Hic Sunt Proxies: Unveiling Proxy Phenomena in Mobile Networks

Raffaele Zullo[†], Antonio Pescapé[†] Korian Edeline[‡], Benoit Donnet[‡],
[†]Università di Napoli Federico II, Italy - r.zullo@studenti.unina.it, pescape@unina.it
[‡]Montefiore Institute, Université de Liège, Belgium - firstname.name@ulg.ac.be

***Abstract*—Over the years middleboxes have established themselves as a solution to a wide range of networking issues, progressively changing network landscape and turning the end-to-end principle into a reminder of an *Arcadian* age of the Internet. Among them, proxies have found breeding ground especially in mobile networks that, moreover, have become the most popular way to access the Internet.**

In this paper, we present Mobile Tracebox, an Android measurement tool, and describe how its methodology, coping with the lack of privileges of mobile devices, can not only detect proxies but also characterize different facets, from their transport layer behavior to their location inside the network. Data collected from a crowdsourced deployment over more than 90 carriers and 350 Wi-Fi networks contributes to describe the potential of the tool and to draw a panorama of proxies across mobile networks. Our study confirms their prevalence and reveals that their scope is not limited to HTTP but can include several TCP services and even non standard ports. We detail the different implementations observed and delve into specific aspects of their configuration, like the initial Receive Window, the Window Scale factor or the set of Options supported, to understand how proxies can affect performance or obstruct extensions. Finally, we focus on fingerprinting and attempt to draw a dividing line between packet modifications performed by proxies and those performed by other classes of middleboxes.

I. Introduction and Related Work

Internet landscape has gradually changed over the years with middleboxes relentlessly gaining ground and embracing a broadening range of functions. The end-to-end principle, a pillar of Internet's early days [1], has increasingly come into question from various directions [2], [3]. Not only network infrastructure has changed but also the way we access the Internet has mutated with the rise of mobile devices at the expense of desk-based and notebook computers. Mobile networks, indeed, with their limited resources (in terms of spectrum, IP address space, etc.) and their innate complications due to user mobility have revealed themselves as the perfect environment for middleboxes proliferation.

Proxies are one of the most popular middleboxes [4], acting as an intermediary between a client and a server. They can be deployed for a wide range of reasons: from enhancing performance on network paths where native performance suffers due to characteristics of the link (Performance Enhancing Proxies) [5] to serving specific tasks at application layer, e.g., HTTP caching, transcoding, even malicious code filtering. Unfortunately, they may also serve less benevolent purposes like content blocking or censorship and sometimes can expose user's security and privacy to unexpected vulnerabilities; even tasks implemented to improve performance, e.g., lowering images quality to reduce fetch time, can on the other hand lead to user's QoE (Quality of Experience) degradation.

Recent works have shed the light on the prevalence of middleboxes and their impact. Sherry et al. [4] demonstrated that networks may contain as many middleboxes as routers, while McQuistin et al. [6] showed how middleboxes are crippling the deployment of transport layer alternatives. In the last few years, the measurement paradigm has also evolved [7] with the rise of crowdsourcing: a novel approach in which users can cooperate to collect measurement data drawing a panorama of Internet that cannot be achieved from fixed observation points [8]. With this approach Wang et al. [9] surveyed more than 100 carriers revealing NAT and firewall policies. Weaver et al. [10], leveraging a wide crowdsourced dataset and multiple detectors at the transport and application level, drew a detailed taxonomy of web proxies based on the purposes for which they are deployed. Vallina-Rodriguez et al. [11] analyzed 71 mobile providers in 6 countries showing, along with network configurations, business models and relationships between operators, how pervasive HTTP and DNS proxies are. Xu et al. [12] used an experimental testbed to investigate transparent web proxies in the four major US mobile providers and how they behave in presence of real web workloads. Recent works has also shown how proxies can degrade performance instead of improving it compared to the old fashion end-to-end communication [13]. Furthermore, Honda et al. [14] focused on how proxy can interfere with TCP options. While it's acknowledged that proxies are part of today's Internet, it is not well known if they are limited to a few services (HTTP, DNS) and what is their detailed behavior at transport level.

In this paper we present Mobile Tracebox, a network measurement app for Android that embodies different techniques to cope with the multifaceted nature of proxies as well as with the innate limitations of mobile devices as measurement tools, and provide an overview of proxies in mobile Internet resulting from its crowdsourced deployment. Unlike other works that rely on application-level interference to discover proxies, we investigate TCP-terminating proxies by analyzing TTL alterations. This broadens the range of detectable proxies to all TCP ports, even non-standard. To complete the picture we also investigate the presence of packet-rewriting proxies that modify traffic as it flows through them without splitting the connection. Furthermore we detail the different configurations observed and discuss their impact on several performance

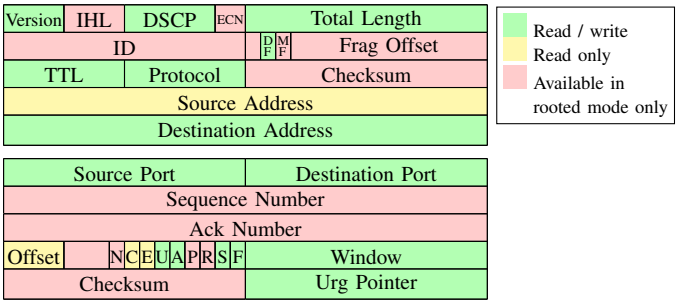


Fig. 1: Mobile Tracebox: IP and TCP header of TCP probe packets forged in non rooted mode.

aspects, including the initial and the maximum TCP Receive Window, the handshake latency, the set of TCP extensions supported. Finally we propose a new detection methodology based on packet modifications performed by proxies and not by other middleboxes.

II. Mobile Tracebox

Methodologies to detect proxies, and more generally middleboxes, are diverse and can rely on active measurements as well as passive measurements [15]. Mobile Tracebox [16], belongs to the first school of thought: it sends specially crafted packets to highlight intermediate boxes that modify those packets or alter the path between source and destination. The way the app forges probe packets is dual: when root privileges are available it relies on raw sockets to set every single bit of the packet, when root privileges are not granted it uses system calls on regular sockets to manipulate IP and transport headers of the packet sent (other than the payload). In rooted mode, that has been presented in our previous work [17], the potentially altered copy of the probe packet is retrieved either from a controlled server or from the quoted packet inside ICMP Time Exceeded messages generated during traceroute[18]. To cope with the lack of privileges of Android devices available to the vast majority of users and thus achieve a wider audience for a crowdsourced measurement campaign, Mobile Tracebox also resorts to non raw sockets to send probe packets: using `bind()`, `setsockopt()`, etc, the value of a number of IP and TCP header fields can be explicitly set or at least manipulated; for a few other fields, that cannot be altered, it is still possible to retrieve the value set by default. Probe packets forged in non rooted mode are sent to a controlled server that returns them to the app in the exact shape they have been received: the app in turn partially reconstructs the packet sent and then compares the fields eligible to be tested. Fig. 1 summarizes IP and TCP header fields that can be tested in non rooted mode. In addition the following TCP Options can be tested on `SYN` probe packets: Maximum Segment Size (MSS), Sack Permitted (SP), Timestamp (TS) and Window Scale (WS). The app can detect if each Option is removed by a middlebox and for MSS and WS if the Option value is altered along the path (for TS the value cannot be tested, while SP has no explicit Option value). It is also possible to detect if those Options are present in the received `SYN` `ACK` allowing to test both directions of the path. Starting from version 6, TCP Fast Open is available on Android (using the experimental kind, 254, and the magic number 0xF989): a specific probe to test it is included in the suite. Table I summarizes TCP Options supported in non rooted mode. Since direct access to ICMP messages requires root privileges, the standard tracebox probe cannot be ported to non rooted devices. In order to test path length it is replaced by a pseudo-traceroute probe in which a regular TCP socket is used and `connect()` is iteratively called with incremental TTL: this allows to identify the number of hops necessary to reach the responding host, whether the intended destination or a TCP-terminating proxy. The range of non rooted probes is completed by a non-responding test in which a `SYN` is sent to a non-responding server to discover a spoofed `SYN` `ACK`. Mobile Tracebox makes use of payload packets other than `SYN` packets for probing: for instance it can send well crafted HTTP requests allowing to detect interference also at application layer. Although it is the main service tested, probes are not limited to HTTP (port 80): available probes also test FTP (21), SMTP (25), HTTPS (443), SIP (5060) and a non standard port (10000). Destination port as well as other parameters can be widely customized by users.

TABLE I: Mobile Tracebox: TCP Options on `SYN` probe packets forged in non rooted mode.

Option	Kind	Value	Note
MSS	2	Read / write	
WS	3	Read / write	
SP	4	-	
TS	8	No access	
TFO (Exp.)	254	-	Android 6.0 or later

Measurement campaign. We deployed Mobile Tracebox by the means of crowdsourcing, recording more than 800 downloads. In this work we analyze data collected in the first 30 months after the release of the non rooted version of the app, from July 2016 till December 2018. Due to the nature of our analyses we had to exclude networks tested only in rooted mode and also tested in non rooted mode but without preserving the default settings on the analyzed fields. In light of this the dataset studied in this work covers 96 carriers and 385 Wi-Fi networks in 69 countries. We will resort to rooted probes collected from a subset of these (17 cellular and 32 Wi-Fi networks) only to delve into a few specific aspects that could not be discerned otherwise. Although the app supports IPv6 in both modes, IPv6 probes collection was quite limited, especially from carriers, and, thus, this work will focus on IPv4 measurements only.

III. Unveiling proxies

Unlike other works that rely on application-level interference [11] [19] [20] [21] to reveal the presence of a proxy along a path, we primarily rely on the concept that a proxy splits a TCP connection in two parts and analyze path length anomalies ascribable to TTL rewriting that emerged from different kinds of probes; for the sake of completeness, we also

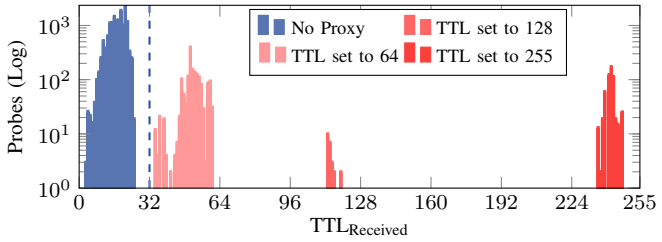


Fig. 2: Proxy detection through TTL rewriting: TTL distribution of received probe packets sent with an initial TTL of 32.

explore the possibility of proxies modifying HTTP content or more generally TCP payload without actually splitting TCP connection.

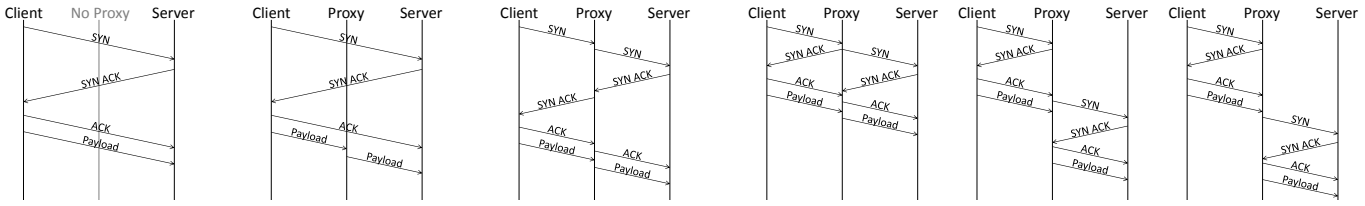
When a SYN sent by the client reaches the proxy it responds with a SYN ACK either immediately or upon SYN ACK reception from the intended destination (to which it has forwarded its own SYN). SYN generated by the proxy is expected to carry a default TTL value (e.g., 64) potentially causing an inconsistency in the path length, measured as the number of hops necessary to reach the responding node (pseudo-traceroute probes) or as the difference between sent packet and received packet TTL (server-based probes). Analyzing path length of pseudo-traceroute probes a considerable amount of anomalous values, especially ranging from 1 to 5 hops, emerged: although the invariance of the path length when targeting different destinations can further confirm the presence a proxy, it's not easy to rigorously draw a dividing line between paths with and without a proxy solely based on the path length. Server-based probes classification can instead be more accurate. Default TTL values used by Mobile Tracebox are 32 and 64: since 64 is a typical TTL value also used by proxies, only probes using TTL of 32 are suitable for proxy detection. In Sec. IV, we will discover how we can extend proxy revelation also to probes using a TTL of 64 (i.e., without relying on TTL alterations). Fig. 2 displays the TTL distribution of received SYN probe packets sent with an initial TTL of 32. The bell curve of the values compatible with the initial value is clearly visible in the region on the left of the dashed line, ranging from 3 to 25, but there are also three replicas above 32 that are instead compatible with a proxy along the path rewriting TTL respectively to 64, 128 and 255. The only TTL values outside of these 4 areas (158-160) were recorded from devices with a US sim card roaming in Europe: since roaming can indeed add further variability to measurements, roaming probes have been isolated in the data cleansing phase, and, thus, those values are not reported in the figure. The plot also shows that 64 is the most common default TTL among proxies analyzed in our dataset, confirming why probes with initial TTL of 64 are not eligible for this classification. The other probes collected can help to further validate our methodology: (*i*) path length distribution of pseudo-traceroute probes targeting our server suggests that portion of paths with a length >32 is negligible; (*ii*) probe packets sent with initial TTL of 64 were received with a minimum TTL of 34, sign that no proxy using a default TTL of 30-32 was observed.

Proxy scenarios. The previous classifications can be easily extended to probes using payload packets instead of SYN: this is crucial to check if payload modifications (especially HTTP) are always performed by TCP-terminating proxies or, in absence of a TTL alteration, are ascribable to packet-rewriting proxies [10]. It is also necessary to highlight a class of proxies that don't alter TTL on SYN but rewrite it on following payload packets: TCP handshake takes place as the proxy was not present, while subsequent packets are actually reforged by the proxy with its default TTL (Fig. 3b). Results of the non-responding server test instead can be used to further discriminate whether a proxy altering SYN responds with a SYN ACK immediately (Fig. 3d) or after SYN ACK reception from the destination server (Fig. 3c). We will further delve in the early SYN ACK scenario in the following section.

IV. Results

Proxy prevalence. We now use the classification described in the previous section to assess the prevalence of proxies across mobile networks. Fig. 4 displays the percentage of networks exhibiting a proxy on HTTP port as well as on the other ports tested by Mobile Tracebox. Our results confirm how HTTP proxies are widely deployed in mobile Internet and also how their detection is not necessarily constant throughout all the measurements recorded from a single network, especially if cellular. This phenomenon is not limited to proxies: data collected shows that also the presence and properties of other middleboxes (Carrier-Grade NATs, middleboxes adding or bleaching TCP Options, remarking IP DSCP, etc) can vary within the same carrier. We also tried to understand if the cellular technology used by the device played a role with regard to proxy presence: only 6 networks actually showed a significant difference in percentages (>20%) when probed with distinct technologies. Our findings also show that proxies are not limited to HTTP but can involve several TCP services and sometimes all TCP traffic. Although the presence of only an HTTP proxy is still the most common scenario, we also found networks in which only another specific service (e.g. SIP or SMTP) is proxied, or HTTP is proxied along with other services (e.g. HTTP, HTTPs and FTP). We also observed networks where all ports tested, including non-standard port (10000), exhibited a proxy. With regard to the last case we further tested 3 networks on up to 100 non-standard ports and all of them revealed a proxy, leading us to infer that all TCP traffic was actually routed through a proxy.

Packet-rewriting proxies. Our dataset also reveals a number of HTTP request manipulations operated by proxies (HTTP Header injections and modifications): almost all these alterations are accompanied by TTL anomalies, sign that they are performed by TCP-terminating proxies and not by packet-rewriting proxies. We identified one carrier and two Wi-Fi in which HTTP alterations occurred without showing an anomalous TTL. For the carrier we collected also a rooted tracebox probe [18], that confirmed the Header injection performed by a node along the path: that middlebox was not reforging the packet (neither TTL nor other fields were altered) but just



(a) No proxy (b) Proxy not altering SYN (c) Proxy altering SYN (without early SYN ACK) (d-1) SYN not deferred (d-2) SYN deferred till ACK (d-3) SYN deferred till Payload

(d) Proxy with early SYN ACK

Fig. 3: Observed scenarios.

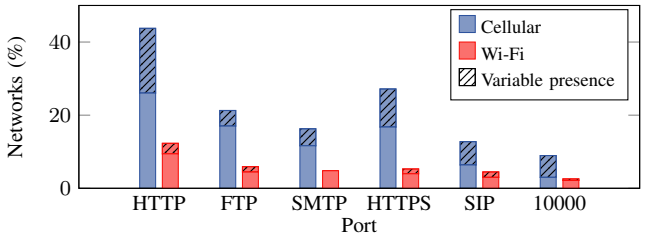


Fig. 4: Proxy detection on different TCP Ports.

TABLE II: Proxy implementations statistics.

Scenario	Fig.	Cellular	Wi-Fi
SYN not altered	3b	14%	15%
SYN altered, no early SYN ACK	3c	4%	8%
SYN altered, early SYN ACK	3d	82%	77%

modifying the payload of the packet as it traversed it, without splitting the connection.

TCP-terminating proxies. We now evaluate the extent of the different proxy scenarios depicted in Fig. 3. Table II shows the proportion of the implementations observed, with the proxy with early SYN ACK being the most common typology in both categories of mobile networks.

Deferred Handshake. Early SYN ACK decouples the handshake between client and proxy from the handshake between proxy and server: Fig. 3d-1 shows a proxy that concomitantly responds with a SYN ACK to the client and forwards its SYN to the original destination but it is also possible that the proxy delays the transmission of its SYN. To discern this aspect we set up a specific test that operates in 3 phases: (*i*) the client sends a SYN to the server; (*ii*) the client sends a SYN and the following ACK; (*iii*) the client performs handshake and sends a payload packet. At the end of each phase the client interrogates the server to check if a SYN has been received from the previously verified server-reflexive address of the client. This test reveals if the proxy retains SYN waiting for client's ACK or payload packet prior to contacting the destination. Although it is required to gain root privileges to expose this subtle dynamic, this probe does not rely on synchronization between client and server [12], that is a requisite harder to be achieved in a crowdsourced measurement. Xu et al. already exposed with their experimental testbed HTTP proxies in US carriers that defer handshake till reception of the actual HTTP request (Fig. 3d-3) [12]. With this test we also unveiled an additional deferred connection scenario in which proxy's handshake with the server is deferred till the completion of the handshake with the client (Fig. 3d-2). Although implications on latency must be taken into account in both cases, scenario in Fig. 3d-2 is less harmful when a client establishes a connection in advance and not immediately before payload is ready to be transmitted.

Initial Receive Window. We now delve into specific aspects of proxies configuration, to understand their impact on performance. We start from the Receive Window advertised on the first payload packet sent by the client after the handshake: Fig. 5 displays the CDF of the initial Receive Window as set by Android devices and by proxies. A preliminary survey on several Android devices revealed that this parameter, as well as the Window Scale factor, can vary depending on the network interface used (cellular or Wi-Fi) and therefore those values are plotted separately. We can easily acknowledge how the most recurring values for proxies are in range between 14K and 30K while for devices are around 64K and above: and in fact in 55% of the probes the Window Size advertised by the proxy is lower than the one originally advertised by the device: this can slow down initial data transmission by the server compared to the non-split connection. The Initial Receive Window limits the number of bytes that can be sent by the counterpart after the first request without waiting for an ACK. RFC6928 [22] recommended in 2013 to increase the initial Congestion Window (IW) to 10 segments. In 2017 Ruth et al. [23] observed how 85% of HTTP servers and 80% of TLS servers from Alexa top 1M list already supported this recommendation. In order to realize the full benefit of the large IW on server side, implementations on client side need to advertise an initial Receive Window of at least 10 segments. To understand this aspect we scaled the values in Fig. 5 by the MSS advertised on the SYN preceding the payload packet and compared devices and proxies settings to the current specification of 10 segments in Table III: the percentage of proxies that cannot benefit from servers supporting IW10 is higher than the the percentage of devices.

TCP Options. Mobile Tracebox can detect if TCP Options tested (see Table I) are actually carried by the SYN that reaches the controlled server but also if they are carried by the SYN ACK received from the responding host, and thus, in presence of a proxy, can test if the proxy supports those Options on both connections. Fig. 6 reports the percentage of proxies adopting the tested Options. Although the overall support of SP, TS and WS options is always above 80%, from our analysis it emerged that some cellular proxies support an Option only

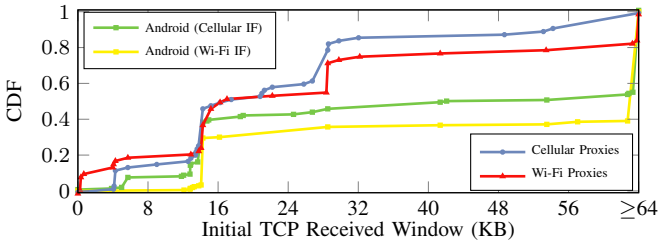


Fig. 5: Initial TCP Window as set by Android devices (using cellular and Wi-Fi interfaces) and by proxies.

TABLE III: Initial TCP Receive Window (in number of segments) as set by Android devices and by proxies.

Window (# of segments)	Cellular Device	Cellular Proxy	Wi-Fi Device	Wi-Fi Proxy
<10	7%	15%	0%	20%
=10	30%	31%	29%	18%
>10	63%	54%	71%	62%

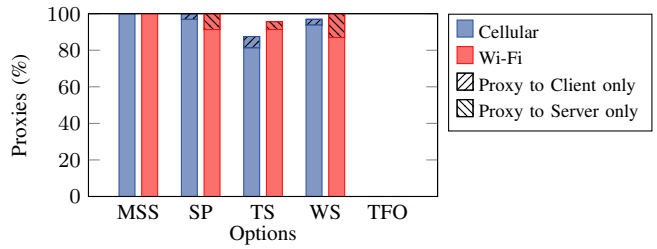


Fig. 6: TCP Options adoption by proxies.

on the connection between proxy and client and not on the connection between proxy and server; the symmetric behavior has been observed for Wi-Fi proxies. Since those options are designed for large Bandwidth-Delay Product (BDP) networks, a possible explanation of these settings is that in cellular networks the portion of the path with higher delay is presumably from the client to the proxy while in Wi-Fi is from the proxy to server. No proxy was traversed by the TFO Option: this shows that analyzed proxies don't support the TFO implementation tested (i.e., using experimental kind, 254) but also suggests that the set of Options on proxy's SYN is fixed and do not depend on client's SYN: where available, rooted probes using SYN with MPTCP Option, an unassigned Option kind or without any Options corroborate this hypothesis.

Window Scale factor. The Window Scale factor determines the maximum Receive Window that can be advertised, which is crucial, especially on paths with a high round-trip time, since the throughput is limited by the ratio *RWIN/RTT*. Fig. 7 displays the CDF of the values set by Android devices and by proxies (proxies that don't support WS on the connection with the server are excluded). In this case the distributions are very close and in fact only in 40% of the recorded probes the proxy sets a WS value lower than the original. The plot reports the WS advertised by proxy's SYN: we also observed that 22% of proxies advertised different WS factors on the connection to the client and to the server.

TCP ECN. TCP ECN is not enabled by default on Android

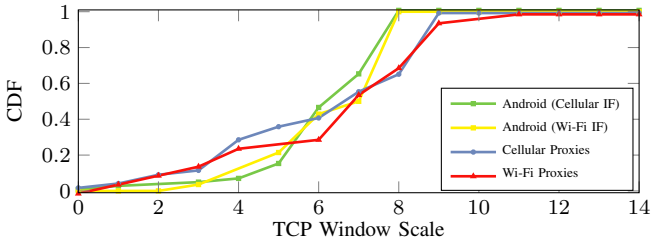


Fig. 7: TCP Window Scale as set by Android devices (using cellular and Wi-Fi interfaces) and by proxies.

and, thus, it could not be tested on client-to-proxy connection. However to investigate ECN deployment across proxies we inspected SYN packets sent by all the proxies tested, checking if the flags used to negotiate ECN (CWR and ECE) were set: we identified only one UK carrier proxy supporting ECN by default on outgoing connections.

Proxy location. We now investigate proxy location inside the network. As detailed in Sec. III a single traceroute probe is not enough to assert the presence of a proxy: for this reason we consider only probes in pseudo-traceroute mode preceded by a server-based probe that has highlighted the presence of a proxy altering SYN. Fig. 8 shows how cellular proxies are located at up to six hops away from the source while Wi-Fi proxies are generally closer, with the majority of them located at the first node.

Fingerprinting. Finally, we attempt to fingerprint proxies in terms of packet modifications drawing a dividing line with other classes of middleboxes. Fig. 9 compares modifications detected on paths with a proxy and without. We can easily acknowledge that modifications on IP Don't Fragment, TCP Window, Window Scale, and to a lesser extent on TCP Offset (and consequently on IP Total Length) are almost exclusively ascribable to proxies. These findings help us to discriminate what happens in the proxy scenario depicted in Fig. 3b, in which SYN did not exhibit TTL rewriting while following payload packets did. For a subset of these proxies, SYN packet had neither TTL alteration nor other fingerprinting modifications leading us to infer that the original SYN flowed till the destination host. For another subset of proxies, modifications on IP Don't Fragment and TCP Window were detected on SYN (even in absence of a TTL alteration) other than on payload packets, so it is more likely that the proxy actually forged its own SYN although keeping the original TTL. The previous considerations can also be of help to infer proxy presence in server-based probes where TTL of 64 was used and more generally when we cannot rely on TTL alterations (e.g., TTL cannot be manipulated).

Measurement caveats. A few caveats emerged from the this section and Sec. III: (*i*) detection methodologies based on a non-responding test and more generally on a SYN only test are not capable to detect all TCP-terminating proxies, due to the presence of proxies not employing early SYN ACK and also not altering SYN; (*ii*) detection methodologies based on fingerprinting packets received on a specific port (e.g. HTTP)

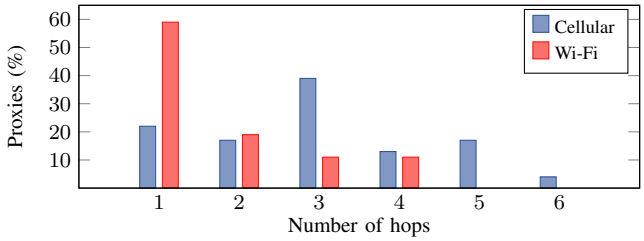


Fig. 8: Proxy location in mobile networks.

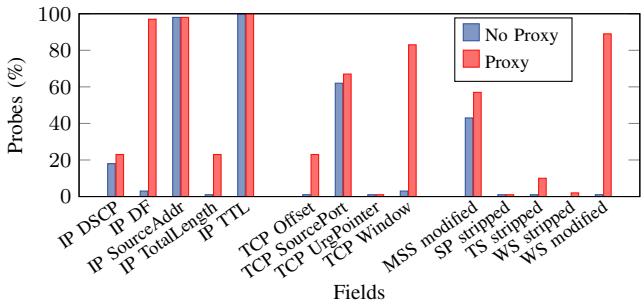


Fig. 9: Fingerprinting proxies: comparison of packet modifications detected on paths with a proxy and without.

and on a non-standard port do not succeed in networks where all TCP traffic is routed through a proxy; (*iii*) a measured path length up to 6 hops does not rule out the presence of a proxy; (*iv*) testing if a certain feature (e.g. TCP Options, ECN) is supported by a proxy on client-to-proxy or proxy-to-server connection does not imply that is supported or not on the symmetric connection.

V. Conclusions and Future Work

Detection and characterization of proxies in mobile networks can be complex due to two factors: their multifaceted interference and the innate limitations of mobile devices as network measurement tools. We addressed these issues with Mobile Tracebox that embodies diverse methodologies to detect proxies as well as other classes of middleboxes. Instead of relying only on application layer modifications as several previous works, we mainly, but not exclusively, grounded on TTL alterations ascribable to proxies to reveal their presence. Analyzing data collected by means of crowdsourcing we showed their prevalence and how their range is not limited to HTTP but can include other services and even all TCP traffic. We described the transport layer behavior of the proxies observed and detailed several aspects of their configuration (from the initial Receive Window to the range of TCP extensions supported) to understand their impact on performance. We plan to implement a new probing scheme to test further aspects and continue our crowdsourced study to understand the trend of proxies in mobile Internet.

Acknowledgments

The research described in this paper has been partially funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 688421. The opinions expressed and arguments employed reflect only the authors' views. The European Commission is not responsible for any use that may be made of that information. The work of Antonio Pescapé has been partially supported by the art. 11 DM 593/2000 for NM2 srl.

References

[1] B. E. Carpenter, "Architectural principles of the Internet," Internet Engineering Task Force, RFC 1958, June 1996.

[2] J. Kempf and R. Austein, "The rise of the middle and the future of end-to-end: Reflections on the evolution of the Internet architecture," Internet Engineering Task Force, RFC 3724, March 2004.

[3] A. Botta and A. Pescapé, "Monitoring and measuring wireless network performance in the presence of middleboxes," in *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services*, Jan 2011, pp. 146–149.

[4] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM*, August 2012.

[5] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradation," Internet Engineering Task Force, RFC 3135, June 2001.

[6] S. McQuistin and C. Perkins, "Reinterpreting the transport protocol stack to embrace ossification," in *Proc. IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)*, January 2015.

[7] Y. Shavitt and E. Shir, "DIMES: Let the internet measure itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 71–74, October 2005, see http://www.netdimes.org.

[8] F. Fuchs-Kittowski and D. Faust, "Architecture of mobile crowdsourcing systems," in *Proc. International Conference on Collaboration and Technology (CRIWG)*, September 2014.

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

[10] N. Weaver, C. Kreibich, M. Dam, and V. Paxson, "Here be web proxies," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2014.

[11] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, N. Weaver, and V. Paxson, "Beyond the radio: Illuminating the higher layers of mobile networks," in *Proc. International Conference on Mobile Systems, Applications, and Services*, May 2015.

[12] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan, "Investigating transparent web proxies in cellular networks," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2015.

[13] J. Hui, K. Lau, A. Jain, A. Terzis, and J. Smith, "How youtube performance is improved in t-mobile network," in *Proc. Velocity*, June 2014.

[14] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2011.

[15] U. Goel, M. Steiner, M. P. Wittie, M. Flack, and S. Ludin, "Detecting cellular middleboxes using passive measurement techniques," in *Proc. Passive and Active Measurement Conference (PAM)*, March 2016.

[16] R. Zullo, "Mobile tracebox," 2016. [Online]. Available: http://play.google.com/store/apps/details?id=be.ac.ulg.mobiletracebox

[17] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet, "Hic sunt NATs: Uncovering address translation with a smart traceroute," in *Proc. IEEE/IFIP Workshop on Mobile Network Measurement (MNM)*, June 2017.

[18] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.

[19] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, "Netalyzr: Illuminating the edge network," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

[20] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, and V. Paxson, "Header enrichment or isp enrichment?: Emerging privacy threats in mobile networks," in *Proceedings of the 2015 ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2015, pp. 25–30.

[21] S. Huang, F. Cuadrado, and S. Uhlig, "Middleboxes in the Internet: a HTTP perspective," in *Proc. Network Traffic Measurement and Analysis Conference (TMA)*, June 2017.

[22] H. J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Rfc6928-increasing tcp's initial window," 2013.

[23] J. Rüth, C. Bormann, and O. Hohlfeld, "Large-scale scanning of tcp's initial window," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 304–310.