port used for BitTorrent (6881 or the random high port). The downlink phase is then run for both BT and CT with the port identified at the first step (BT uplink).

IV. Architecture

As mentioned in the previous section, the tool consists of a client and a server. The client is executed on MONROE nodes, whereas the server is executed on a dedicated machine. Client and server communicate using two channels. One of the channels is the coordination channel, which is used to organize the activities between client and server. The other channel is used to perform tests. More in detail, the coordination channel is used to synchronize client and server and to make them communicate out-of-band for the following operations: i) start

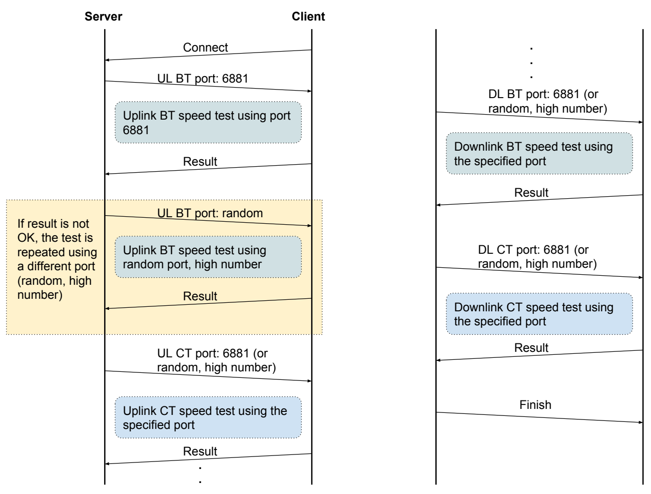

Server
Client
Connect
UL BT port: 6881
Uplink BT speed test using port 6881
Result
If result is not OK, the test is repeated using a different port (random, high number)
UL BT port: random
Uplink BT speed test using random port, high number
Result
UL CT port: 6881 (or random, high number)
Uplink CT speed test using the specified port
Result
DL BT port: 6881 (or random, high number)
Downlink BT speed test using the specified port
Result
DL CT port: 6881 (or random, high number)
Downlink CT speed test using the specified port
Result
Finish

Fig. 1: Interaction between client and server

of new tests, ii) transfer of results, iii) aborting measurements (if necessary).

The coordination channel is established at the beginning of a measurement session. The server listens for connections using port 10000 (this is the default port, but another one can be configured). If the coordination channel fails, the measurement is canceled and the server goes back into the listening state. A separate channel for coordination is useful because:

- Measurements are carried out in the two directions. During the downlink phase, results are computed on the client. During the uplink phase, results are computed on the server. The client transfers its own results to the server using the coordination channel. On the server, all results are saved onto persistent storage.
- If an ISP blocks BitTorrent traffic (a possible violation of net neutrality), client and server are still able to communicate using the coordination channel. In this case, a communication block is reported and logged.

The server cyclically operates as follows: i) it creates a socket and starts accepting connections; ii) when a client connects, it performs the measurement operations. The server processes the requests coming from a single client at a time. This is done to avoid interferences during the measurement phase caused by cross-traffic and increased load. Clients that desire to carry out a measurement when the server is busy are queued, and they will be served as soon as the current measurement completes. In any case, the server requires a public IP address or, if the server is located behind a NAT, port forwarding is required. When all the tests are finished, results are written to a file in JSON format.

The client is implemented as an infinite loop that is ended only by a finish or abort message from the server, or if no coordination messages are received in a certain amount of time. When a coordination message is received, the client starts the corresponding test and, when finished, sends back the results in JSON format to the server. Figure 1 shows the interaction between client and server.

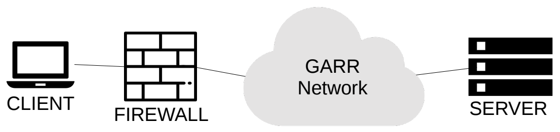


Fig. 2: Validation architecture

V. Implementation on the MONROE testbed

NeutMon has been implemented to run in the MONROE platform, a mobile broadband testbed including over 400 nodes spread in four European countries (Italy, Norway, Spain, and Sweden) [6]. Nodes are connected to the Internet via mobile operators. A single node can carry up to two SIM cards of different operators. The total number of available operators is 13. Table I summarizes MONROE operators grouped by country (first two columns). The platform includes both static nodes and mobile nodes installed on trains, trucks, or buses. Nodes provide operating-system-level virtualization (also known as containerization). A container is an isolated user-space instance that can see only the resources and devices that the kernel is assigning to it. This is done for security reasons, to prevent software executed on nodes from harming the MONROE platform. To run experiments on MONROE, a user must build a docker container with installed the experiment's software [17]. Then, the container can be scheduled for execution on MONROE nodes via a web interface.

In MONROE, when an experiment gets assigned a time slot on a node, it is executed in mutual exclusion with other experiments (on the same node). This ensures that the network metrics observed by NeutMon are not distorted by cross traffic generated by other experiments on that node.

MONROE software executed on all nodes provides relevant contextual information to interested applications. Information is provided according to a publish-subscribe paradigm. Examples of published information include the position of the node, the network the node is attached to, modem events, etc. NeutMon used this mechanism to collect data that can be useful to distinguish intentional violations from normal variability of network conditions (e.g. RSSI).

MONROE nodes run a Debian operating system, and provide to containers a limited set of resources. To ensure portability of code on MONROE nodes and speed the implementation process up, we implemented NeutMon in the Python language, using only libraries that are available to MONROE users. The NeutMon code is open source and publicly available at the NeutMon project website[1].

VI. Validation

To validate the proposed tool we ran a set of measurements in a controlled environment. The test setup was made of the following three elements: i) a client machine located at IIT-CNR in Pisa, ii) a server machine located at the University of Pisa, iii) a Palo Alto firewall located at IIT-CNR. Palo Alto[2] is a tool able to perform application recognition via deep packet inspection (DPI). It is also able to block, shape and route traffic according to user-assigned policies.

The University of Pisa and the IIT-CNR networks both run at 1 Gbps and they are connected via the GARR network (the Italian research network), which is a high speed network with links with a capacity higher than 20 Gbps. The validation architecture is depicted in Figure 2. Since the implementation of NeutMon is the same for uplink and downlink, we just tested one phase, in particular the uplink phase.

We tested the ability of NeutMon in identifying the presence of throttling for one of the two classes of traffic. We set up the Palo Alto firewall to shape BT for obtaining a 4 Mbps maximum throughput (while CT was unlimited). We ran the test ten times to ensure reliability. In each of the ten runs NeutMon measured a BT throughput of exactly 4 Mbps, while CT obtained a 689 Mbps average throughput. We can thus state that NeutMon is able to correctly measure the available bandwidth and to recognize when BT is subject to limitations.

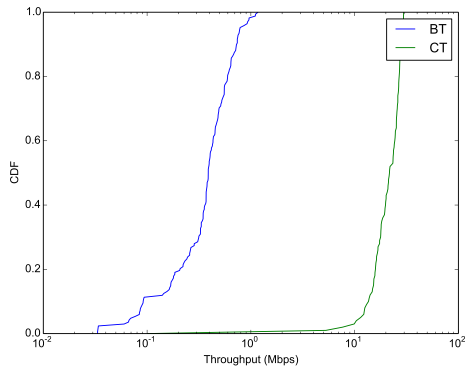


Fig. 3: Empirical CDF of throughput for the first run at 02:00 for Vodafone Italy

VII. Results

We ran a measurement campaign with the 13 operators available in MONROE. For each operator we ran measurements across a 24 hours period. Measurement sessions were carried out at four time slots, respectively at 2:00, 8:00, 14:00, and 20:00, to possibly discover differentiation policies that are applied only during peak hours. During each session, at least three runs of the tool were executed. This way, we obtained at least three results for both BT and CT at each session. Each run was executed 30 minutes after the previous one (e.g., for the third time slot, experiments were run at 20:00, 20:30 and 21:00). In some cases we had more than three runs (some experiments partially failed because of runtime problems, such as connectivity losses, and we had to repeat them). Experiments were conducted in November-December 2017.

[1]http://vecchio.iet.unipi.it/neutmon/repository/

[2]https://www.paloaltonetworks.com/

For each run we computed the empirical CDFs of the measured throughput for both BT and CT, in the downlink and uplink cases. Some differences between the two CDFs are particularly evident even at first sight. For example let us consider the empirical CDFs for Vodafone Italy collected at 02:00 shown in Figure 3. As can be seen, the two CDFs are very distant, and BT experiences much poorer performance than CT. In particular, almost all BT throughput values are less than or equal to 1 Mbps, while almost all CT throughput values are greater than 10 Mbps. This suggests the presence of throttling mechanisms applied to BT. However, in other cases the difference is not so clear, or there are differences that may have been caused by network variability rather than intentional differentiation.

We computed the boxplots of the downlink mean throughput values obtained by BT and CT by all operators at the different times (Figure 4). Uplink plots are not shown as they do not add significant information. Table I summarizes the results for all operators. In the table we show if we detected blocking on port 6881, and, if so, which percentage of our runs were affected by blocking. In addition we highlight the suspected cases of throttling.

As can be seen, Norwegian operators do not seem to apply any form of differentiation. In some cases BT obtains a lower performance than CT (for example, with Telenor Norway (Figure 4e) at 14:00 and 20:00), but these differences can be easily explained by the variability of network conditions. In fact, in the other experiments, the performance of both BT and CT is always good and comparable.

In Italy, we found that Vodafone Italy and Wind both block traffic on port 6881 in certain runs. The first also seems to apply some forms of throttling. Vodafone tests experienced blocking on port 6881 in all runs except those executed at 02:00. This could indicate that traffic on port 6881 is blocked only in "working hours". In addition, BT traffic on a random high port always obtains very poor performance (max $\sim$ 0.3 Mbps). CT usually obtains better results, but in some cases it obtains poor performance as well (Figure 4b). However the lack of variability of BT results could confirm that its performance is due to traffic shaping. Results also seem to indicate that Vodafone classifies traffic via deep packet inspection, since BT and CT are exchanged between the same hosts and ports but CT is not affected. Also Wind performs blocking on port 6881 in some runs (approximately $\sim 41\%$ of the times). In particular, our tests registered blockages in almost all runs at 02:00 and 20:00.

In Spain, Vodafone Spain shows the same behavior of Vodafone Italy. This is not surprising, as they are owned by the same company. Yoigo always blocks traffic on port 6881. In addition, Yoigo seems to throttle both BT and CT that always obtain extremely low performance. This could happen as some shapers classify traffic using high port numbers as peer-to-peer [14], and Yoigo could adopt such monitoring mechanism, less refined than deep packet inspection.

In Sweden, our tests with Telenor Sweden experienced blocks on port 6881 in 50.3% of the cases, mainly in the busiest hours (14:00 and 20:00). Moreover, the performance obtained by both BT and CT is in general very low.

TABLE I: Measurement results.

Country	Operator	Port 6881 blocked	Throttling
Italy	TIM	0%	
	Vodafone	86.4%	BT (sometimes CT)
	Wind	41.2%	
Norway	ICE	0%	
	Telenor	0%	
	Telia Mobile	0%	
	Telia Norge	0%	
Spain	Orange	0%	
	Vodafone	73.9%	BT (sometimes CT)
	Yoigo	100%	Further investigation needed
Sweden	H3G	0%	
	Telenor	58.3%	Further investigation needed
	Telia Mobile	0%	

For all the other operators nothing significant emerges from obtained results.

VIII. Conclusion

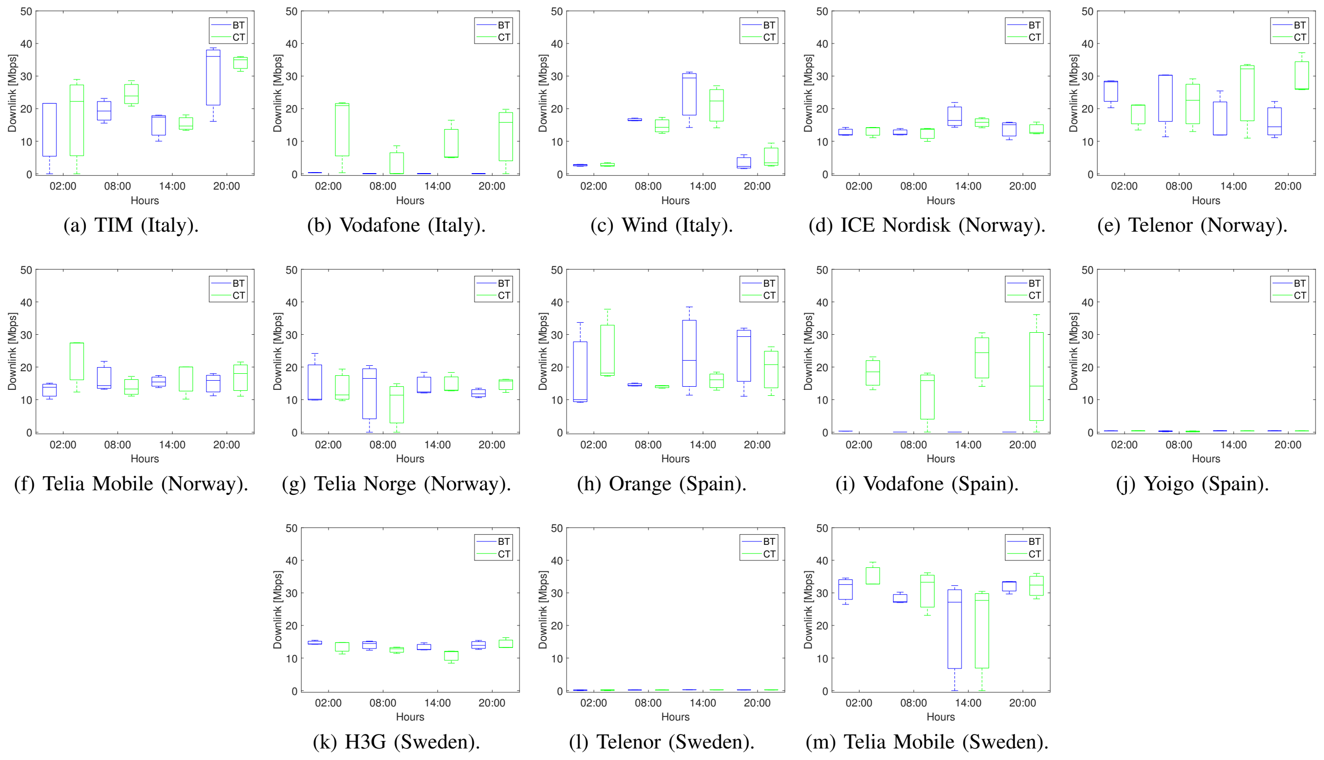
Mobile networks have become the preferred choice for accessing a large number of networked services and applications, from social networks to video streaming. This trend is expected to continue in the next few years because of both technological advancements (e.g. 5G) and regulations (e.g. those related to roaming in EU). This motivates the need for additional studies about net neutrality in a wireless scenario. In this paper we performed a study on 13 mobile operators of four European countries with NeutMon, a tool that helps end users assess the neutrality of network operators. Experiments have been carried out using the MONROE testbed.

This study demonstrates that even nowadays, despite EU regulations, in three out of four of the considered countries, at least one major mobile network operator seems to show some forms of neutrality violation.

Future work will focus on implementing NeutMon for other platforms, such as smartphones, in order to gather data from a larger number of mobile network operators. In addition, more measurements will be carried out, to deepen the analysis on those operators that showed suspected behavior. We also plan to evaluate the benefits/drawbacks of a testbed-based approach with respect to a data collection campaign based on crowdsourcing (which has been successfully used for monitoring large-scale networks [18], [19]). Finally, NeutMon will be extended to also detect the path traversed by the different classes of traffic, to evaluate if differentiation is also performed in terms of routing.

Acknowledgment

This work is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644399 (MONROE) through the second open call for projects. The views expressed are solely those of the authors.



(a) TIM (Italy). (b) Vodafone (Italy). (c) Wind (Italy). (d) ICE Nordisk (Norway). (e) Telenor (Norway).

(f) Telia Mobile (Norway). (g) Telia Norge (Norway). (h) Orange (Spain). (i) Vodafone (Spain). (j) Yoigo (Spain).

(k) H3G (Sweden). (l) Telenor (Sweden). (m) Telia Mobile (Sweden).

Fig. 4: Downlink average throughput for BT and CT at different times.

References

[1] B. Obama, "Net Neutrality: President Obama's Plan for a Free and Open Internet," https://obamawhitehouse.archives.gov/node/323681, 2008, [Online; accessed 4-october-2017].

[2] J. Pil Choi and B.-C. Kim, "Net neutrality and investment incentives," *RJE*, vol. 41, no. 3, pp. 446–471, 2010.

[3] "BEREC Guidelines on the Implementation by National Regulators of European Net Neutrality Rules," http://berec.europa.eu/eng/document_register/subject_matter/berec/download/0/6160-berec-guidelines-on-the-implementation-b_0.pdf, 2016, [Online; accessed 6-october-2017].

[4] N. Weaver, R. Sommer, and V. Paxson, "Detecting Forged TCP Reset Packets," in *Proc. NDSS '09*, 2009.

[5] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi, "Detecting Bittorrent Blocking," in *Proc. ACM SIGCOMM IMC '08*, 2008, pp. 3–8.

[6] Ö. Alay, A. Lutu, R. García, M. Peón-Quirós, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. S. Khatouni, M. Mellia, M. A. Marsan, R. Monno, and H. Lonsethagen, "Measuring and assessing mobile broadband networks with MONROE," in *Proc. IEEE WoWMoM '16*, 2016, pp. 1–3.

[7] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: Enabling End Users to Detect Traffic Differentiation," in *Proc. USENIX NSDI '10*, 2010, pp. 27–27.

[8] C. Dovrolis, K. Gummadi, A. Kuzmanovic, and S. D. Meinrath, "Measurement Lab: Overview and an Invitation to the Research Community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 3, pp. 53–56, 2010.

[9] S. Basso, A. Servetti, and J. C. D. Martin, "The network neutrality bot architecture: A preliminary approach for self-monitoring of Internet access QoS," in *Proc. ISCC '11*, 2011, pp. 1131–1136.

[10] M. B. Tariq, M. Motiwala, N. Feamster, and M. Ammar, "Detecting Network Neutrality Violations with Causal Inference," in *Proc. ACM CoNEXT '09*, 2009, pp. 289–300.

[11] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting Traffic Differentiation in Backbone ISPs with NetPolice," in *Proc. ACM SIGCOMM IMC '09*, 2009, pp. 103–115.

[12] P. Kanuparthy and C. Dovrolis, "ShaperProbe: End-to-end Detection of ISP Traffic Shaping Using Active Methods," in *Proc. ACM SIGCOMM IMC '11*, 2011, pp. 473–482.

[13] ——, "Diffprobe: Detecting ISP Service Discrimination," in *Proc. IEEE INFOCOM'10*, 2010, pp. 1649–1657.

[14] A. Molavi Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. Choffnes, P. Gill, and A. Mislove, "Identifying Traffic Differentiation in Mobile Networks," in *Proc. ACM SIGCOMM IMC '15*, 2015, pp. 239–251.

[15] V. Bashko, N. Melnikov, A. Sehgal, and J. Schönwälder, "BonaFide: A traffic shaping detection tool for mobile networks," in *Proc. IFIP/IEEE IM '13*, 2013, pp. 328–335.

[16] B. Cohen, "The BitTorrent Protocol Specification," http://www.bittorrent.org/beps/bep_0003.html, 2017, [Online; accessed 6-october-2017].

[17] "Docker," https://www.docker.com/.

[18] A. Faggiani, E. Gregori, L. Lenzini, V. Luconi, and A. Vecchio, "Smartphone-based crowdsourcing for network monitoring: Opportunities, challenges, and a case study," *IEEE COMMAG*, vol. 52, no. 1, pp. 106–113, January 2014.

[19] E. Gregori, A. Improta, L. Lenzini, V. Luconi, N. Redini, and A. Vecchio, "Smartphone-based crowdsourcing for estimating the bottleneck capacity in wireless networks," *Elsevier JNCA*, vol. 64, no. Supplement C, pp. 62–75, 2016.