MIRAGE:
Mobile-app Traffic Capture and Ground-truth Creation

Giuseppe Aceto[1,2], Domenico Ciuonzo[1], Antonio Montieri[1], Valerio Persico[1], Antonio Pescapé[1,2]
[1]University of Napoli "Federico II" (Italy) and [2]NM2 s.r.l. (Italy)
{giuseppe.aceto, domenico.ciuonzo, antonio.montieri, valerio.persico, pescape}@unina.it

***Abstract*—Network traffic analysis, i.e. the umbrella of procedures for distilling information from network traffic, represents the enabler for highly-valuable profiling information, other than being the workhorse for several key network management tasks. While it is currently being revolutionized in its nature by the rising share of traffic generated by mobile and hand-held devices, existing design solutions are mainly evaluated on private traffic traces, and only a few public datasets are available, thus clearly limiting repeatability and further advances on the topic. To this end, this paper introduces and describes MIRAGE, a reproducible architecture for mobile-app traffic capture and ground-truth creation. The outcome of this system is MIRAGE-2019, a human-generated dataset for mobile traffic analysis (with associated ground-truth) having the goal of advancing the state-of-the-art in mobile app traffic analysis. A first statistical characterization of the mobile-app traffic in the dataset is provided in this paper. Still, MIRAGE is expected to be capitalized by the networking community for different tasks related to mobile traffic analysis.**

***Index Terms*—Android apps; encrypted traffic; mobile apps; mobile traffic; reproducible research; open dataset; traffic classification;**

I. INTRODUCTION

As in almost all experimental-research fields, *replicability* and *reproducibility* are critical concerns in achieving significant and grounded progress [1]. Accordingly, a strong push for them is currently observed, with research artefacts being also evaluated in the standard peer-review process [2]. In fact, provisioning datasets as well as carefully documenting workflows for obtaining them, are critical to foster replicability and reproducibility, respectively, with both fueling research dissemination. Indeed, by leveraging wisely sourced and constructed datasets that are available to the research community, research outcomes become (i) *easier to reproduce*, (ii) *comparable* against other studies, and (iii) *generalizable* to other data/systems not studied yet.

These possibilities have been sorely missing for a long time [3] in the field of network traffic analysis (i.e., distilling information to perform key management and profiling tasks, such as traffic classification, modeling, and prediction). In spite of this huge interest, the availability of data for performing research within this field has remained quite limited. Many appealing research solutions have been (and are) mostly validated on private datasets, thus precluding repeatability and safe advances on the topic. Still, it is shared opinion that this aspect contributes to slow down and limit the understanding of the tackled problems, since it constitutes a severe drawback in both design and experimental validation phases.

Even worse, in recent years new challenges have arisen. Smartphones have become the main medium of communication, providing means for connecting groups of users as well as users to services. According to Ericsson mobility report [4], the number of smartphone subscriptions is expected to reach 7.2 billions by 2024, with a corresponding $+30\%$ predicted compound annual growth rate of traffic generated by mobile networks. Due to the users massive shift toward mobile devices and the mobile applications (in short, *apps*) running on them, overall network traffic reached huge volumes and started evolving at an unprecedented pace. Such rate of change is further increased for mobile apps due to the software distribution systems (the *apps marketplaces*), that have fostered one-click installation and quick-paced automatic updates. The encryption mechanisms being increasingly adopted further impair the analysis, while unprecedented privacy concerns limit the collection and publishing of raw mobile traffic traces. These challenges call even more for the availability of *mobile-traffic datasets*, considering that even when they are made available to the community, the presence of bot-generated traffic or experiments run in controlled environments limit the validity of the proposed analysis or design solutions.

The contribution of this paper is *fourfold*: (i) we survey the encrypted-traffic datasets publicly available, highlighting their main characteristics as well as main shortcomings and limitations; (ii) we present and describe our reproducible MIRAGE architecture for generating new mobile-app traffic datasets and automatically creating the related high accurate ground-truth. The focus is on apps running on Android, currently retaining the 76% of the whole market share[1]; (iii) we release the MIRAGE-2019 dataset[2], thus fostering the replicability of the analysis and its extension to multiple use cases; (iv) we provide a characterization for MIRAGE-2019, according to metrics of interest at three different granularities (flows, packets, and metadata), thus proving its suitability for a wide range of tasks.

The rest of the paper is organized as follows: Sec. II reviews current publicly-available datasets for encrypted (and possibly mobile) traffic analysis; Sec. III describes the whole MIRAGE architecture involved (including capture and ground-truth building phases), with Sec. IV describing the release format of the MIRAGE-2019 dataset. Then, Sec. V presents example tasks enabled by MIRAGE-2019, by means of mobile

[1]http://gs.statcounter.com/os-market-share/mobile/worldwide
[2]MIRAGE-2019 is publicly released at http://traffic.comics.unina.it/mirage.

978-1-7281-0875-9/19/$31.00 ©2019 IEEE.

Table I: Summary of previous datasets on encrypted traffic analysis.

Ref	Dataset	Capture Span	📱	🧍	TO	Raw	Pkt	TO stats	Meta	Task	Diversity	R
[5]	ISCXVPN2016	Mar. '15 - Jun. '15	○	●	B	✔		✔		TI&TC	7 TTs	◐
[6]	ISCXTor2016	Lug. '15 - Feb. '16	○	●	B	✔		✔		TI&TC	8 TTs / 18 apps	◐
[7]	Anon17	'14 - '17	○	●	F		✔	✔	✔	TI&TC	3 ATs / 8 TTs / 21 apps	◐
[8]	QUIC	Mar. '18	○	○	B		✔			TI&TC	5 QUIC services	◐
[9]	MTD	Oct. '16 - Mar. '17	●	●	B			✔	✔	TI&TC	12 Apps / 10 DEVs / 10 EXPs	◐
[10]	UNSWIoT	Oct. '16 - Apr. '17	◐	●	B	✔				DevID	28 DEVs	●
[11]	NTD Reddit	Apr. '18 - May '18	○	○	W	✔				WebAN	5 BWs	◐
[12]	Video Streams	Aug. '15 - May '16	○	○	F	✔				VidID	2.1k VTLs	●
[13]	Youtube Video	Sep. '17 - Feb. '18	●	○	P		✔		✔	VT-QoE	3 VTLs / 374 h	●
[14]	Netflix UE	Oct. '18 - Feb. '19	○	○	F			✔		VT-QoE	10 LOCs / 2.6k VTLs	◐
Ours	MIRAGE-2019	May '17 - May '19	●	●	B		✔	✔	✔	TI&TC	40 apps / 3 DEVs / 280+ EXPs	●

Traffic Nature: 📱 = Mobile, 🧍 = Human-generated. **Traffic Object (TO)**: `BF` = Biflow, `F` = Flow, `P` = Packet, `W` = Webpage. **Released Data**: `Raw` = PCAP files, `Pkt` = Packet-level data, `TO stats` = TO statistics, `Meta` = Metadata. **Task**: `DevID` = Device Identification, `TI&TC` = Traffic Identification and Classification, `VidID` = Video Identification, `VT-QoE` = Quality-of-Experience in Video Traffic, `WebAN` = Website Analysis. **Diversity**: `AT` = Anonymity Tool, `BW` = Browser, `DEV` = Device, `EXP` = Experimenter, `LOC` = Location, `TT` = Traffic Type, `VTL` = Video Title. **Reproducibility (R)**.

traffic characterization & modeling attempts, whereas Sec. VI discusses the ethical considerations concerning the acquisition phase. Finally, Sec. VII provides concluding remarks.

II. Related Work

This section reviews the most related public traffic datasets and categorizes them according to the taxonomy defined in Tab. I. Only datasets collecting *encrypted network traffic* have been considered. Other datasets e.g., related to network-anomaly detection or attack classification have not been taken into account, due to their peculiar focus on network security. In detail, we categorize each dataset based on whether (a) it focuses on the mobile scenario, (b) it is generated by real human experimenters (as opposed to bots or scripts) and (c) the description of the capture system employed for generating the traffic makes it partially/completely reproducible. As a complementary information, we also provide the capture span for each dataset, either explicitly specified in the corresponding paper or obtained by direct inspection of the artifacts. Additionally, we surface (i) the traffic object considered (i.e. the traffic segmentation criterion used), (ii) the type(s) of released data, and (iii) the diversity of the collection space. Specifically, with respect to point (ii), we classify the form of released data in **Raw** (PCAP files are available), **Pkt** (fields from each packet are available), **TO stats** (i.e. summarizing statistics for each traffic object are available) and **meta** (complementary metadata are available). Differently, referring to point (iii), we provide the number of different services/applications or types of objects considered. Finally, an explicit mention to the main intended task approached is provided.

First, referring to the **capture span**, we observe that all the datasets considered have been collected in the last five years (2014–19), reflecting the only recent trend toward the growing adoption of encrypted protocols (e.g., TLS) [15]. Further, we observe that the experimental campaigns last from months to years, with longer ones typically associated with human interaction (see later discussion). A similar rationale applies to MIRAGE-2019, collected in the last two years and expected to reflect better the current nature of mobile traffic.

Differently, focusing on network traffic generated exclusively by **mobile** (handheld) devices, it is apparent that only the datasets MTD [9] and MIRAGE-2019 (ours) have captured this type of traffic. The only exceptions are represented by UNSWIoT [10] (in which some background traffic is generated from mobile devices) and Youtube Video [13] (where streaming video was analyzed on smartphones).

As anticipated, not all the considered datasets have been generated by **human users**. Indeed, we point out that the above feature may be crucial when analyzing traffic generated by complex interaction patterns from the users, as in the case of anonymity tools [5, 7] and mobile applications (object of this work) [9] with automated tools non reflecting completely the above complex behaviour. For example, [8, 11, 12, 14] have employed Selenium[3] for automating web browsing.

Referring to **traffic object** segmentation, most of the works consider either flows [7] or biflows as the relevant traffic analysis unit, with the sole exception of [11] using webpages as the relevant object of analysis.

Referring to the main **task** approached, most of the datasets have been collected with the aim of performing and evaluating traffic identification and classification, with specific focus on anonymity tools (to assess their degree of anonymity) [6, 7], specific traffic services (e.g. Google QUIC protocol) [8], Virtual Private Networks (VPNs) [5] or mobile apps classification [9]. Differently, other datasets are specifically focusing on (encrypted) video traffic analysis, with either considering title fingerprinting [12] or Quality-of-Experience (QoE) prediction [13, 14]. Finally, some works focus on identifying specific devices generating traffic, e.g. IoT devices [10], and other delve into website analysis [11].

Referring to **diversity**, datasets provide information with different degrees of variety, also according to the goal and the scope of the work they are proposed with. Overall,

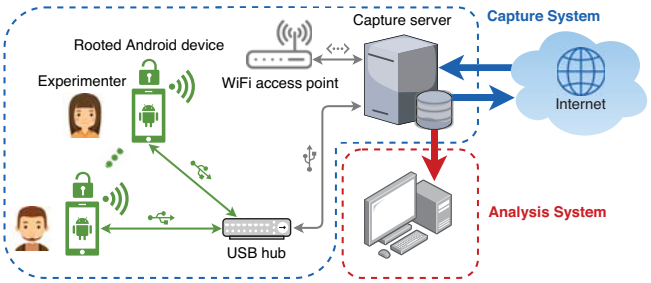
[3]https://www.seleniumhq.org

different applications/services/contents are considered, as well as multiple experimenters (in case of human-generated traffic), capture devices, and locations. For instance, ISCXTor2016 [6] contains *8* different traffic traffic types (browsing, audio, etc.) corresponding to *18* applications which are run under *two* different scenarios (one to detect Tor traffic flows and the other to detect the application type). Differently, Anon17 [7] contains info about the traffic types and applications running on Tor, I2P and JonDonym is provided in the form of *3-level labels* for each flow. However, each dataset focuses on a limited set of the mentioned aspects in line with the nature of the problem to investigate. In fact, none of those surveyed covers all these aspects at best. At most, 21 applications [7], 28 devices [10], and 10 experimenters [9] are considered. MIRAGE-2019 takes into consideration traffic generated by 280+ experimenters using 40 applications via 3 devices.

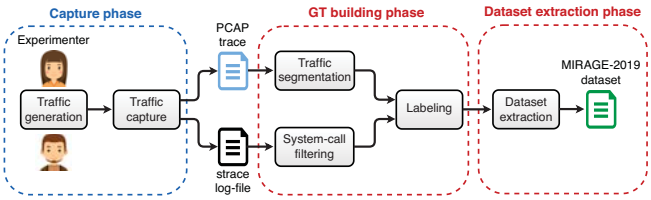
Finally, referring to **reproducibility**, it is apparent that not in all the cases the details, the setup and the procedures required to reproduce the same experimental setup for the capture have been reported. Nonetheless, in some cases, a detailed description of the whole experimental setup has been provided, see e.g. [10, 12, 13].

III. The MIRAGE Architecture

At a high level, the MIRAGE system architecture consists of two main components: the **Capture System** and the **Analysis System**. Figure 1(a) shows the architecture of the Capture System. The *capture server* is a workstation equipped with an IEEE 802.11g *access point*, which provides connectivity to the *mobile devices* that generate the traffic when human *experimenters* operate the apps. A wired connection brings the access of the capture server to the public Internet, performing Network Address Translation (NAT). Notably, the upstream connections to the public Internet do not constitute a bottleneck in terms of bandwidth, thus not impacting the properties of the traffic stream flowing through the access WLAN. Each mobile device is also physically attached to the capture server through the *USB hub*; this allows leveraging the Android Debug Bridge (ADB) to send commands from the capture server to the attached devices and receive responses on an off-band channel. Such procedure requires the devices to be *rooted* in order to successfully run the traffic capture. Notably, our architecture is able to handle traffic capture of multiple devices simultaneously. The captured traffic is then processed by the Analysis System (in our prototype, hosted on the capture server itself) to produce MIRAGE-2019.



(a) MIRAGE architecture.



(b) MIRAGE-2019 dataset building.

Figure 1: The MIRAGE architecture: (a) diagram of main components and (b) workflow of dataset building.

In detail, the MIRAGE architecture builds the dataset in three phases as shown in Fig. 1(b): (1) *Capture phase*: traffic traces are collected in PCAP format together with `strace` log–files keeping track of network system-calls. (2) *GT building phase*: PCAP traces are segmented and log-files are filtered. this information is then combined to produce labeled traffic objects. (3) *Dataset extraction phase:* labeled traffic objects are processed to extract the information of interest (including statistical features) to be used as input data for exploratory analysis or machine learning tasks. The next subsections provide details about these three phases.

A. Capture phase

A capture session begins when an experimenter connects a mobile device to the USB hub: this procedure automatically kicks off the capture of the network traffic as well as the logging of the system calls. In detail, the traffic is captured on the wired interface of the capture server by means of `tcpdump`[4] Leveraging traffic filters based on the MAC address of the connected devices allows to collect the traffic generated by multiple devices at the same time, without any ambiguity. On the other hand, the system-call tracing is enabled on the mobile device itself by using the `strace` utility via ADB[5] `strace` is set to log network-related and process-management system calls (e.g., `connect`, `bind`, `getsockname`, `fork`, `wait`, `exec`, etc.), also logging the related `<IP:port>` pairs and associating each socket descriptor to the name of the Android package which originates the call.[6] As a result, this phase provides an `strace` log-file with ground-truth information for each PCAP trace collected (see Fig. 1b). This kernel-based mechanism for gathering information being given, the finest granularity we can achieve is the (bi)flow level.

The described capturing system allows the capture of mobile-app traffic when accessing the Internet through a Wi-Fi channel. In principle, the behavior of the apps could be different from the one shown when connecting through 3G/4G channel. Indeed, Android applications may detect the type of network that is used to transport their data, telling apart Wi-Fi, Mobile, and VPN connections and potentially acting

[4]http://www.tcpdump.org/.

[5]https://developer.android.com/studio/command-line/adb.html

[6]To this aim, we monitor the Android *zygote process* that handles the forking of each new application process. Then we extract the Android package names from the PIDs returned by the `fork` system calls.

in different ways according to that.[7] Other mobile-app traffic capture approaches leverage an encrypted connection (VPN) that tunnels the traffic over 3G/4G to a gateway/capture server [9]. Compared to our method, the VPN-based ones not only suffer from the identical issue (apps are able to know if the transport network is a VPN the same way), but possibly add *uncontrolled changes* to the statistical properties of captured traffic due to the underlying tunneling protocol mechanics and the network between the mobile device and the capture gateway, resulting in a traffic timing less accurate and possibly traces less representative of plain 3G/4G setups.

B. GT building phase

In the GT building phase, first the PCAP traces are segmented to obtain *traffic objects*. Then, each of these objects is labeled taking advantage of the information extracted from the associated `strace` log-file.

PCAP traces are segmented into *biflows*, that is we group together all the packets having the same 5-tuple *(source IP and port, destination IP and port, and transport-level protocol)*, with source and destination addresses and ports that can be swapped. As opposed to common heuristics leveraged to define the 5-tuple that identifies each biflow (i.e. first packet captured, TCP SYN, etc.) [3], we are able to order the items consituting the 5-tuple based on the knowledge of IP addresses of the Android devices used for the captures. Therefore, in the dataset the addresses in biflows are ordered in the upstream direction (i.e. local-to-remote, with reference to the mobile terminal). Note that biflow segmentation is a common choice for traffic objects [3, 16] and perfectly fits the metadata extracted from the network (viz. socket) system-calls.

Then, each biflow is labeled with the Android package-name that *exactly* matches the 5-tuple in the `strace` log-file, considering `getsockname` and `connect` system calls. In the case a perfect match cannot be found in the log-file for some biflows[8], the procedure assigns labels according to a heuristic, i.e. labeling these biflows with the *most-common label* (i.e. package name) in the PCAP trace. As this approach can potentially cause mislabeling, the case of heuristic labeling is explicitly marked as such in the dataset, allowing the final user to decide between considering in the GT all traffic objects (possibly noisy) or keep only strict matching ones.

C. Dataset extraction phase

This phase is in charge of taking each labeled traffic object and extract the relevant information to feed any potential application of the collected data (e.g., exploratory mobile-app traffic analysis or machine learning algorithms to solve specific tasks). The output of this phase constitutes the final MIRAGE-2019 dataset. Since the traffic objects here considered correspond to biflows, information can be drawn in the form of summarizing statistics from the whole traffic object, of from a subset of the constituting packets. The specific types of information provided for MIRAGE-2019 and the related context are described in full details in the following section.

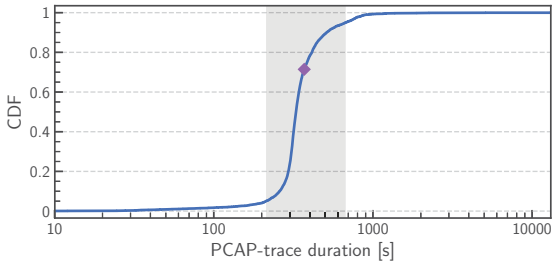


Figure 2: Cumulative distribution of the duration (s in log-scale) of the PCAP traces collected. The diamond marker reports the average, whereas the shaded box highlights the $(5, 95)^{th}$ percentile region.

IV. Dataset Description

We have collected the MIRAGE-2019 dataset in the ARCLAB laboratory at the University of Napoli "Federico II". The capture sessions (cf. Sec. III-A) span from May 2017 to May 2019. We employed three devices to generate the mobile traffic, namely: (i) Xiaomi Mi5, (ii) Google Nexus 7, and (iii) Samsung Galaxy A5. In detail, we installed the custom firmware CyanogenMod v13.0 (corresponding to the Android version 6.0.1) on all the devices and enabled the root mode.

More than 280 experimenters took part to the dataset construction on a voluntary basis, by performing one or two experimental sessions each. The experimenters involved in this activity were students of three different courses[9] held at the University of Napoli "Federico II", aged $19 \div 25$ years, with a $85/15\%$ share between males and females. Each experimental session lasted two hours, at most. Altogether, during each experimental session, each experimenter performed 12 capture sessions of $5 \div 10$ minutes (each resulting in one PCAP traffic trace and one `strace` log-file, cf. Sec. III). In each capture session the experimenter was asked to perform activities mimicking common uses of a single app with the intent to explore its functionalities in addition to first-time install, login, registration. We report the ethical considerations underlying the aforementioned traffic-capture procedure in Sec. VI.

Overall, the MIRAGE-2019 dataset gathers the traffic generated by 40 Android apps belonging to 16 different categories according to Google Play apps distribution portal [17]. Before each experimental session, the exercised app is updated to the latest version available on the Italian Play Store. The MIRAGE website (http://traffic.comics.unina.it/mirage) reports the detailed app metadata together with the links to their pages

[7] The `ConnectivityManager` method `getType()` returns, among the others, `TYPE_WIFI`, `TYPE_MOBILE`, `TYPE_VPN` according to the active connection type (see https://developer.android.com/reference/android/net/ConnectivityManager for API level before 21, or https://developer.android.com/reference/android/net/NetworkCapabilities for the analogous constants prefixed with `TRANSPORT_` in API level 21 and following).

[8] This could be caused by either the pre-existence of these biflows before the capture started or by a failure of the `strace` in following the corresponding child-processes forked by zygote.

[9] Namely: Computer Architectures, Computer Networks, and Internet Analysis and Performance.

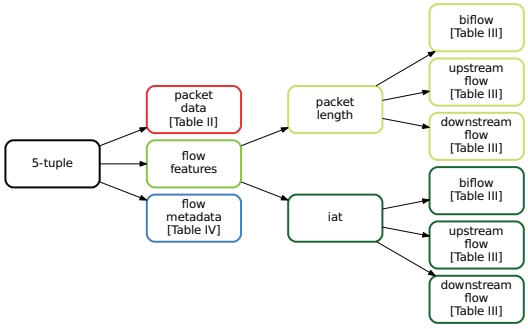


Figure 3: Structure of the JSON files constituting MIRAGE-2019.

on Google Play. As a whole, a total of 4606 PCAP traces were collected within MIRAGE-2019[10].

Figure 2 shows the cumulative distribution of the duration of the traces collected, having an average duration of 370 s. It can be noted that the majority of traces has a duration corresponding to that prescribed for capture sessions (i.e. $5 \div 10$ min.), being the median equal to 329 s and the 5- and 95-percentile equal to 213 s and 674 s, respectively. Differently, outliers are due to unintended disconnections from the traffic capture system, erroneous procedures carried out by the experimenters, but also specific experimental scenarios (e.g., prolonged video-playing, calls, etc.).

We release the MIRAGE-2019 dataset in JSON format to foster its compatibility and increase its usability: one JSON file corresponds to one PCAP trace captured (i.e. a self-contained capture session). In detail, for each biflow—identified by its 5-tuple—we have extracted: (i) *per-packet data*, (ii) *per-flow features*, and (iii) *per-flow metadata*. Figure 3 shows the structure of each JSON file. The successive paragraphs provide details about released data and their format.

Per-packet data: We extract 6 informative header fields and the L4 payload of the first 32 packets of each biflow. Table IIa describes the data D_i extracted. In detail, each D_i identifies a list of up to 32 elements. Researchers performing traffic classification [18, 19] and website fingerprinting [20] via machine and deep learning used these inputs in their works.

Per-flow features: To provide information on the whole biflow and corresponding upstream and downstream flows, we select 17 statistical features computed on the sets of upstream, downstream, and complete (i.e. both of them) IP packet lengths and inter-arrival times, for a total of 102 per-flow features. Table IIb reports the per-flow features F_i extracted. Previous works in the field of traffic classification via machine learning successfully leveraged these features to feed the classification algorithms they devised [16, 21–23].

Per-flow metadata: Table IIc describes per-flow metadata complementing per-flow features, being also related to complete biflow and upstream/downstream flows. GT (viz. the biflow-label extracted) granularity (i.e. exact or most-common) refers to the GT-building procedures described in Sec. III-B.

[10]We have filtered out the PCAP traces having an `strace` log-file less than 200 kB or a duration less than 10 seconds. We plan to continue the experimental activities and update the MIRAGE-2019 dataset over time.

Table II: The MIRAGE-2019 dataset. D_i, F_i, and M_i report the dict-keys in the released JSON files.

(a) Per-packet data extracted from the first 32 packets of each biflow.

D_i	Description
`src_port`	Source transport-layer port
`dst_port`	Destination transport-layer port
`packet_dir`	Packet direction (0 upstream, 1 downstream)
`L4_payload_bytes`	Number of bytes in L4 payload
`iat`	Inter-arrival time
`TCP_win_size`	TCP window size (0 for UDP packets)
`L4_raw_payload`	Byte-wise raw L4 payload (integer $\in [0, 255]$)

(b) Per-flow features extracted from the sets of upstream, downstream, and complete IP packet lengths and inter-arrival times.

F_i	Description
`min`	Minimum
`max`	Maximum
`mean`	Arithmetic mean
`std`	Standard deviation
`var`	Variance
`mad`	Mean absolute deviation
`skew`	Unbiased sample skewness
`kurtosis`	Unbiased Fisher kurtosis
`q_percentile`	q^{th} percentile ($q \in [10 : 10 : 90]$)

(c) Per-flow metadata M_i related to the complete biflow (`BF`) and upstream (`UF`) and downstream (`DF`) flows.

M_i	Description
`BF_label`	Android-package name
`BF_labeling_type`	Exact or most-common labeling
`{BF,UF,DF}_num_packets`	Number of packets
`{BF,UF,DF}_IP_packet_bytes`	Total bytes in IP packets
`{BF,UF,DF}_L4_payload_bytes`	Total bytes in L4 payloads
`{BF,UF,DF}_duration`	(Bi)flow duration in seconds

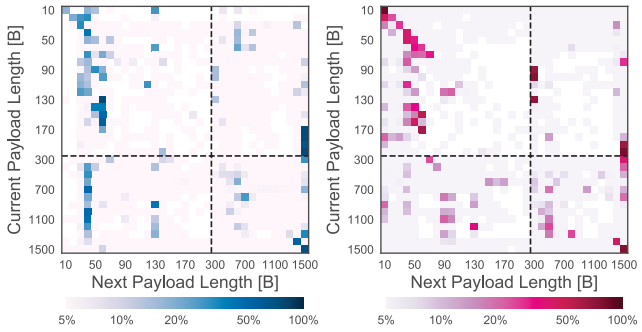
V. EXAMPLE TASKS ENABLED BY MIRAGE-2019

The dataset we release can suit different kinds of tasks, ranging from app traffic modeling and prediction, to biflow-based traffic classification. Herein, we present an app traffic characterization that can be directly derived from the dataset, concerning per-packet data (Sec. V-A), per-flow features (Sec. V-B), and per-flow metadata (Sec. V-C). While providing a detailed characterization of MIRAGE-2019, our analyses also point at tasks enabled by the released dataset.

A. Per-app modeling based on per-packet data

Modeling network traffic represents a key task in the study and design of Internet architectures, as realistic (yet manageable) traffic models are needed to predict, interpret, and solve performance-related issues of current and future networks.

To prove the suitability of MIRAGE-2019 to support this class of tasks, we provide a preliminary study aimed at devising *per-app* traffic models by means of Markov Models, similarly as done in [24]. In detail, for each app we consider the sequence of the `L4_payload_bytes` of each biflow (i.e.



(a) com.dropbox.android. (b) air.com.hypah.io.slither.

Figure 4: Transition matrices of payload lengths. Axes are divided in two different linear scales. The color-bar is in log-scale.

focusing on the first 32 packets, see Sec. IV) and derive the so-called *transition matrix*, whose $(X, Y)^{th}$ entry represents the probability that next packet comes with Y bytes of payload if the last observed packet has a payload of X bytes (save from rounding errors due to binning). Notably: (i) packets with null payload are filtered out (i.e. only packets transferring contents generated by the application layer are retained [22], while signaling such as TCP SYNs, RSTs, and *pure* ACKs are discarded, as not of interest for this analysis); (ii) the payload-size interval between 1 B and 1500 B is divided in 33 non-uniform bins, namely: bins from 1 to 20 (resp. from 21 to 33) are 10 B (resp. 100 B) wide. This scheme allows to better appreciate the model dynamics by mitigating the impact of the high presence of packets with null or very-small payload.

As an example, Fig. 4 reports the transition matrices for Dropbox (Fig. 4a) and Slither.io (Fig. 4b). By looking at the two matrices, the following observations can be derived: (i) the presence of high values (darker colors) along the diagonal witnesses the tendency to remain in the same state, i.e. sending/receiving consecutive packets of similar sizes. This observation holds for both apps, and becomes evident for very-small values (top-left corner) as well as for very-large values (bottom-right corner). (ii) Vertical patterns highlight the trend in entering to a specific state, whichever the current payload length. This is particularly evident for Fig. 4a (next expected payload length is within 30–40 B when current payload length lies within 300–1400 B) and in both apps when the current payload belongs to 1400–1500 B. (iii) Finally, scattered darker points highlight app-specific patterns.

B. Per-flow statistic characterization

Characterizing mobile-app traffic based on its statistical features is a capability of the utmost importance that mitigates a number of issues and benefits different tasks in network administration. For instance, trends in network applications and protocol design (e.g., protocol encapsulation, encrypted transmission, use of non-standard ports, concerns about users' privacy) heavily challenge *traffic classification* when exploiting some of the developed techniques (e.g., port-numbers and payload inspection). In fact, approaches based on statistical properties of network traffic provide viable alternatives [3, 22].

Leveraging the information directly provided by MIRAGE-2019, the mobile-app traffic can be modeled at different granularities, considering both the apps and the related categories. As an interesting example, Fig. 5 reports the joint scatter plot (per biflow) of the *mean* packet lengths and inter-arrival times for three different app categories: `Productivity` (Fig. 5a), `Sports` (Fig. 5b), and `Games` (Fig. 5c). Each figure reports the apps with different colors, while the kernel density estimation of the marginal distributions is shown in the side plots. Several considerations can be drawn from this analysis.

First, given the different spatial concentration of the points over the plane, different categories result in different joint scatter plots. For instance, while points in `Games` mostly lie within $[50, 400]$ B (x-axis) and $[10\,\text{ms}, 100\,\text{s}]$ (y-axis), the same *does not apply* for the other two categories.

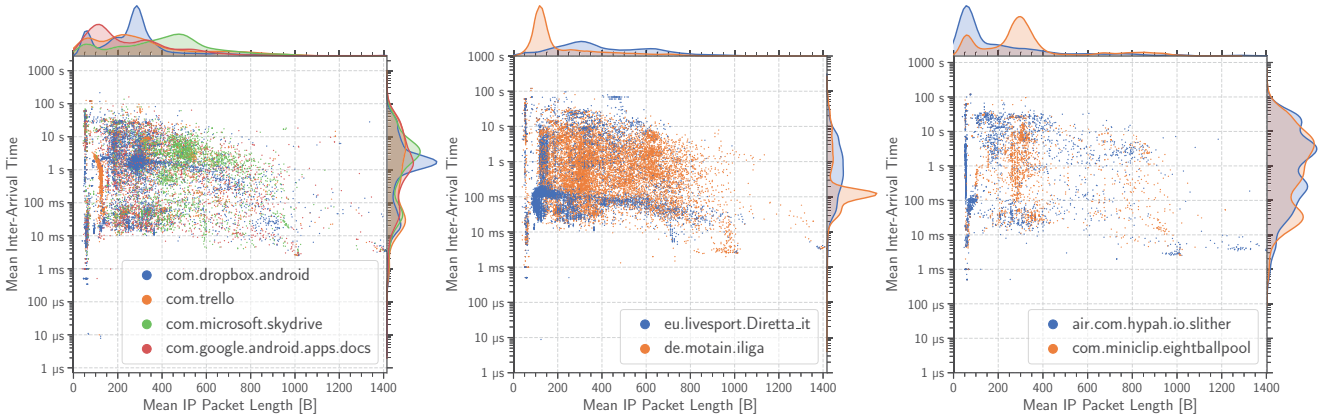
Secondly, apps within the same category often show their own peculiar statistical profile. For example, in Fig. 5b, *OneFootball* (`de.motain.iliga`) updates generate points mostly concentrated at 100 B and 100 ms, as opposed to *Diretta* (`eu.livesport.Diretta_it`). Differently, for `Games` (Fig. 5c), while inter-arrival time does not hold great discriminating power, the packet length allows to separate biflows associated to the two considered games easily enough.

Finally, this rationale is not as much evident for apps within `Productivity` (see Fig. 5a, except for Dropbox). Indeed, the presence of traffic generated by several (similar) apps is one of the main challenges of mobile-app traffic analysis. This results in very-complex patterns of current traffic which cannot be appropriately captured by common statistical features. Accordingly, the application of novel machine- and deep-learning based techniques is foreseen to be the effective workhorse to cope with this challenge [16].

C. Per-flow volume distribution

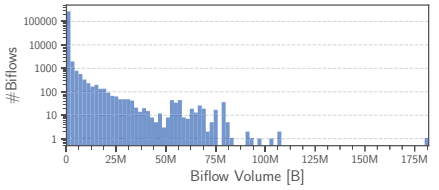
The info in MIRAGE-2019 also enables a characterization of the traffic based on the transferred volumes. Figure 6a shows the volume of the biflows in MIRAGE-2019, reporting the histogram of their sizes (bin width: 2 MB) which range from few bytes to several megabytes. In general terms, voluminous flows are less frequent than (possibly short-lived) biflows transferring a limited amount of bytes (e.g., due to request-response interactions). As more than 88% of the biflows have volumes ≤ 100 kB, Fig. 6b reports the histogram (bin width: 1 kB) for such dataset portion, revealing that biflows with volumes within $[5, 10]$ kB are the most common.

To deepen the nature of the exchange, Fig. 6c reports (in the form of cumulative distributions) the *share* ρ_d of the downstream bytes for each biflow (i.e. $\rho_\text{d} \triangleq \frac{B_\text{d}}{B_\text{d}+B_\text{u}}$, where B_d and B_u are the number of *downstream* and *upstream* bytes of the biflow, respectively) for the whole MIRAGE-2019 ($\approx$ 270k biflows). *On average*, downstream traffic accounts for $\approx 65\%$ of the volume of the whole biflow traffic. However, a clear pattern can be noticed when considering biflows with different volumes: the smaller the biflow volume, the smaller

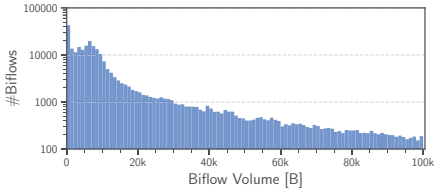


(a) Productivity. (b) Sports. (c) Games.

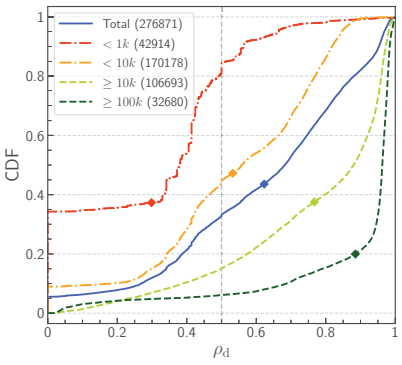
Figure 5: Joint scatter-plot of mean (payload length, inter-arrival time) of each biflow for three different app categories. The packet length is reported in linear scale (x-axis), whereas the inter-arrival time is shown using a log-scale ($10\log_{10}(x)$).



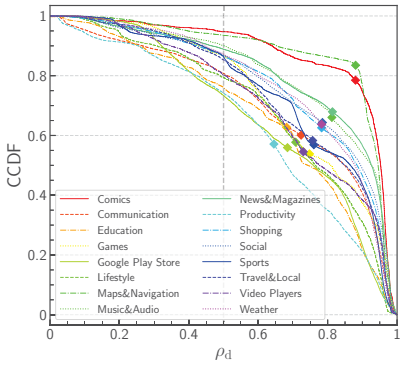
(a) Volume histogram of all biflows (bin width: 2 MB).



(b) Volume histogram of biflows with volume $\leq$ 100 kB (bin width: 1 kB).



(c) Cumulative distribution of ρ_d. Diamonds mark the average.



(d) Complementary cumulative distribution of ρ_d for each app category (biflows with volume $\geq$ 10 kB). Diamonds mark the average.

Figure 6: Per-flow characterization of MIRAGE-2019 biflows in terms of byte volume (a-b) and downstream volume share ρ_d (c-d).

the share of downstream traffic ($\rho_d \leq 0.5$). Indeed, for biflows very small in volume (i.e. $\leq$ 1 kB) $\rho_d = 0.35$ on average. Notably, for around 5% of the biflows no downstream packets were captured. As expected, all these communications are very small in volume (i.e. $\leq$ 10 kB) and are the results of anomalous conditions, with the mobile app asking for services to external servers and no reply returned due to failures possibly at the communication endpoint or along the path. Differently, when only flows with larger sizes are retained ($\geq$ 10 kB and $\geq$ 100 kB), the downstream traffic accounts for most of the volume (mean ρ_d equals to 0.77 and 0.89).

The analysis of the exchanged traffic volumes and of the downstream-upstream (un)balance has an impact on access link dimensioning (for the operator) and per-app traffic volume costs, and transmission power consumed (for the user). Indeed, breaking the above results down by different app categories (see Fig. 6d), it is possible to quantify the expected difference of transmission resources required. According to this analysis, two categories (`Comics` and `Maps&Navigation`) show a distinct distribution of ρ_d from the other ones (a higher mean value and only $\approx 5\%$ of the biflows with $\rho_d \leq 0.5$).

VI. Ethical Considerations

Based on both the presented design choices and the experimental setup, the capture process and the collected dataset do not imply any ethical concern [1]. We remark that experimenters involved in the acquisition phase have been beforehand informed and warned about the objectives of their activities (e.g. network traffic analysis) and the possible public release of the corresponding traces for research purposes, even in a complete form. Additionally, each experimenter received a thirty-minutes training phase (with the help of a written document listing all the instructions) regarding the capture technologies employed and the related working principles.

Moreover, the employed mobile devices were provided and used within the ARCLAB. The logged source IP addresses

belong to the *private* IPv4 space, with no privacy implications. Equally important, purposedly-created app accounts were employed for the capture sessions. The GT building phase is automated and does not contribute any additional experimenter-related information. Hence, no personal information of the experimenters was involved at any time from the capture phase to the dataset creation. Collected traces were made available to experimenters that captured them for educational purposes as well as to verify the non-sensitive nature of their contents.

VII. Conclusions

This paper introduces and details the reproducible MIRAGE architecture for capturing mobile-app traffic and building the related ground-truth. Leveraging the implemented architecture, we collected mobile-app traffic for two years and release the MIRAGE-2019 dataset, supporting and fostering replicable traffic analysis. Example tasks enabled by the released dataset with focus on traffic characterization and modeling are also shown in this paper. We expect that the proposed dataset will be used as a benchmark for future studies concerning mobile traffic analysis. Our intention is to periodically release updated (and enriched) versions of the dataset.

VIII. Acknowledgements

We are grateful to the students of '17/'18 and '18/'19 classes of "Internet Analysis and Performance", "Computer Architectures" and "Computer Networks" held at University of Napoli "Federico II" for willingly participating to the capture campaign needed to generate MIRAGE-2019.

References

[1] V. Bajpai, A. Brunstrom, A. Feldmann, W. Kellerer, A. Pras, H. Schulzrinne, G. Smaragdakis, M. Wählisch, and K. Wehrle. The Dagstuhl beginners guide to reproducibility for experimental networking research. *ACM SIGCOMM CCR*, 49(1), 2019.

[2] A. Dainotti, R. Holz, M. Kühlewind, A. Lutu, J. Sommers, and B. Trammell. Open collaborative hyperpapers: a call to action. *ACM SIGCOMM CCR*, 49(1), 2019.

[3] A. Dainotti, A. Pescapè, and K. C. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 26(1), 2012.

[4] F. Jejdling et al. Ericsson mobility report. *Ericsson AB, Business Area Networks, Stockholm, Sweden, Tech. Rep. EAB-19*, 3442, 2019.

[5] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani. Characterization of encrypted and VPN traffic using time-related features. In *SciTePress ICISSP'16*.

[6] A. H. Lashkari, G. Draper-Gil, M. Mamun, I. Saiful, and A. A. Ghorbani. Characterization of Tor traffic using time based features. In *SciTePress ICISSP'17*.

[7] K. Shahbar and A. N. Zincir-Heywood. Packet momentum for identification of anonymity networks. *Journal of Cyber Security and Mobility*, 6(1), 2017.

[8] V. Tong, H. A. Tran, S. Souihi, and A. Mellouk. A novel QUIC traffic classifier based on convolutional neural networks. In *IEEE GlobeCom'18*.

[9] R. Wang, Z. Liu, Y. Cai, D. Tang, J. Yang, and Z. Yang. Benchmark data for mobile app traffic research. In *ACM MobiQuitous'18*.

[10] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans. Mobile Comput.*, 2018.

[11] M. Lescisin and Q. H. Mahmoud. Dataset for web traffic security analysis. In *IEEE IECON'18*.

[12] R. Dubin, A. Dvir, O. Pele, and O. Hadar. I know what you saw last minute-encrypted HTTP adaptive video streaming title classification. *IEEE Trans. Inf. Forensics Security*, 12(12), 2017.

[13] T. Karagkioules, D. Tsilimantos, S. Valentin, F. Wamser, B. Zeidler, M. Seufert, F. Loh, and P. Tran-Gia. A public dataset for YouTube's mobile streaming client. In *IEEE/IFIP MNM Workshop'18*.

[14] S. C. Madanapalli, H. H. Gharakhieli, and V. Sivaraman. Inferring netflix user experience from broadband network measurement. In *IEEE/ACM TMA'19*.

[15] Sandvine. Global Internet Phenomena Spotlight: Encrypted Internet Traffic., 2016.

[16] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans. Netw. Service Manag.*, 16(2), jun 2019.

[17] 42matters. Google play categories. https://42matters.com/docs/app-market-data/android/apps/google-play-categories, 2019.

[18] Z. Wang. The applications of Deep Learning on traffic identification. *BlackHat USA*, 2015.

[19] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapé, and C. Sansone. Identification of traffic flows hiding behind tcp port 80. In *2010 IEEE International Conference on Communications*, May 2010.

[20] V. Rimmer, D. Preuveneers, M. Juarez, T. V. Goethem, and W. Joosen. Automated feature extraction for website fingerprinting through deep learning. In *NDSS'18*.

[21] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Trans. Inf. Forensics Security*, 13 (1), 2018.

[22] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. Multi-classification approaches for classifying mobile app traffic. *Journal of Network and Computer Applications*, 103, 2018.

[23] V. Carela-Español, P. Barlet-Ros, M. Solé-Simó, A. Dainotti, W. de Donato, and A. Pescapé. K-dimensional trees for continuous traffic classification. In *Traffic Monitoring and Analysis*, Berlin, Heidelberg, 2010.

[24] A. Dainotti, A. Pescapé, P. Salvo Rossi, F. Palmieri, and G. Ventre. Internet traffic modeling by means of hidden Markov models. *Computer Networks*, 52(14), 2008.