Characterization and Prediction of Mobile-App Traffic using Markov Modeling

Giuseppe Aceto, Giampaolo Bovenzi, Domenico Ciuonzo, *Senior Member, IEEE*, Antonio Montieri, Valerio Persico and Antonio Pescapé, *Senior Member, IEEE*

***Abstract*—Modeling network traffic is an endeavor actively carried on since early digital communications, supporting a number of practical applications, that range from network planning and provisioning to security. Accordingly, many theoretical and empirical approaches have been proposed in this long-standing research, most notably, *Machine Learning* (ML) ones. Indeed, recent interest from network equipment vendors is sparking around the evaluation of solid information-theoretical modeling approaches complementary to ML ones, especially applied to new network traffic profiles stemming from the massive diffusion of mobile apps. To cater to these needs, we analyze mobile-app traffic available in the public dataset *MIRAGE-2019* adopting two related modeling approaches based on the well-known methodological toolset of Markov models (namely, *Markov Chains* and *Hidden Markov Models*). We propose a novel heuristic to reconstruct application-layer messages in the common case of encrypted traffic. We discuss and experimentally evaluate the suitability of the provided modeling approaches for different tasks: characterization of network traffic (at different granularities, such as application, application category, and application version), and prediction of network traffic at both packet and message level. We also compare the results with several ML approaches, showing performance comparable to a state-of-the-art ML predictor (Random Forest Regressor). Also, with this work we provide a viable and theoretically sound traffic-analysis toolset to help improving ML evaluation (and possibly its design), and a sensible and interpretable baseline.**

***Index Terms*—Android apps; encrypted traffic; Markov models; mobile apps; traffic characterization; traffic modeling; traffic prediction.**

I. Introduction

THE CLEAR UNDERSTANDING of the processes occurring in networks is paramount for multiple stakeholders, including network operators, who aim at the full visibility required by both network management and security [1–3]. Accordingly, modeling network traffic is of the utmost importance to understand traffic peculiarities, predict its characteristics, enforce traffic engineering, perform network planning and provisioning, manage the QoS, profile user activities, identify anomalies, emulate real traffic for testing purposes, etc.

However, this process is challenged by the nature of the traffic traversing today's networks that is impacted by the way users behave, interact, and access the network. In fact, operators have experienced in the last years tremendous growth of the traffic to be managed in their networks, mostly generated by mobile devices [4]. According to the latest Ericsson mobility report [5], between Q3 2018 and Q3 2019, mobile data traffic has grown 68%, being fueled by both the rising number of smartphone subscriptions and the increasing average data volume per subscription. Overall, it is forecasted that mobile subscriptions will reach 8.9 billions by 2025, corresponding to a mobile data traffic of 160 exabytes per month against the 38 exabytes of 2019. This phenomenon exacerbates the need for accurate characterization, modeling, and predictability of network traffic generated by mobile devices at fine grain.

Recent contributions to mobile-traffic characterization mainly focused on traffic at an aggregate scale [6, 7]. Opposed to this, a number of tasks cannot overlook fine granularity. Indeed, fine-grained source traffic models are the key to reproduce realistic mobile application behavior in simulative environments or in network testbeds, or to better predict traffic evolution and enforce smarter network traffic management. Hence, such modeling granularity requires more advanced approaches than those applied for aggregated network traffic.

In fact, the implementation of effective approaches for fine-grained traffic characterization and modeling must overcome several challenges. The *broad adoption of encrypted protocols*, e.g. Transport Layer Security (TLS), blocks the road to modeling approaches based on packet inspection, as encrypted network traffic represents the majority of mobile traffic (80% of all Android apps, and 90% of apps targeting Android 9 or higher). Thus, modeling approaches cannot rely on clear-text patterns to identify application fingerprints or a specific execution state [8], or even reconstruct application message boundaries. Also, mobile traffic is an extremely *complex* and *dynamic* phenomenon [9]. Indeed, generated traffic can show wildly different and complex fingerprints, due to the multifold nature of both tasks carried on by means of mobile terminals and user activities within the same app, besides potential device/OS/app-version diversity [9, 10]. Finally, *the availability of high-quality and up-to-date datasets and ground truths* for needed analyses is quite limited, given both the variety and the dynamicity of mobile apps, and the privacy concerns implied in the collection and sharing of such data [11].

These characteristics exacerbate the difficulties in designing and evaluating Machine Learning (ML) techniques applied to traffic modeling, characterization, and prediction. These issues are the more worrying the less interpretable the ML technique is, with the currently-popular neural networks (in their deep learning "flavour") representing the most exposed ones. Hence, characterization and prediction of mobile-app traffic remains

Manuscript received 1st May 2020; revised 21th September 2020, 14th December 2020 and 8th January 2021.

The authors are with the Department of Electrical Engineering and Information Technologies (DIETI) at University of Naples Federico II, Italy.

E-mail: {giuseppe.aceto, giampaolo.bovenzi, domenico.ciuonzo, antonio.montieri, valerio.persico, pescape}@unina.it.

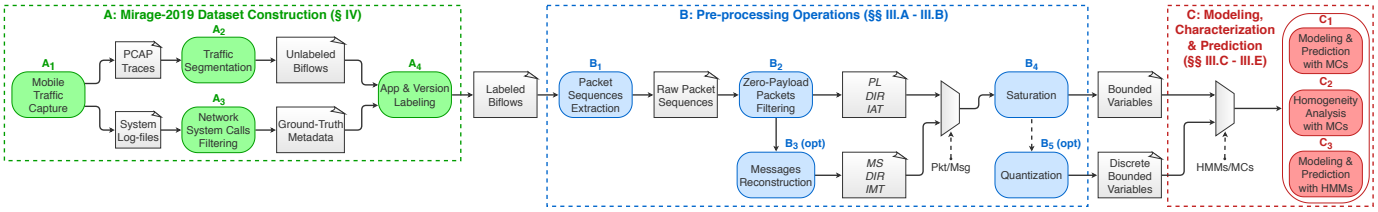


Figure 1: Overall proposed workflow to support traffic modeling, characterization, and prediction via Markov models. It consists of three main phases: *(A)* Dataset Construction; *(B)* Pre-processing; and *(C)* Traffic Modeling, Characterization, and Prediction.

an open and hot challenge, and well-established modeling approaches can provide a complementary, theoretically-sound and easily-interpretable view of this traffic.

Prompted by the interest of a global network solutions provider[1], we investigate the applicability of Markov models to fine-grained characterization and modeling of network traffic generated by mobile applications, as collected in a public (and reliably-annotated) dataset. We also compare the performance of such methods to a number of relevant ML techniques (i.e. Random Forest, k-Nearest Neighbors, and Linear Regressor), showing comparable results and finding high overlap of the outcomes but also room to interpret ML failures. Figure 1 depicts a scheme for the overall proposed workflow and provides the context to highlight the contributions of this work. In detail, *three* main *stages* can be identified: (A) building on a previous proposal [11], we leverage the MIRAGE traffic capture procedure to obtain a labeled mobile-app traffic dataset; (B) we design and evaluate a pre-processing procedure to deal with the nature of the observed data; (C) we design and evaluate *different modeling and prediction strategies* as well as a novel *strategy to quantitatively assess traffic heterogeneity*. Beyond the design and evaluation of the single blocks implementing novel approaches pursuing paths not investigated beforehand, we highlight that our proposal does not merely consist in the straightforward adaptation of well-known approaches, as each block has to be tailored in line with the specific problem to address. Further, some non-trivial dependencies among blocks occur. In detail, the contributions of this paper can be summarized as follows:

- We characterize and predict mobile-app network traffic *at packet level*, i.e. focusing on payload lengths, inter-arrival time, and packet direction. Such novel application scenario is considered herein *for the first time*, up to our knowledge. To accomplish this task, we leverage Markov models due to their simplicity and interpretability.
- We propose a solid *composite hypothesis testing* methodology to evaluate traffic similarities at different granularities, namely (a) intra-app (same app, different versions), (b) inter-app (different apps), (c) inter-category (different categories) similarities. Such a methodology supports the intent of investigating and assessing the need for different models, as opposed to a single model for different apps.
- We introduce a *novel heuristic to reconstruct application-level messages* from the series of packets in order to capitalize the knowledge derived from different levels of abstraction; this proposed heuristic is also experimentally validated against timing and `PUSH`-flag based criteria.
- We design and evaluate per-app modeling and prediction by means of *Hidden Markov Models* and *high-order Markov Chains*, *both* at packet- and message-levels; per-app modeling and prediction tasks are supported by a sequence of *pre-processing operations* which are required to deal with the nature of the data.
- We compare the designed prediction approaches against state-of-the-art ML baselines in order to assess the trade-off between the achieved performance and the interpretability degree achieved by different approaches.
- We perform all analyses on the *public dataset* MIRAGE-2019 adopting the same guidelines as specified in [11] to foster reproducibility. The above study benefits from the flexibility of MIRAGE-2019 when experimenting on relevant mobile traffic analytics scenarios.

The paper is organized as follows. Section II surveys related works from both application scenario and methodological viewpoints, positioning our work against past contributions. Section III describes the considered methodology; the dataset employed and the experimental results are discussed in Secs. IV and V, respectively; finally, Sec. VI provides conclusions and future perspectives.

II. Related Works

In this section, we first survey works that provided remarkable contributions in the field of mobile-app network traffic characterization and modeling (cf. Sec. II-A). Then, we discuss past works that used Markov-based approaches for modeling and predicting network traffic behavior at different timescales (cf. Sec. II-B). Finally, we position our contribution against both aspects and highlight its novelty (cf. Sec. II-C).

A. Characterization and modeling of mobile-app traffic

Due to the increasing interest towards the network traffic generated by mobile devices, a number of works have focused on its characterization to catch the peculiarities of this traffic from different points of view. The first remarkable attempts focused on the *usage of mobile devices* from a network perspective and based the analysis on anonymized packet-level data representing residential DSL customers [6] or data from users participating in collecting network traffic traces or traffic statistics (i.e. bytes sent and received) [7]. These works resulted in characterizing the usage of mobile devices in terms of most popular applications and device brands, and

[1]NDA prevents the disclosure of further details.

also provided performance assessments in terms of protocol overhead, packet loss, and throughput. Network measurements from a US national level tier-1 cellular network provider also allowed for investigating smartphone-app usage patterns (highlighting similarities in terms of geographic coverage, correlation among app occurrence, and local patterns) [12] or comparing traditional smartphone traffic with cellular network based machine-to-machine communication [13]. Recently, Wei et al. [14] designed a framework to group bring-your-own-handheld devices according to their behavior, via profiling dimensions such as control plane, data plane, and temporal behavior. Dai et al. [8] devised an approach for generating network profiles to distinguish mobile apps based on patterns occurring in HTTP headers and the host the app connects to. Van Ede et al. [9] proposed a semi-supervised approach for fingerprinting (iOS and Android) mobile apps based on network-traffic features, such as packet size, inter-flow time, and contacted destinations. Also, they provided a preliminary analysis of app evolution with focus on their fingerprints.

B. Markov-based modeling and prediction approaches for network traffic

The scientific literature witnesses that Markov models provide a set of valuable, well-known, and interpretable tools for analyzing network traffic in several different contexts and with diversified goals. Accordingly, we survey the most relevant works in the following.

A number of works recently leveraged models based on **Markov Chains** in the context of anomaly detection (e.g., in cybersecurity applications) [3, 15, 16] and traffic modeling and prediction [17], also showing the effectiveness of multi-order approaches (which model a longer dependence) [16].

Models based on Markov-Modulated Poisson Processes (MMPPs) were also investigated [18–20]. Muscariello et al. [18] presented an MMPP traffic model that accurately approximates the long-range dependence characteristics of Internet traffic traces over the relevant time scales, with a few trainable parameters. The core of the model is based on the notion of sessions and flows, trying to mimic the real hierarchical generation of packets in the Internet. Okamura et al. [19] consider a parameter estimation problem for Markovian arrival processes with focus on MMPPs, which are examined via numerical experiments and some analyses with real traffic data. Casale et al. [20] developed the first counting process fitting algorithm for the marked MMPP (M3PP), a generalization of the MMPP for modeling traces with events of multiple types.

Recent literature also witnesses the suitability of **Hidden Markov Models (HMMs)** to model network traffic at a number of levels of abstraction [21–27]. Colonnese et al. [21] addressed the modeling of traffic generated by video sources (e.g. H.264) operating in the context of adaptive streaming services via HMMs and derived a model of the sequence of the encoded video-frame sizes. The proposed HMM model—which models the sequence of Groups of Pictures (GOPs) as a first-order homogeneous Markov Chain—significantly outperforms a previously-proposed model [28]. Dainotti et al. [22] proposed an application-specific packet-level model for various Internet applications, based on HMMs. In detail, the model jointly considers the inter-packet time and packet size distributions. The applicability of the proposed model is assessed using real traffic (SMTP, HTTP, online gaming and online messaging). After presenting the results of the model, the prediction capability of HMM is also evaluated, showing good accuracy in predicting short-term patterns. Maheshwari et al. [23] focused on modeling wireless Internet traffic and forecasting the QoS parameters for the networks such as end-to-end delay, inter-packet delay variation, and packet size. The proposal is validated evaluating the forecasting accuracy over both self-collected and publicly-available datasets (covering several source-destination network access conditions). The same authors also proposed a similar HMM-based solution to represent packet-level network traffic, applied to 4G/5G network traffic [24]. The results showed that the end-to-end delay and the inter-packet delay variation can be modeled using an HMM with good accuracy and prediction performance. **Extensions of classic HMM models** (such as Hidden Semi-Markov Model (HSMM) [25], Markov Modulated Gamma (3D-MMG) model [26], and Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) [29]) have been also proposed and provided good performance (e.g. to model multiview video traffic or round-trip time measurements).

C. Positioning of our Contribution

Concerning the **application scenario** we address in this paper, *none of the works* discussed in Sec. II-A (up to our knowledge) *has provided a characterization of the traffic generated by mobile apps* at *both* packet and message level in order to highlight the peculiarities of the (encrypted) traffic for a specific app, its version, and the corresponding category. Indeed, a preliminary attempt in this direction is only provided by the use-cases investigated by our previous work [11]. To this end, in this paper, we deeply investigate the suitability of different Markov-based approaches for modeling and predicting (at the finest granularity achievable) network traffic generated by mobile devices—rather than their usage. Notably, strategies benefiting from packet inspection as those investigated by previous works [8] are not suitable for encrypted traffic (representing $> 80\%$ of mobile-app traffic [30]). Similarly to very recent contributions [9], we also investigate how the network traffic changes when apps evolve over time, by resorting to a recent, public, and human-generated dataset. However, our study focuses on modeling, characterization, and prediction (as opposed to fingerprinting). Additionally, our approach does not rely on destination-related features. Also, in this work we do not consider any specific use case for the obtained modeling and prediction results, because our approach is meant to be *orthogonal* to the intended usage. Indeed, the fine grain at which the analysis is performed makes our proposal able to support also those use cases that require modeling and prediction capabilities at very small scales.

Concerning our proposed **methodology**, this work investigates the adoption of (high-order) Markov Chains and HMMs for evaluating the peculiarities of network traffic generated by mobile apps by (i) modeling, (ii) characterizing, and

(iii) predicting their fine-grained characteristics at packet- and message- level. For predictability analysis, Markov-originated predictors are also herein exploited, for the first time, as a sound attempt to provide interpretability for ML-based prediction results [31]. Notably, the considered fine-grained analysis (i.e. at packet level) results in a *conceptually-different* methodology from the works reviewed in Sec. II-B, with the sole exception of the work by Dainotti et al. [22] (focusing however *only* on the modeling part and unidirectional flows). While research efforts addressed the problem of traffic classification also at packet level [32], to the best of our knowledge, previous solutions for network traffic modeling and prediction have addressed these problems at other granularities than packet and message level. Indeed, the states of our models reflect the (bidirectional) characteristics of the network traffic (e.g., packet and message size, inter-arrival times between packets, and their direction) rather than other aspects (e.g., application-level frame characteristics usually considered for video traffic [21, 28] or destination-related features [9]). Searching the state-of-art applications of ML to prediction problems strictly similar to the one we tackle, we found two works as the most related. Both adopt Random Forest Regressor (RFR) for the prediction (i) of TCP state machine transitions [33] and (ii) of the Quality of Experience degradation from encrypted traffic [34]. Moreover k-Nearest Neighbors (kNN) has been used for traffic classification based on the joint distribution of packet size and inter-packet times [32], representing two of the relevant features modeled/predicted in our setup. Accordingly, we selected RFR and kNN as the most relevant ML techniques to analyze together with Markov approaches, and considered the Linear Regressor as a further relevant baseline. Although this set of baselines is not intended to be exhaustive (we believe that ML approaches have the potential to provide further benefits) these methods were never applied to the problem of mobile traffic prediction at packet and message level, to date. Hence, we believe that they are the first ones that are worth to be considered.

Moreover, with respect to the related literature, the characterization part in our work relies on *composite hypothesis testing* for evaluating homogeneity (and, in negative case, assessing the degree of dissimilarity) among traffic of different versions/apps/categories, as opposed to goodness-of-fit tests explored for anomaly detection [3, 15]. Finally, none of the related works adopts a *message-level* view and analysis. To this purpose, we propose a novel heuristics based only on *maximum segment size* and network-aware, measuring the per-packet *actual* maximum segment size value. We discuss the heuristics applicability, and evaluate it by comparison with timing-based and `PUSH`-flag based approaches. To the best of our knowledge, the only related work performing message-level reconstruction [35] on encrypted traffic adopts an (arbitrary) fixed maximum segment size corresponding to the *maximum transfer unit*, and also requires (arbitrary) fixed time-based segmentation, without any empirical or theoretical discussion of the adopted choices. Other works [36, 37] focused on TLS protocol messages: however, they do not aim at inferring the message size, but rather at defining classification features ("fingerprints"), namely in terms of `TLS Message Type` code and their variations over the packet sequence. Hence the above aims, input data, and methods differ significantly from those tackled in our work.

Summing up these considerations, our contributions provide a methodological tool for in-depth analysis and modeling specifically targeted at the challenging mobile app-generated traffic, which evolves at fast pace [9, 10].

III. Methodology

When analyzing network traffic, different degrees of granularity can be considered in aggregating packets. These can result in different *Traffic Objects (TOs)*, used as a unit for traffic modeling, prediction, or generation [4]. The properties of traffic traversing networks can vary because of a number of *parameters* of different nature, resulting in different *random variables* that can be defined. In practical scenarios, these parameters are controlled or tracked, or can be inspected to characterize a type of traffic. Our analysis considers the *bidirectional flow* (biflow) as TO. The latter is defined through the quintuple `(IP src, IP dst, port src, port dst, protocol)`, with the source and the destination pairs interchangeable [4]. The objective of this work is to model mobile applications' network traffic at *three* different *granularities*: *(i)* version, *(ii)* application, and *(iii)* category.

In detail, we point to model the *payload length* (`PL`), the *direction* (`DIR`), and the *inter-arrival time* (`IAT`) of packets belonging to the same TO, being `(PL,DIR,IAT)` a common choice for network traffic modeling [22]. Specifically, `PL` is defined as the size (in bytes) of the payload of TCP/IP transport layer, whereas `IAT` is defined as the time between two packet arrivals. In addition, we also investigate traffic modeling based on application-layer *message size* (`MS`)—and related *inter-message time* (`IMT`). Using payload length to characterize application *intrinsic* network behavior is subjected to interfering phenomena, some related to network conditions, other due to the modern protocol sub-layers constituting the application layer, possibly including HTTP/2 multiplexing[2], WebSocket multiplexing[3], TLS segmentation and padding[4]. When TLS or other encryption sub-layer is used, these effects are opaque to traffic analysis and modeling. The last segmentation processing is performed by TCP, that constrains payload length to the—network-dependent and dynamically estimated—Maximum Segment Size: with `MS` and `IMT` reconstruction this path-dependent effect can be mitigated. Accordingly, `MS` and `IMT` are expected to carry pieces of information whose impact is worth to be investigated, providing a complementary view to the common `PL` and `IAT` one.

The reconstruction of `MS`s (and consequently the inference of `IMT`s) from the observed payload lengths is achieved via a heuristic we propose (cf. later Sec. III-B) to solve the lack of visibility due to traffic encryption (usually present in mobile traffic). Notably, based on the range of values the considered random variables may assume and the modeling approaches we adopt, some pre-processing operations may be required.

[2]RFC 7540: https://rfc-editor.org/rfc/rfc7540.txt
[3]RFC 6455: https://rfc-editor.org/rfc/rfc6455.txt
[4]RFC 8446: https://rfc-editor.org/rfc/rfc8446.txt

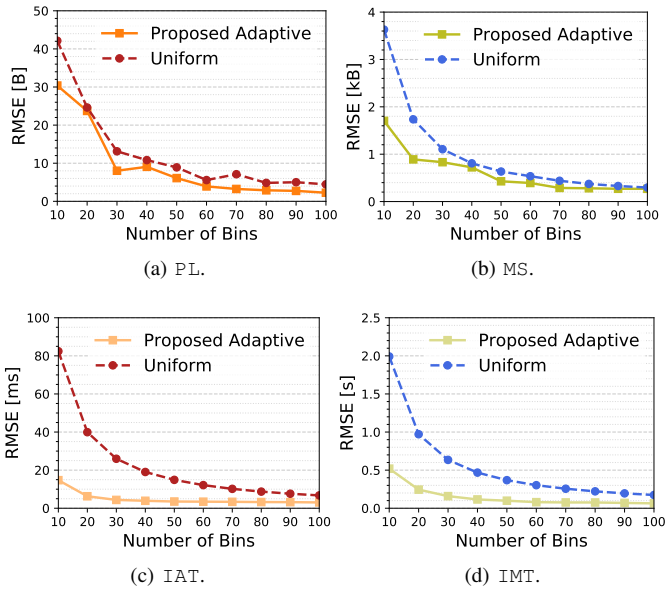


Figure 2: RMSE of the proposed adaptive (based on K-Means) and uniform binning strategies for (a) PL, (b) MS, (c) IAT, and (d) IMT.

The overall workflow based on the proposed methodology is reported in Fig. 1. In more detail, in order to obtain the (labeled and human-generated) TOs, we refer to a reproducible procedure we recently proposed in [11], whose main relevant aspects (depicted by green blocks, steps from A_1 to A_4 in Fig. 1) are detailed in later Sec. IV. Then, a number of pre-processing operations (blue blocks, steps from B_1 to B_5) are required to address issues deriving from the nature of the random variables under investigation and provide the data with the format required by the adopted modeling strategies as described in Sec. III-A. These operations also encompass the heuristic procedure we propose to reconstruct MS and IMT, whose details are provided in Sec. III-B (step B_3). Finally (red blocks), the proposed modeling and prediction approaches for mobile-app traffic based on multi-modal Markov Chains (step C_1) and HMMs (step C_3) can be applied, as presented in Sec. III-C and Sec. III-E, respectively. The same pre-processing steps are also adopted to feed the procedure we enforce for a quantitative comparison of the snapshot characterizing network traffic presented in Sec. III-D (step C_2).

A. Pre-processing operations

First, the collected traffic biflows are processed in order to obtain *raw packet sequences* (B_1). Then, a number of pre-processing operations are applied to deal with the nature of the variables under investigation and obtain the input to effectively feed Markov models. *Zero-payload packets* are removed from the considered biflows[5], and IATs are calculated on the resulting sequences (B_2). Optionally, the *message reconstruction* procedure described in Sec. III-B is applied when the analysis aims at considering MSs and IMTs (B_3).

[5]Zero-payload packets are assumed to be non-informative, as they are representative of transport-layer mechanisms (e.g., TCP handshake, pure acknowledgments, etc.) rather than application behaviors.

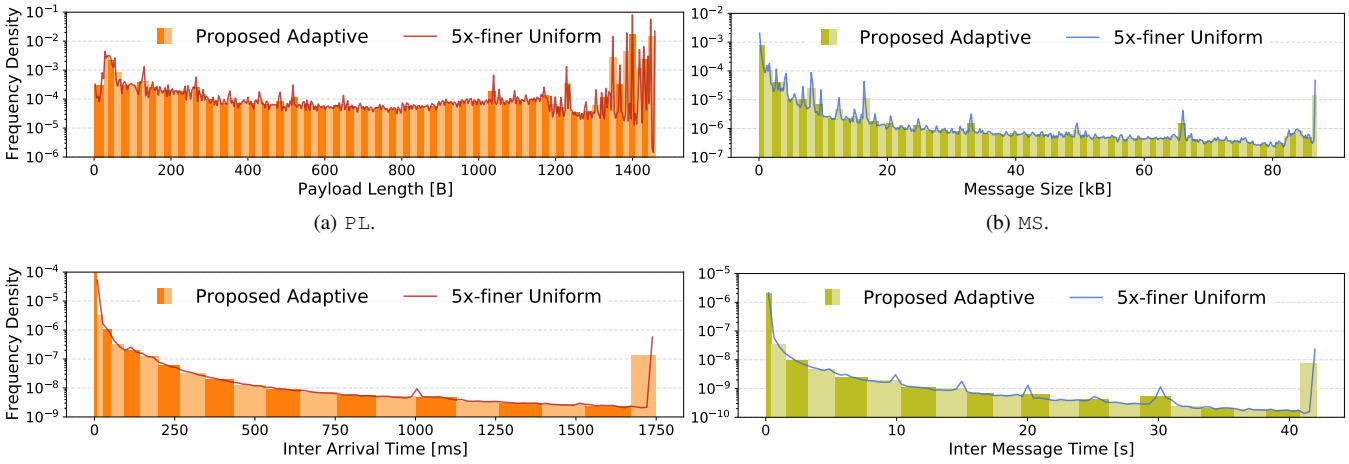
It is worth noting that the variables we deal with (i.e. PL, MS, IAT, and IMT) originally have different nature and characteristics: MS, IAT, and IMT are virtually unbounded (while PL values are practically limited at 1500 B); IAT and IMT have a continuous nature (captured with 1μs-granularity) whereas PL and MS have a discrete nature (at a very fine—1B—granularity). Beyond our need to obtain *bounded* and *discrete* random variables when employing Markov Chains (cf. Secs. III-C and III-D), note that the number of selected bins defines the number of states S_m of each modality and therefore directly impacts the complexity of the resulting model. Moreover, binning selection may improve the accuracy of the predictive models by reducing the noise or non-linearity. Accordingly, to realize the binning procedure we further apply *two* kinds of operations: *saturation* (B_4) and (optionally) *quantization* (B_5), which are aimed at obtaining bounded and discrete random variables, respectively, to apply Markov Models on. We remark that the quantization step B_5 is *only applied to Markov Chains*, whereas HMMs do not require this step as they rely on a continuous emission distribution.

Hence, we first saturate all the IATs and IMTs to the 99^{th}-percentile values to remove the effects due to outliers in measuring times. We also apply the same procedure to the MSs due to the wide (and possibly unbounded) range of values. For MIRAGE-2019 dataset (described in Sec. IV), this results in 86.85 kB, 1.75 s, and 42.09 s for 99^{th}-percentile values of MS, IAT, and IMT, respectively (cf. x-axis ranges in Figs. 3b, 3c, and 3d). On the other hand, to address quantization issues we resort to an *unsupervised* approach, i.e. taking into account neither app nor category labels. This allows obtaining a unique (independent) binning choice which can be used to perform comparisons (resp. predictions) among (resp. for) different versions, apps, or categories.

To this end, in Fig. 2 we report the *Root-Mean-Square Error (RMSE)* due to quantization versus a varying number of bins for PL (a), MS (b), IAT (c) and IMT (d), on the *whole* MIRAGE-2019 dataset. In the above figure, *two* quantization strategies are compared: (i) a uniform quantization and (ii) a non-uniform (adaptive) quantization, obtained via K-means clustering. In particular, results on MIRAGE-2019 highlight an RMSE settlement at ≈ 80 (resp. ≈ 20) bins for both strategies and PL and MS (resp. IAT and IMT) features, with the adaptive approach performing the best. Specifically, by selecting 80 bins for both PL and MS, the adaptive strategy achieves an RMSE of ≈ 3 B and ≈ 280 B, respectively. Differently, by considering 20 bins for both IAT and IMT, the RMSE corresponds to ≈ 6.9 ms and ≈ 244.3 ms, respectively.[6] The above result indicates that a higher number of bins would imply only marginal RMSE reduction, while leading to (i) an inaccurate Markov Chain estimate (due to insufficient number of samples) and (ii) a higher complexity of the model.

To support the above claim, we select an 80-bin (resp. 20-bin) non-uniform quantization for PL and MS (resp. IAT and IMT) and show the corresponding outcomes in Fig. 3. In detail, the bars in Fig. 3 report the results of the detailed

[6]The relative gain (viz. reduction) of quantization RMSE against uniform binning is $\approx 40\%$ and $\approx 25\%$ for PL and MS, respectively. Instead, for IAT and IMT this corresponds to $\approx 84\%$ and $\approx 75\%$, respectively.



(a) PL. (b) MS. (c) IAT. (d) IMT.

Figure 3: Histograms of (a) PL, (b) MS, (c) IAT, and (d) IMT obtained via the proposed adaptive binning (80 bins for MS/PL and 20 bins for IAT/IMT) and a $5\times$-finer uniform binning (400 bins for MS/PL and 100 bins for IAT/IMT) strategy. It is worth to underline that peaks at the highest values (last bins) of MS, IAT, and IMT are due to the presence of saturation at the 99^{th} percentile.

preprocessing operations on the whole dataset, while y-axes show the corresponding *frequency density*.[7] Further, to qualitatively assess the (lossy) impact of our binning procedure, we compare its results with a $5\times$-finer binning strategy (i.e. based on 400 and 100 bins for PL/MS and IAT/IMT, respectively). The latter histogram is highlighted via solid (red/blue) lines in Fig. 3. Compared with a $5\times$ more complex binning approach, the proposed binning—allowing to deal with Markov Models with a manageable number of states—incurs a limited and acceptable quantization error.

B. Reconstructing message size and inter-message time

Packet-based analyses (see Fig. 3a) highlight the ubiquitous frequent occurrence of PLs close to a maximum size $\approx 1460B$. This is the expected effect of encapsulation in network-layer packets, that from one hand clamps all packet sizes to a maximum, and on the other hand introduces spurious small-size packets (the remainder of the clamping). To mitigate this "masking effect", we propose and discuss a heuristic for reconstructing the more application-related MS parameter.

For applications using TCP as the transport-layer protocol, messages (defined as a unit of application-layer communication protocol) exchanged between the endpoints are inserted into a stream of bytes, not preserving the application-layer message boundaries. Before encapsulation in network-layer packets, the stream is segmented in chunks of up to MSS bytes[8]. Before encapsulation in network-layer packets, the stream is segmented in chunks of up to *Maximum Segment Size* (MSS) bytes[9]. More precisely, for each chunk (*segment*) the size is calculated as:

$$\mathrm{MSS}_{sgm} \triangleq \min(\mathrm{MSS}_{adv}, \mathrm{MSS}_{lcl}) - \mathrm{OPT}_{\mathrm{IP}} - \mathrm{OPT}_{\mathrm{TCP}} \quad (1)$$

[7]This is defined as the relative frequency of occurrences divided by the width of each bar.

[8]RFC 879: https://rfc-editor.org/rfc/rfc879.txt

[9]RFC 879: https://rfc-editor.org/rfc/rfc879.txt

where MSS_{adv} is the MSS advertised from the other endpoint during the three-way handshake setting up the TCP connection. Differently, MSS_{lcl} is based on the local network interface Maximum Transfer Unit (MTU)—that is advertised to the other endpoint. Finally, $\mathrm{OPT}_{\mathrm{IP}}$ and $\mathrm{OPT}_{\mathrm{TCP}}$ are the sizes of optional fields of IP and TCP headers, respectively. The calculation in Eq. (1) is performed at both endpoints, getting a common upper bound for the connection MSS achievable between them. We highlight that traversed network paths could dynamically reduce this value, possibly asymmetrically: this is managed via the Path MTU Discovery (PMTUD) algorithm[10].

Reverting TCP segmentation to reconstruct the application-layer *messages* would require to infer the TCP status machine: this is far from trivial or exact [38]. Moreover, as TCP does not commit to preserving application-layer message boundaries, not even a perfect inference of the TCP state machine would separate upper-layer Protocol Data Units without payload analysis (prevented by encryption). To solve this problem, we define an MSS-*based message-reconstruction heuristic*, that has to take into account only the MSS of the flow, and the length of the previous segment. The heuristic works under the hypothesis that a packet with transport-layer payload equal to current MSS is always part of a bigger application-layer message: in this case, the segment is joined with the following one; this is repeated until the first smaller-than-MSS segment is received, that is joined with the previous one(s) and marks the end of the application-layer message. Hence, the size of the reconstructed message (i.e. the MS) is the sum of the PL of its joined segments, while the corresponding IMT is defined as the time lag between the first segments of two consecutive messages. The heuristic is reported as pseudocode in Alg. 1.

In addition to the simple heuristic described above, the following conditions have been checked (and measured as negligible in the analyzed dataset). In principle reductions of MSS during the capture are possible, and are continuously

[10]RFC 1191: http://www.rfc-editor.org/rfc/rfc1191.txt

Algorithm 1: MS reconstruction heuristic: remove_opts_size() implements Eq. (1).

```
Input: flow MSS, packet sequence
Output: MS sequence
previous_is_MSS ← False ;
for each packet do
    MSS_sgm ← remove_opts_size(MSS, current_pkt) ;
    PL ← payload_length(current_pkt) ;
    if previous_is_MSS then
        MS ← MS + PL ;
    else
        MS ← PL ;
    end
    if PL = MSS_sgm then
        previous_is_MSS ← True ;
    else
        previous_is_MSS ← False ;
        output MS ;
    end
end
```

detected by PMTUD algorithm[10] by means of ICMP messages: as no `ICMP fragmentation needed and DF set` packets have been found in the traces, no variation of the MSS has been experienced during the capture. Regarding the presence of re-transmitted or out-of-order segments: almost no re-transmissions (≈0.5% of all the packets, ≈0.7% of those with payload) or reorderings (≈0.4% of all the packets, ≈0.6% of those with payload) were detected in the analyzed dataset. Interestingly, re-transmissions and out-of-order segments affect also packet-level analysis, but in the case of MS reconstruction their relative impact is smaller (as the length of a segment constitutes a fraction of whole message size, instead of a full payload length). Therefore if performance constraints discourage online stateful detection of retransmissions and reordering, message-level analysis is more robust to these sources of error at the cost of little computation (3 sums and 3 comparisons) per packet and little memory (a *previous-is-MSS* flag and MSS value) per flow (see Alg. 1).

Another phenomenon potentially affecting message size reconstruction is related to the rare and strongly discouraged[11] `URGENT` flag, signaling out-of-band message inserted in the TCP stream. We checked that it *never* occurred in the whole dataset, and it can be safely assumed[11] to be generally unused in modern and future applications.

Two conditions are excluded from the above analysis: they are described hereafter and their potential effect on MS reconstruction is discussed. We first consider an erroneous joining of different messages. Indeed, an application-layer message could end exactly on the border of an MSS-sized segment: for encrypted protocols this information is effectively or purposely hidden[4]. For uniformly-distributed *actual* message lengths this would happen on average with probability $1/\text{MSS}$ (smaller than 10^{-3} for common values of MSS). Similarly, in case Nagle's algorithm [39] is enabled *and triggered*, the reconstruction heuristics would erroneously join small-sized ($<$MSS) messages. As interactive applications (such as web servers [40]) *disable* Nagle's algorithm to avoid impact on responsiveness, we assume that for the analyzed dataset (focusing on mobile apps) either Nagle's algorithm has been disabled, or it causes negligible performance impact, and consequently has a minor effect on our analysis.

The opposite error (the heuristic failing to join packets belonging to the same message) matches the event that the MSS-sized buffer is not filled up at the moment of packet sending, despite more data from the application (same message) will be available afterwards. We investigate and discuss these additional cases based on experimental data in Sec. V-A.

We explicitly highlight that, although in principle the timing information could be used to inform the message-reconstruction inference, we decided to *not* employ it in our heuristic and instead consider it as a coherence check, to experimentally validate the heuristic. The same consideration has been applied to the TCP `PUSH` flag, whose semantic[12] implies that the sending application signaled TCP that the message is complete (no other data will be immediately provided), and data *should* be sent without waiting for segment filling. The optional nature of this behavior, the lack of relation between the presence of the flag and termination of the message inside the transmitted segment[12], and the possible presence of a *transport sub-layer* of application level (e.g. TLS) can affect the reliability and general validity of this message-termination signal. The choice of not including neither of this additional information in the heuristic has the effect of keeping it as simple as possible, focused on removing the segmentation *masking* effect, with the least assumptions (hence robust to specific case variations) and the most straightforward implementation. Both timing and `PUSH` flag are analyzed experimentally for impact on the heuristic in Sec. V-A.

Trading-off with these characteristics, the proposed heuristic *generally applies* to encrypted traffic, and constitutes an advancement with respect to simpler approaches[13] that have been effectively adopted for traffic classification purposes [35].

C. Modeling & Predicting with (multimodal) Markov Chains

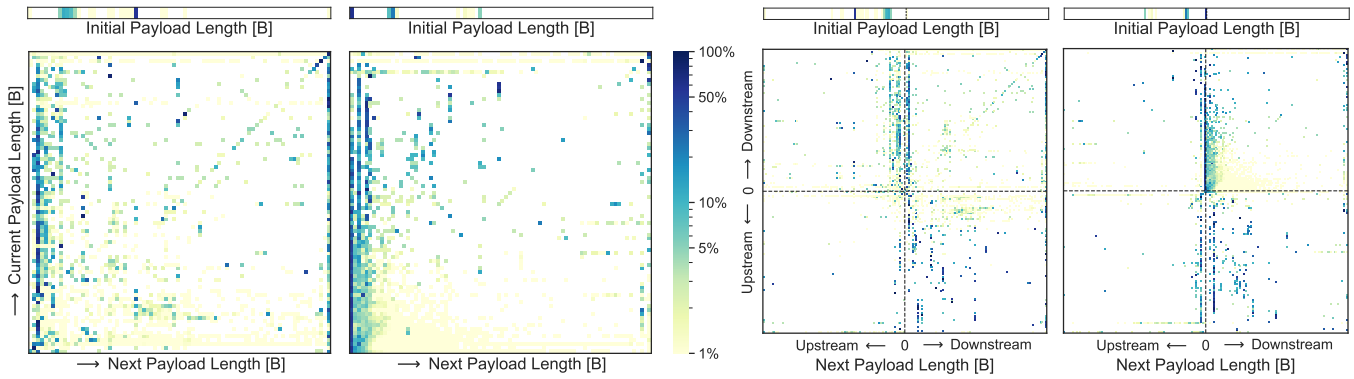
A multimodal Markov Chain is a stochastic model which specifies how an M-dimensional random variable changes over time. Specifically, let $\boldsymbol{x}^n$ be the discrete-valued random variable (i.e. whose m^{th} modality x_m^n can assume only S_m possible values) which describes the state of a system at discrete time $n \in \mathbb{N}_0$. In this paper, we initially refer to stationary first-order Markov Chains which are a type of discrete-time stochastic process satisfying two assumptions: (i) (first-order) *Markov property* $\Pr(\boldsymbol{x}^{n+1}|\boldsymbol{x}^n, \boldsymbol{x}^{n-1}, \ldots) = \Pr(\boldsymbol{x}^{n+1}|\boldsymbol{x}^n)$, namely the probability distribution of the state at time $n+1$ depends only on the state at time n, (ii) *homogeneity* $\Pr(\boldsymbol{x}^{n+1}|\boldsymbol{x}^n) = \Pr(\boldsymbol{x}^n|\boldsymbol{x}^{n-1}), \forall n$, namely the state transition distribution is independent on time n.

Accordingly, a stationary Markov Chain can be equivalently represented through two data structures, i.e. the *transition probability matrix* $\boldsymbol{P} \in [0,1]^{S \times S}$—with $p_{i,j} = \Pr(\boldsymbol{x}^n =$

[11] RFC6093: https://tools.ietf.org/html/rfc6093

[12] RFC 1122: https://tools.ietf.org/html/rfc1122

[13] Specifically, the authors in [35] adopt a fixed MSS of $1460B$, and any sequence of MSS packets is considered as a *single* message, of length MSS, discarding the actual message length. By contrast, we extract actual MSS from the handshake and calculate it for each packet accounting for options in the headers, and sum up all the segment lengths to get the message size.



(a) PL Diretta. (b) PL Slither.io. (c) (PL,DIR) Diretta. (d) (PL,DIR) Slither.io.

Figure 4: Initial distributions ($\boldsymbol{q}^0$) and transition matrices ($\boldsymbol{P}$) based on: (a)-(b) PL, and (c)-(d) (PL,DIR). Increasing (per-direction in (c)-(d)) PLs are shown by arrows.

$\boldsymbol{s}_j | \, \boldsymbol{x}^{n-1} = \boldsymbol{s}_i)$ and $\sum_{j=1}^{S} p_{i,j} = 1$—and the *initial* ($n = 0$) *probability vector* $\boldsymbol{q}^0$—with $q_i^0 = \Pr(\boldsymbol{x}^0 = \boldsymbol{s}_i)$ and $\sum_{i=1}^{S} q_i^0 = 1$—both defined on the Cartesian product of the individual modality spaces, having size $S \triangleq \prod_{m=1}^{M} S_m$. For the sake of a compact notation, in what follows, we define as $\boldsymbol{X} \triangleq \{\boldsymbol{x}^0, \ldots, \boldsymbol{x}^{N-1}\}$ the time series of the vector observation over the whole sequence $n = 0, 1, \ldots, N-1$.

In practical traffic analysis cases, the pair $(\boldsymbol{P}, \boldsymbol{q}^0)$ describing the Markov Chain is *unknown* and it is learned from repeated observations, usually as a training set $\mathcal{D}$ comprising multiple time series of the vector variable, namely $\mathcal{D} \triangleq \bigcup_{t=1}^{T} \boldsymbol{X}_t$. Accordingly, the transition matrix and initial probability vector are usually estimated via a *Maximum-Likelihood* (ML) procedure. This results in the ML estimates $\hat{p}_{i,j} \triangleq \frac{\eta_{i,j}}{\eta_{i,\star}}$ and $\hat{q}_i^0 \triangleq \frac{\eta_i}{\eta}$, respectively. In the former case, $\eta_{i,j}$ is the number of transition pairs $(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}) = (\boldsymbol{s}_j, \boldsymbol{s}_i)$, whereas $\eta_{i,\star}$ is the number of transition pairs such that $(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}) = (\cdot, \boldsymbol{s}_i)$. In the latter case, η_i is the number of instances $\boldsymbol{x}^n = \boldsymbol{s}_i$, whereas η denotes the total number of observations.

In this work we are concerned with *the separate analysis of both* $\boldsymbol{P}$ and $\boldsymbol{q}^0$, which are investigated to highlight different characteristics of mobile traffic. Indeed, regarding $\boldsymbol{P}$, we investigate relevant patterns which can be inferred from the analysis and give an intuitive interpretation. Complementarily, regarding $\boldsymbol{q}^0$, we investigate whether the initial distribution is application-independent (due to a common transport sublayer, e.g. TLS tunneling) or not. In the latter case, this provides room for early discriminability.

For simplicity, we refer to the illustrative cases $M = 1$ and $M = 2$ in Fig. 4 (for two different apps), where x^n and $\boldsymbol{x}^n$ represent the PL and the (PL,DIR) at time n, respectively. The example shows how the resulting matrices change when both PL and DIR (i.e. telling if a packet is either transmitted or received, Figs. 4c and 4d) are taken into account, rather than the simple PL (Figs. 4a and 4b). When multiple modalities are taken into account, the resulting matrix can provide a more detailed view, able to catch the interplay of multiple parameters. In this example, the resulting $\boldsymbol{P}$ and $\boldsymbol{q}^0$ provide a snapshot of the behavior of the application at network level in terms of the payload length of two consecutive packets and of the initial behavior, respectively. In detail, each element in the matrix reports the probability of observing a packet with an L4 payload of a given size (associated to the column) *right after* a packet having a payload of the size associated to the row.

As a result, several patterns may be spotted in the matrices whose interpretation is discussed below (higher values in the matrices are associated to darker color) with reference to PL (same considerations apply to the other modalities):

Horizontal patterns (→) with approximately same values on a whole row (e.g., $\approx 1.25\%$ in the case of 80 bins) highlight that the current PL does not provide any hint about the next one (i.e. a uniform distribution of next-PL).

Dark patterns on the main diagonal (↗) witness applications prone to generate couples of same-PL packets.

Dark vertical patterns (↑) surface highly-likely PLs in the observed traffic, i.e. regardless the current PL.

Darker areas (blocks ■) that span across multiple consecutive rows and columns report applications prone to likely generate couples of packets with PLs falling in a small set of similar values.

The presence (and even the absence) of these patterns provides an intuitive easily-observable *qualitative* snapshot of the characteristics of the traffic generated by a mobile app. Still, Sec. III-D is devoted to exemplify how the considered representation can be used to support a *quantitative* analysis of traffic similarity, based on hypothesis testing tools.

Stationary multimodal Markov Chains can be generalized to an arbitrary order W, at the expenses of a loss of visualization/interpretability. In detail, these models satisfy two assumptions: (i) W^{th}-order *Markov property* $\Pr(\boldsymbol{x}^{n+1} | \boldsymbol{x}^n, \ldots) = \Pr(\boldsymbol{x}^{n+1} | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)})$, namely the probability distribution of the state at time $n+1$ depends only on the more recent W observations; and (ii) *homogeneity* $\Pr(\boldsymbol{x}^{n+1} | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)}) = \Pr(\boldsymbol{x}^n | \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-W}), \forall n$, namely the state transition distribution is independent on n. Accordingly, a stationary Markov Chain can be equivalently represented via the *joint initial* probability distribution $\Pr(\boldsymbol{x}^{W-1}, \ldots, \boldsymbol{x}^0)$ and the *transition probability* distribution $\Pr(\boldsymbol{x}^n | \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-W})$. Such distributions are learnt via ML estimation from the usual

training set $\mathcal{D}$, comprising multiple time series of the vector variable, namely $\mathcal{D} \triangleq \bigcup_{t=1}^{T} \boldsymbol{X}_t$.

Once trained, a W^{th} order stationary Markov Chain can be used for making *predictions* (this also applies for $W = 1$). Namely, for $n = -1, \ldots, W-2$ (first W packets), the algorithm uses the *predictive distribution* $\Pr(\boldsymbol{x}^{n+1}|\boldsymbol{x}^n, \ldots, \boldsymbol{x}^0)$, obtained from the joint initial distribution $\Pr(\boldsymbol{x}^{W-1}, \ldots, \boldsymbol{x}^0)$. Differently, for $n \geq (W-1)$, the transition distribution $\Pr(\boldsymbol{x}^{n+1}|\boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)})$ is employed, that is Markov property allows the use of a sliding window consisting of the last W observations, which are used as inputs of the algorithm. The corresponding prediction $\hat{\boldsymbol{x}}^{n+1}$ can be obtained either via (i) a Maximum a Posteriori (MAP) approach or (ii) a Minimum Mean Square Error (MMSE) approach. In the former case, $\hat{\boldsymbol{x}}^{n+1}_{\text{map}} \triangleq \arg\max_{\boldsymbol{s}_j} \Pr(\boldsymbol{x}^{n+1} = \boldsymbol{s}_j|\boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)})$ (the posterior mode), whereas in the latter case $\hat{\boldsymbol{x}}^{n+1}_{\text{mmse}} \triangleq \sum_{j=1}^{S} \boldsymbol{s}_j \Pr(\boldsymbol{x}^{n+1} = \boldsymbol{s}_j|\boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)})$ (the posterior mean).

D. Traffic Characterization with Markov Chains: homogeneity analysis for versions/apps/categories

Herein, we are concerned with investigating whether the set of TOs (biflows) generated by (a) two versions of the same app, (b) two apps or (c) two app categories generate homogeneous network traffic. This is tantamount to test the hypothesis that two sets of time series, defined as $\mathcal{D}_x \triangleq \bigcup_{t=1}^{T_x} \boldsymbol{X}_t$ and $\mathcal{D}_y \triangleq \bigcup_{t=1}^{T_y} \boldsymbol{Y}_t$, respectively, can be described by either the same Markov model ($\mathcal{H}_0$) or by two different ones ($\mathcal{H}_1$). In mathematical terms, this test can be stated as:

$$\begin{aligned} \mathcal{H}_0 &: \begin{cases} \boldsymbol{X}_t \sim \text{Markov}(\boldsymbol{P}^\star) & t = 1, \ldots, T_x \\ \boldsymbol{Y}_t \sim \text{Markov}(\boldsymbol{P}^\star) & t = 1, \ldots, T_y \end{cases} \\ \mathcal{H}_1 &: \begin{cases} \boldsymbol{X}_t \sim \text{Markov}(\boldsymbol{P}^X) & t = 1, \ldots, T_x \\ \boldsymbol{Y}_t \sim \text{Markov}(\boldsymbol{P}^Y) & t = 1, \ldots, T_y \end{cases} \end{aligned} \tag{2}$$

In the above formulation, we have indicated with $\boldsymbol{P}^\star$ the common transition matrix under the null hypothesis, whereas with $\boldsymbol{P}^X$ (resp. $\boldsymbol{P}^Y$) the transition matrix associated to the set $\mathcal{D}_x$ (resp. $\mathcal{D}_y$) for the hypothesis $\mathcal{H}_1$. Before proceeding, we highlight that for the considered test we do not take into account the initial distribution $\boldsymbol{q}^0$: in practical cases, (a) the observation of the biflow since the first packet is not guaranteed and (b) we aim at assessing heterogeneity in network traffic models at a general view.

A widespread approach to solve the aforementioned problem resorts on the evaluation of the corresponding *empirical probability laws* associated to the Markov Chain pairwise transitions [3]. In other terms, the problem reduces to testing whether the two sets can be generated by a common probability law or are explained by two different probability laws $p_x(\boldsymbol{x}^n, \boldsymbol{x}^{n-1})$ and $p_y(\boldsymbol{y}^n, \boldsymbol{y}^{n-1})$. These probability laws can be equivalently described through $S \times S$ matrices $\boldsymbol{\Pi}^x$ and $\boldsymbol{\Pi}^y$, i.e. similarly as the transition matrices (indeed each $\boldsymbol{\Pi}$ is in one-to-one mapping with the corresponding $\boldsymbol{P}$). Additionally, since these probability laws are *unknown*, the actual values are replaced by the corresponding ML estimates $\hat{\boldsymbol{\Pi}}^x$ and $\hat{\boldsymbol{\Pi}}^y$, with $(i,j)^{th}$ entry equal to $\hat{\pi}^x_{i,j} \triangleq \frac{\eta^x_{i,j}}{\eta^x}$ and $\hat{\pi}^y_{i,j} \triangleq \frac{\eta^y_{i,j}}{\eta^y}$, respectively.

Accordingly, the following decision statistic is employed herein [41]:

$$H\left(\hat{\boldsymbol{\Pi}}^x, \hat{\boldsymbol{\Pi}}^y\right) \underset{\hat{\mathcal{H}}=\mathcal{H}_0}{\overset{\hat{\mathcal{H}}=\mathcal{H}_1}{\gtrless}} \gamma \tag{3}$$

where $H(\cdot,\cdot)$ denotes the *Hellinger distance measure* between discrete probability distributions, which in our case equals:

$$H\left(\hat{\boldsymbol{\Pi}}^x, \hat{\boldsymbol{\Pi}}^y\right) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{S}\sum_{j=1}^{S} (\sqrt{\hat{\pi}^x_{i,j}} - \sqrt{\hat{\pi}^y_{i,j}})^2} \tag{4}$$

The Hellinger distance is a particular type of f-divergence, and the test based on this statistic shares the same asymptotic optimality as those Hoeffding-originated [3] or based on alternative divergence measures [41]. Nonetheless, its practical advantages are that (a) it is well-defined even in case of zero-probability bins and (b) it is bounded as $0 \leq H(\cdot,\cdot) \leq 1$. We remark that Eq. (3) implicitly assumes the same number of transition pairs for evaluating $\hat{\boldsymbol{\Pi}}^x$ and $\hat{\boldsymbol{\Pi}}^y$. In Sec. V-C, we will show how this restriction will be accounted.

The performance of the above test is assessed via the well-known True Positive Rate (TPR, i.e. $\Pr(\hat{\mathcal{H}}_1|\mathcal{H}_1)$) and False Positive Rate (FPR, i.e. $\Pr(\hat{\mathcal{H}}_1|\mathcal{H}_0)$), both monotonically-decreasing with γ. Accordingly, we focus on a Neyman-Pearson setup: we assess the TPR achieved by the test subjected to a (small) prescribed FPR (i.e. $\gamma : \Pr(\hat{\mathcal{H}}_1|\mathcal{H}_0) = \alpha$).

To obtain a reasonable estimate of the TPR subjected to the prescribed FPR α, the following *data-driven* procedure is pursued herein. In detail, we first obtain the empirical CDF of the Hellinger statistic under $\mathcal{H}_0$ by randomly under-sampling two equal-sized subsets (a) F_0 times from $\mathcal{D}_x$, and (b) F_0 times from $\mathcal{D}_y$ and evaluating $H(\cdot,\cdot)$ each time. Then, the aforementioned CDF (based on $2F_0$ samples) is exploited to set γ so as to satisfy $\Pr(\hat{\mathcal{H}}_1|\mathcal{H}_0) = \alpha$. Once γ has been obtained, two equal-sized subsets are randomly under-sampled from $\mathcal{D}_x$ and $\mathcal{D}_y$, respectively, for F_1 times, to obtain the empirical CDF of $H(\cdot,\cdot)$ under $\mathcal{H}_1$. Accordingly, the TPR is obtained as the fraction of times the statistic exceeds γ.

Remarks: The computation of the Hellinger distance in Eq. (4) relies on the preliminary phase of histogram quantization (described in Sec. III-A) of `PLs` and `IATs` (resp. `MSs` and `IMTs`), to construct the aforementioned Markov Chains. Hence, the distance-computation falls within the well-known class of "plugin estimators" of f-divergences [42]. This class of approaches ensures estimation consistency under mild conditions while retaining simplicity and interpretability in their use. The only prerequisite is that *accurate Markov Chain estimates* can be obtained. Hence, though the quantization operation should aim at negligible loss, there should be enough number of samples to estimate the considered model at the chosen granularity (viz. number of bins). For this reason, in Sec. III-A, we have selected the number of bins so as to limit the above issue.

A graphical depiction of the above procedure (and of the corresponding result) is shown in Fig. 5, referring to homogeneity testing of network traffic generated by consecutive versions of the same app (scenario (a) among those introduced at the beginning of this section). The x-axis reports the

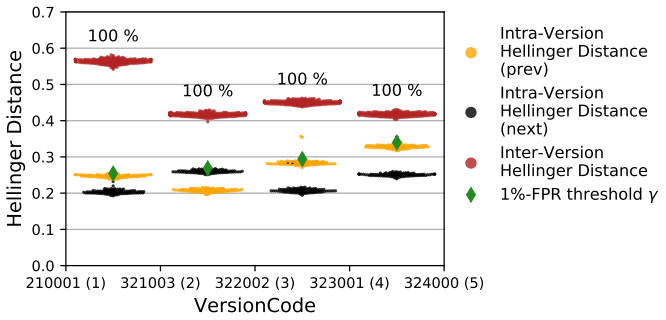


Figure 5: Homogeneity testing between consecutive versions of `Subito`. The percentage above inter-version Hellinger distance is the TPR.

identifier of versions of the app under analysis; black points and yellow points mark the *intra-version distance* (i.e. under $\mathcal{H}_0$) calculated among random subsamples of previous and next versions, respectively. Based on the above procedure, both these sample groups are exploited to obtain the *threshold level* γ corresponding to the desired FPR ($\alpha = 0.01$), marked with a green diamond. Differently, red dots highlight the statistic realizations with subsamples from the set comparing previous and next versions, namely the *inter-version distance*. The numerical value on red points reports the *TPR*: a value close to 100% shows that for (almost) all the subsamples the distance is higher than the reference γ, i.e. there is very high confidence the models associated to the two versions differ, whereas a value close to 0% means that the two versions share the same Markov model, i.e. there is very high confidence in rejecting $\mathcal{H}_1$. Finally, it is worth noticing that, even in the case the TPR equals 100%, we may discern different degrees of dissimilarity based on how distant the red points are from γ, i.e. low-dissimilarity and high-dissimilarity pairs.

E. Modeling and Predicting with Hidden Markov Models

HMMs are structured probabilistic models that form a probability distribution of the time series $\boldsymbol{x}^0, \boldsymbol{x}^1, \ldots, \boldsymbol{x}^n$ described by (i) the hidden-state variable (h^n), that is modeled through a hidden (not observable) Markov Chain, (ii) and the observable variable ($\boldsymbol{x}^n$), that depends *uniquely* on the hidden state. In other words, HMMs are generative models in which the joint distribution of observations and hidden states is modeled.

An HMM is described by the complete set of parameters $\lambda \triangleq \{\boldsymbol{q}^0, \boldsymbol{P}, \boldsymbol{\theta}\}$, whose meaning is explained as follows:

- the initial state distribution $\boldsymbol{q}^0$, whose i^{th} entry is defined as $q_i^0 \triangleq \Pr(h^0 = s_i)$;
- the state transition matrix $\boldsymbol{P}$, whose $(i,j)^{th}$ entry equals $p_{i,j} \triangleq \Pr(h^n = s_j | h^{n-1} = s_i)$;
- the vector $\boldsymbol{\theta}$ collecting all the parameters completely specifying the emission PDF/PMF conditioned on each state, namely $p(\boldsymbol{x}^n | h^n = s_i)$, for $i = 1, \ldots, S_\text{h}$.

First of all, a *training* phase is usually required (based on a dataset of sequences $\mathcal{D}_x \triangleq \bigcup_{t=1}^{T_x} \boldsymbol{X}_t$) to learn the best parameters λ of the HMM describing $\mathcal{D}_x$. Such parameters are those maximizing the "likelihood" of the model $\hat{\lambda} \triangleq \arg\max_\lambda \sum_{t=1}^{T} p(\boldsymbol{X}_t; \lambda)$. This problem is solved with the *Baum-Welch* algorithm, a special case of the expectation-maximization algorithm.[14] Once trained, HMMs can be used for prediction (e.g., to predict the next sample $\boldsymbol{x}^{n+1}$) according to the trained model.

In this case, the *prediction task* can be accomplished by evaluating the predictive distribution $\Pr(\boldsymbol{x}^{n+1} | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W_\text{h}-1)})$ based on a sliding window W_h and calculating the MMSE estimator (namely, its mean) as $\hat{\boldsymbol{x}}^{n+1} \triangleq \mathbb{E}\{\boldsymbol{x}^{n+1} | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W_\text{h}-1)}\}$. The explicit expression of MMSE estimator is $\hat{\boldsymbol{x}}^{n+1} = \sum_{i=1}^{S_{HM}} \boldsymbol{\mu}_i \Pr(h^{n+1} = s_i | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W_\text{h}-1)})$. The left term in the sum is simply the mean of the emission PDF/PMF conditioned on the i^{th} state s_i (i.e. $p(\boldsymbol{x}^n | h^n = s_i)$). Differently, the right term is obtained by using (i) the *forward-backward* algorithm (which allows for obtaining $\Pr(h^n | \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W_\text{h}-1)})$) and (ii) the transition matrix $\boldsymbol{P}$.

Remarkably, the HMMs admit also an interesting segmentation interpretation, which is obtained leveraging the *Viterbi algorithm* for the *reconstruction* of the most probable sequence of hidden states for a given sequence of N observations, namely $(\hat{h}^1, \ldots, \hat{h}^N) = \arg\max_{(h^1,\ldots,h^N)} P(h^1, \ldots, h^N | \boldsymbol{x}^1, \ldots \boldsymbol{x}^N)$ (i.e. the so-called Viterbi path).

IV. Dataset Description

In our experimental evaluation, we employ the human-generated MIRAGE-2019 dataset [11] collected in the AR-CLAB laboratory at the University of Napoli "Federico II" during '17-'19 using three Android (6.0.1) devices, namely: (i) Xiaomi Mi5, (ii) Google Nexus 7, and (iii) Samsung Galaxy A5. Overall, MIRAGE-2019 gathers the traffic generated by 40 Android apps belonging to 16 different categories. $280+$ experimenters have contributed to dataset construction by mimicking the common usage of apps with the aim of exploring their different functionalities (e.g., login, registration, common activities, etc.). In detail, they carried out several capture sessions of $5 \div 10$ mins, each resulting in a PCAP traffic trace and additional system log-files with ground-truth information. As a whole, $4600+$ PCAP traces were collected within MIRAGE-2019.

Based on ground-truth metadata, each biflow is reliably labeled with the corresponding Android package-name that matches the 5-tuple in the system log-file, considering network-related system calls. In detail, we associate to each socket descriptor—encompassing *<IP:port>* pairs—the name of the Android package which originates the call. Furthermore, for each mobile app, MIRAGE-2019 provides a finer-grain label (*VersionCode*), that is monotonically increasing over releases[15]: each traffic capture session in the dataset has been performed with the up-to-date version of the app [11]. This labeling allows us to investigate the characterization power of the proposed Markov-based homogeneity testing

[14] Since the expectation-maximization is an iterative optimization approach, a number of different "restarts" R_h are usually employed to favour convergence toward the global optimum of the likelihood.

[15] https://developer.android.com/studio/publish/versioning

(and, in negative case, the degree of dissimilarity) at the *finest granularity* available in a public dataset (see later Sec. V-C).

In the next analyses, for each biflow, we leverage a subset of per-packet data provided in MIRAGE-2019, corresponding to PL, DIR, and IAT, along with per-flow metadata related to the complete biflow and upstream/downstream flows. In detail, we focus on mobile applications whose traffic is collected using the Mi5 device. Given the variety and the high number of mobile apps present in MIRAGE-2019, for each following analysis we report results pertaining only to subsets of the considered dataset for both reasons of brevity and to highlight the relevant trends.

V. RESULTS AND DISCUSSION

First, we evaluate the proposed message-reconstruction heuristic in Sec. V-A, while in Sec. V-B we investigate the Markov-based modeling for PLs and MSs. Then, in Sec. V-C, we assess the network traffic similarity based on the homogeneity testing procedure introduced in the previous section. Differently, in Sec. V-D we analyze the prediction capabilities of HMMs and (high-order) Markov Chains and compare them with baseline and ML approaches. Finally, in Sec. V-E we provide an interpretability analysis of ML techniques by means of Markov-based modeling.

A. Message-reconstruction heuristic evaluation

As the message-reconstruction heuristic has been defined according to the L4-payload length only, here we evaluate how this heuristic is supported by information on timing and by the presence of the PUSH flag. In detail, we focus on the following *five* mobile applications: Facebook, Messenger, Spotify, Subito, and Google Play Store. We have chosen these apps based on their representativeness in terms of number of samples (≈30k biflows, representing ≈30% of overall Mi5 biflows constituting MIRAGE-2019 dataset) and categories (i.e. *social*, *communication*, *music and audio*, *lifestyle*, and *app market*, respectively).

Regarding timing, we define two metrics: *Intra-Message Gap* (IntraMG), i.e. the inter-arrival time between consecutive packets belonging to the *same message*; *Inter-Message Gap* (InterMG), i.e. the inter-arrival time between the last packet of a message, and the first packet of the following message (approximating the silence time between two *consecutive messages*). If the heuristic is correct, the average time between packets *inside* a message should be smaller than the one *between messages*. Looking at Fig. 6 it is evident that this is actually the case, with an order of magnitude of difference between the respective means (dashed lines), 95^{th} percentiles, and maxima.

Considering the PUSH flag, if the "end of message" signal was deterministically set by the application and *always* honored by TCP (neither of the two are guaranteed), the heuristic being correct implies that the packets carrying the flag should be all and only the ones terminating the message. Indeed, only 1.46% of messages shows the flag set in packets different from the ending one. Conversely, the 87.79% of all packets with the PUSH flag occurs at the end of a message. This suggest that the heuristic is both in good accord and *conservative* in joining packets with respect to the PUSH flag presence.

As both these evaluations are assessed independently from the heuristic definition, we conclude that our heuristic, albeit based on a single metric, is strongly coherent with other related metrics and constitutes a sensible and practical approximation of the actual message size (which is not theoretically attainable from transport-layer observation).

B. Markov-based mobile app analysis: packets vs. messages

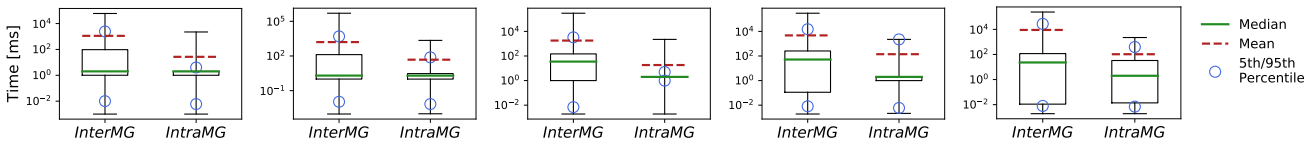
Figures 7 and 8 show the initial distribution $\boldsymbol{q}^0$ and the transition matrix $\boldsymbol{P}$ (cf. Sec. III-C) for PL and MS, respectively, for the same five apps considered in Sec. V-A.

Inspecting the results of this analysis, we can notice a substantial difference in the visual outcome for different apps. First, the initial distributions of PLs are way scattered than those of MSs. This points out that message reconstruction has no significant impact on the first packets of biflows, as the latter are usually involved in application-level negotiation/signaling (e.g., SSL/TLS handshake) and thus are not segments of bigger messages. Going into details of Fig. 7, we can notice that PL matrices for Facebook, Messenger, and Spotify report more evident patterns on the main diagonal ↗ than those of Subito and Google Play Store. Also, while Facebook and Messenger show darker vertical patterns for smaller next PL values (both between 20 B and 50 B) Spotify and Subito result in more scattered values on the matrices, especially for smaller (next) PL values.

Moving to MS matrices, Fig. 8 witnesses how message reconstruction is able to unveil further peculiarities not spotted in the corresponding PL matrices. For instance, PL matrix for Facebook (cf. Fig. 7a) presents two dark vertical patterns for both small and big PLs, and a lighter pattern on the main diagonal ↗. Differently, for the MS (cf. Fig. 8a) we can observe a single dark vertical block at the (next) MS-size range of 1–800 B spread on a large number of (current) MS values (up to 86 KB), and a lighter main diagonal pattern ↗. The appeal of modeling the MS is also confirmed when comparing apps that show similar patterns for the PL transition matrices: although Facebook and Messenger traffic exhibits a similar behavior when modeled considering PLs, these apps unveil different patterns when leveraging MSs, thus witnessing the discriminating power obtained with MS reconstruction.

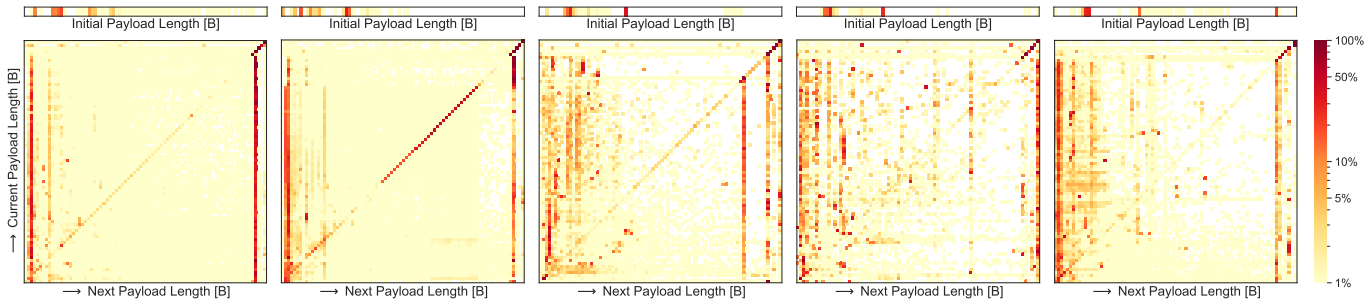
Comparing Figs. 7 and 8, it is also generally evident that packets with small PLs, are often parts of larger messages. For instance, Subito and Google Play Store both have a scattered behavior for the PL (cf. Figs. 7d and 7e), but exhibit an evident effect of the reconstruction when considering the MS (cf. Figs. 8d and 8e). Finally, in some cases we can notice slightly similar patterns for PLs (cf. Fig. 7c) and MSs (cf. Fig. 8c) generated by Spotify (i.e. dark vertical patterns for small and large sizes, and a lighter main diagonal ↗) but—similarly to the other apps—narrower vertical patterns due to merging of smaller packets into messages.

A notable phenomenon can be spotted in both Figs. 8a and 8b: the presence of diagonal patterns running parallel to the anti-diagonal ↘. Each of such lines represents *(current, next)* MS-pairs that sum up to a constant value (with a



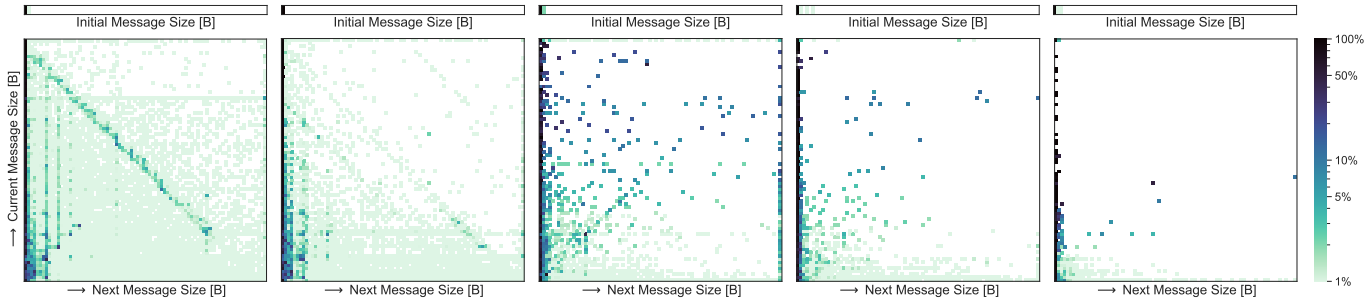
(a) Facebook (b) Messenger (c) Spotify (d) Subito (e) Google Play Store

Figure 6: Comparison between *Inter-Message Gap* (InterMG) and *Intra-Message Gap* (IntraMG) distributions.



(a) Facebook. (b) Messenger. (c) Spotify. (d) Subito. (e) Google Play Store.

Figure 7: Initial distributions $\boldsymbol{q}^0$ and transition matrices $\boldsymbol{P}$ of selected apps for the PL. Binning refers to the quantization depicted in Fig. 3a. Increasing PLs are shown by arrows.



(a) Facebook. (b) Messenger. (c) Spotify. (d) Subito. (e) Google Play Store.

Figure 8: Initial distributions $\boldsymbol{q}^0$ and transition matrices $\boldsymbol{P}$ of selected apps for the MS. Binning refers to the quantization depicted in Fig. 3b. Increasing MSs are shown by arrows.

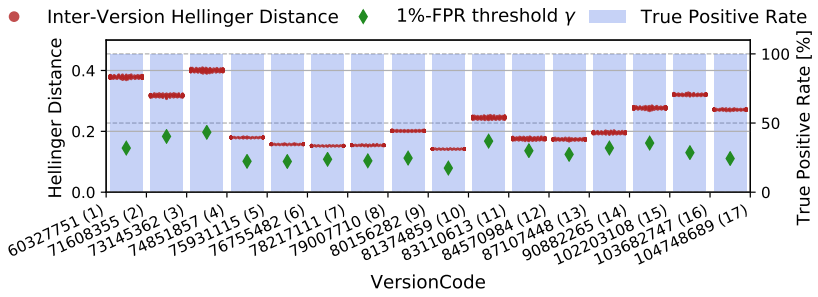
maximum variation of about 5 bins). To estimate the impact of such a phenomenon, we have considered the relative frequencies of *sums* of the size of each pair of consecutive messages for Facebook, and found a small peak (regarding the $\approx 2.29\%$ of messages) for bins ranging from 79kB to 85kB. We investigated whether a fraction of these could be due to the heuristic failing to join packets belonging to a same (relatively-common sized) message. Indeed, we compared the timing- and PUSH-based evaluation (cf. previous Sec. V-A) with and without the joining, and found negligible differences (less than 18ms for the average InterMG, and 0.46 for the percentage of messages not ending with a PUSH-flagged packet). We derive that either it is not an artifact, or neither timing nor the PUSH flag would help in removing it (no other method being available). Therefore, we consider these anti-diagonal patterns in the characterization of the network behavior of the application, at the message granularity level, also noting that this behavior is not universal, thus it actually helps differentiating applications.

Overall, this investigation (together with that related to other apps, not shown for brevity) testifies that message-based analysis performed leveraging our MSS-based reconstruction heuristic, is able to provide a *complementary view* of mobile app traffic with respect to PL-based characterization. Therefore, we consider both packets and messages as the relevant network traffic atoms in the following analyses.

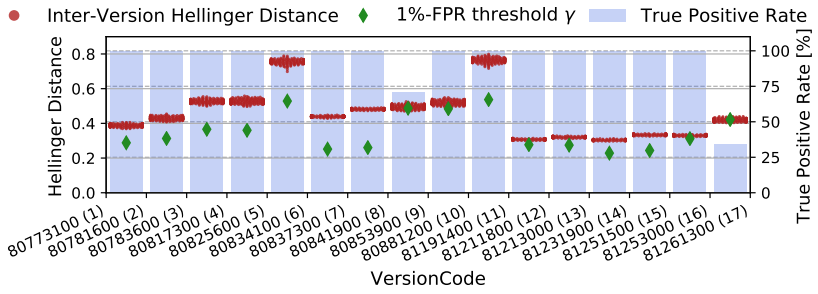
C. Homogeneity Analysis via Markov-based testing

In this section, we apply the Markov-based homogeneity test detailed in Sec. III-D to investigate the change of network behavior within mobile apps, at different granularities (i.e. version, app, category). To this end, Fig. 9 reports the results of the testing procedure performed on first-order multi-modal Markov Chains modeling (PL, DIR). Specifically, we focus on testing traffic homogeneity between two consecutive versions of the same app (cf. Figs. 9a-b), pairwise apps (cf. Fig. 9c), and pairwise categories (cf. Fig. 9d).

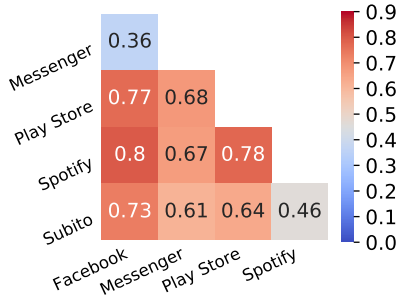
Going into details, Figs. 9a and 9b report two instances of the homogeneity test result described in Fig. 5 related to the Facebook and Google Play Store, respectively.



(a) Homogeneity testing between consecutive versions of `Facebook`.



(b) Homogeneity testing between consecutive versions of `Google Play Store`.



(c) Inter-app dissimilarity index.

(d) Inter-category dissimilarity index.

Figure 9: Homogeneity analysis of `(PL,DIR)` sequences via Markov-based testing: (a, b) versions, (c) applications, and (d) categories. Dissimilarity index is the difference between median inter-category/-app distance and the 1%-FPR threshold γ.

The reason for our choice is that the above apps are those with the higher number of versions within MIRAGE-2019, i.e. 17. For the sake of readability, for each pair of consecutive versions, we report the threshold γ corresponding to the desired 1% FPR (viz. $\alpha = 0.01$, obtained with $F_0 = 100$) and compare it with the different realizations (red dots) of the inter-version distance statistic ($F_1 = 100$). We can notice that for `Facebook` the TPR equals 100% in all the cases (i.e. the considered statistic always exceeds the 1%-FPR threshold), highlighting the (statistically-recognizable) heterogeneity of the traffic behavior between consecutive versions. On the other hand, in Fig. 9b, the result of the test highlights a more varied outcome when comparing consecutive versions of `Google Play Store`. Indeed, in two cases, the homogeneity test results in TPR<100%, namely 71% and 34% for the version pairs 8–9 and 16–17, respectively. The latter case denotes very-likely traffic homogeneity in the corresponding version pair. Moreover, the distance statistic exhibits a higher variance for certain pairs (e.g., 5–6, 9–10, and 10–11). When extending the analysis to non-consecutive versions, i.e. testing traffic heterogeneity between all the version pairs, results (not shown for brevity) report a clear general trend which highlights higher heterogeneity for further-apart versions.

Lastly, we have conducted similar analyses for each pair of considered apps and category. In the former case, we have employed the same five apps as the analyses in Secs.V-A and V-B. In the latter case, since the categories *music and audio*, *lifestyle*, and *app market* encompass only one app (i.e. `Spotify`, `Subito`, and `Google Play Store`, respectively), for this analysis we have selected the five categories with the highest number of apps in MIRAGE-2019: `Social`, `Communication`, `Productivity`, `Shopping`, and `Travel & Local`.

By looking at the numerical evidence, we find that for each pair, the TPR always equals 100% when subjected to a 1%-FPR threshold, pointing out the *clear heterogeneity* between the traffic models of different apps/categories. Accordingly, we have performed a more detailed investigation, reporting in Fig. 9c (resp. Fig 9d) the difference between the median inter-app (resp. inter-category) distance and the related 1%-FPR threshold. Such difference constitutes a *dissimilarity index*: lower values unveil a low degree of heterogeneity between traffic models (being however statistically different given TPR=100%) and vice-versa. As expected, the apps sharing common services and development platform (e.g., `Facebook` and `Messenger`) show the lowest dissimilarity (0.36). Instead, `Google Play Store` proves its unique nature (it manages updates checks and downloads of the other apps), as it shows high heterogeneity when compared to all the other apps—showing a dissimilarity index $\in (0.64, 0.78)$. Interestingly, `Spotify` exhibits a traffic profile not very different from `Subito` (a shopping app), while its traffic is much more different from that of `Facebook`, despite being both multimedia apps.

With regards to categories, it can be noted that `Productivity`, encompassing mostly cloud storage apps, presents the highest dissimilarity index (up to 0.82) with respect to all the other categories. Conversely, `Social` and `Communication` show alike network behaviors. We can speculate that the apps belonging to these categories have several (third-party) services and functionalities (e.g., text/voice

chat, content sharing, etc.) in common. Similar remarks apply for `Travel&Local` and `Shopping` that contain apps (e.g., Booking and Groupon, respectively) sharing features (e.g., image viewer, maps, etc.).

D. Prediction via HMMs and high-order Markov Chains

In this section we provide selected examples of traffic modeling aimed at *prediction* of mobile application traffic, leveraging both kinds of Markov models (i.e. HMMs and Markov Chains). In the following, all results are shown for a single app (`Spotify`) in order to have a coherent focus for comparisons. Similar analyses and results have been obtained for other apps, which are shown in aggregated form at the end of this section for the sake of brevity and completeness.

We evaluate the prediction performance of `PL`, `MS`, `IAT`, and `IMT` in terms of $\mathrm{RMSE}_m \triangleq \sqrt{\frac{1}{N_{\mathrm{pred}}}\sum_{t=1}^{T}\sum_{n=0}^{N_t-1}\left[\hat{x}_m^t(n+1)-x_m^t(n+1)\right]^2}$, where $N_{\mathrm{pred}} \triangleq \sum_{t=1}^{T} N_t$ denotes the no. of predictions, $x_m^t(n)$ the sequence of values of the m^{th} feature observed from the t^{th} biflow, and $\hat{x}_m^t(n)$ the corresponding sequence of values provided by the prediction model. Differently, we use the well-known G-mean as a compact measure to assess predictive capabilities of the (binary-valued) `DIR`, namely $\text{G-mean} \triangleq \sqrt{p_{\mathrm{dir}}^{\mathrm{dw}}\, p_{\mathrm{dir}}^{\mathrm{up}}}$, where $p_{\mathrm{dir}}^{\mathrm{dw}} \triangleq \Pr(\hat{\mathrm{x}}_{\mathrm{dir}}(n+1)=\mathrm{DW}\,|\,\mathrm{x}_{\mathrm{dir}}(n+1)=\mathrm{DW})$ and $p_{\mathrm{dir}}^{\mathrm{up}} \triangleq \Pr(\hat{\mathrm{x}}_{\mathrm{dir}}(n+1)=\mathrm{UP}\,|\,\mathrm{x}_{\mathrm{dir}}(n+1)=\mathrm{UP})$. As a relevant element of comparison, we report in all the results the performance of the (naive) *baseline*, whose predictions are $\hat{\boldsymbol{x}}_{\mathrm{base}}^{n+1} \triangleq \boldsymbol{x}^n$ (i.e. the next observation value is predicted with the current one).

Finally, for each prediction analysis, our evaluation is based on a *ten-fold cross-validation*, representing a stable performance evaluation setup. This allows (for each fold) to use a subset of the biflows belonging to a given app for training and the rest for evaluation purposes. Accordingly, we report both the mean and the variance (in the form $\mu \pm \sigma$) of each performance measure as a result of the evaluation on the ten different folds. We highlight that the common prediction setup performed using part of the already-observed time series to update the model (or to learn a time-series-specific one) would likely result in better performance metrics. Despite this, the real-world application of such procedure is infeasible in practice due to real-time contexts and complexity constraints associated to the fine-grained prediction task considered: this motivated the biflow-based cross-validation granularity choice we adopted in this analysis.

The application of **(Gaussian) HMMs**[16] to prediction is considered first. The family of HMMs provides a powerful tool to interpret the modeled behavior in terms of *hidden states* (here inferred via the Viterbi algorithm) that evolve accordingly to the sequence of inputs.

To this end, in Fig. 10 we show an example of such *segmentation* behavior, superimposing the observed values (for `DIR`, `PL`, and `IAT`) to the inferred status of the app (background color). The example corresponds to a biflow *not belonging* to the set used for training the model for a certain fold.[17] When a small number of hidden states is used (e.g., $S_{\mathrm{h}} = 3$, as in Figs. 10a, 10b, and 10c) some macro-behaviors associated to the observations are clearly visible (e.g., the sudden change after packet 30, highlighted by the transition from red to yellow). This visibility into the inner working of the trained model allows for informed refinements of the model (e.g., by varying the number of states—a tuning parameter of HMMs) and for exploring hypotheses on the nature of the modeled app (e.g., single purpose versus multi-modal usage). A high number of states may be necessary to characterize the modeled traffic for some specific aim: in Figs. 10d, 10e, and 10f the same example biflow is segmented with $S_{\mathrm{h}} = 57$ states, for comparison. A possible application of such model is for traffic prediction purposes: in this case the optimal number of states minimizes the prediction error. We show in Fig. 11 an example of the analysis varying the number of (hidden) states $S_{\mathrm{h}} \in \{3,\ldots,60\}$, with $W_{\mathrm{h}} = 5$ (the results have highlighted no significant difference with longer window values). Experimental results witness that, compared to the baseline, `IAT` can be better predicted already with the minimum number of considered states ($S_{\mathrm{h}} = 3$); for `PL`, $S_{\mathrm{h}} = 6$ states are sufficient for obtaining a model which performs better than the baseline (-8 bytes in terms of RMSE), while at least $S_{\mathrm{h}} = 18$ states are necessary for improving the baseline in terms of prediction of next-packet direction (`DIR`). The general trend shows that the higher the number of states the better the performance, up to 57 states, where a maximum is hit as the overfitting effect starts showing.

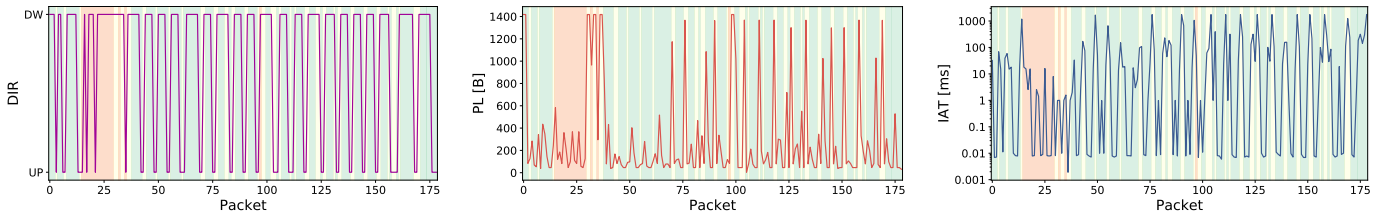
To end our analysis, we also report the performance of the prediction when performed with (high-order) **Markov Chains (MCs)**. For this family of models, the main tuning parameter is the *order* of the chain W (also coinciding with the length of the sliding window). Accordingly, Figs. 12a, 12b, and 12c

Table I: Prediction performance summary for the Markovian approaches (MC MMSE, MC MAP, and HMM), ML approaches (RFR, k-NNR, and LR), and Baseline for `Spotify`. The **best performance** for each metric—for both packets (first group) and messages (second group)—is highlighted in boldface. Results are in the format *avg. (± std.)* obtained over the 10 folds.

		S_{h}/W	G-mean DIR	RMSE PL [B]	RMSE IAT [ms]
Packets	*MC MMSE*	3	**0.90 (±0.01)**	**173 (±30)**	113.72 (±22.33)
	MC MAP	10	0.87 (±0.01)	202 (±35)	135.11 (±34.50)
	HMM	57	0.78 (±0.06)	242 (±30)	124.24 (±15.96)
	RFR	3	0.89 (±0.01)	175 (±29)	**104.59 (±20.53)**
	k-NNR	5	0.84 (±0.09)	181 (±31)	114.19 (±23.34)
	LR	10	0.54 (±0.04)	242 (±43)	138.64 (±28.14)
	Baseline	N/A	0.49 (±0.03)	286 (±52)	186.00 (±37.63)
		S_{h}/W	G-mean DIR	RMSE MS [kB]	RMSE IMT [s]
Messages	*MC MMSE*	10	**0.85 (±0.01)**	12 (±1)	4.19 (±0.31)
	MC MAP	10	**0.85 (±0.01)**	13 (±1)	4.50 (±0.31)
	HMM	60	0.20 (±0.18)	14 (±2)	4.43 (±0.53)
	RFR	3	0.80 (±0.01)	**10 (±2)**	**3.83 (±0.40)**
	k-NNR	3	0.79 (±0.08)	11 (±2)	4.07 (±0.34)
	LR	10	0.24 (±0.02)	12 (±2)	4.42 (±0.60)
	Baseline	N/A	0.27 (±0.04)	20 (±3)	5.65 (±0.46)

[16] In other terms, the emission PDF is Gaussian and the vector $\boldsymbol{\theta}$ collects the mean vectors and the covariance matrices corresponding to all the states.

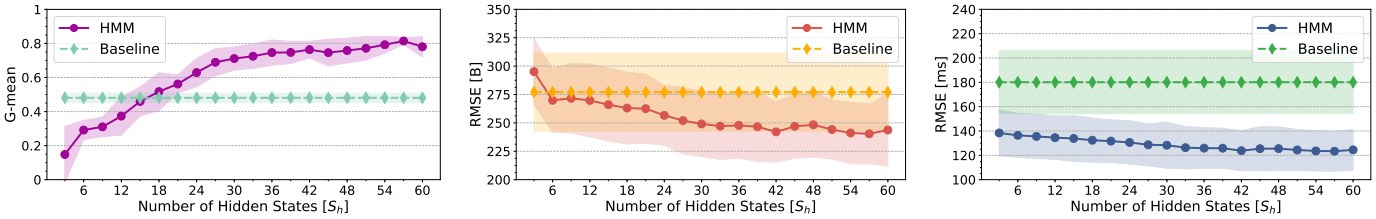
[17] In our analysis, we have considered $R_{\mathrm{h}} = 10$ restarts for training each HMM model.



(a) DIR ($S_\mathrm{h} = 3$ states). (b) PL ($S_\mathrm{h} = 3$ states). (c) IAT ($S_\mathrm{h} = 3$ states).

(d) DIR ($S_\mathrm{h} = 57$ states). (e) PL ($S_\mathrm{h} = 57$ states). (f) IAT ($S_\mathrm{h} = 57$ states).

Figure 10: DIR, PL, and IAT time-series of a biflow generated by Spotify and corresponding most-likely sequence of states (background color, repeated for the three metrics) obtained using the Viterbi algorithm: $S_\mathrm{h} = 3$ states in (a,b,c), $S_\mathrm{h} = 57$ states in (d,e,f).



(a) G-mean DIR. (b) RMSE PL. (c) RMSE IAT.

Figure 11: HMM prediction performance vs. number of states S_h, on Spotify traffic. Values are shown as *avg.* $\pm$ *std.* over 10 folds.

show the related experimental results in terms of G-mean (for DIR) and RMSE (for PL and IAT) when varying the order $W \in \{1, 3, 5, 10\}$ and considering both the MAP and MMSE prediction approaches. For completeness we also report the baseline, which is always over-performed by *both* approaches even when $W = 1$ is chosen. It can be noticed that moving from $W = 1$ to $W = 3$ provides an absolute performance gain for predictions of DIR (up to $+0.05$ of G-mean), PL (up to -60 bytes in terms of RMSE), and IAT (up to -13.59 milliseconds in terms of RMSE).

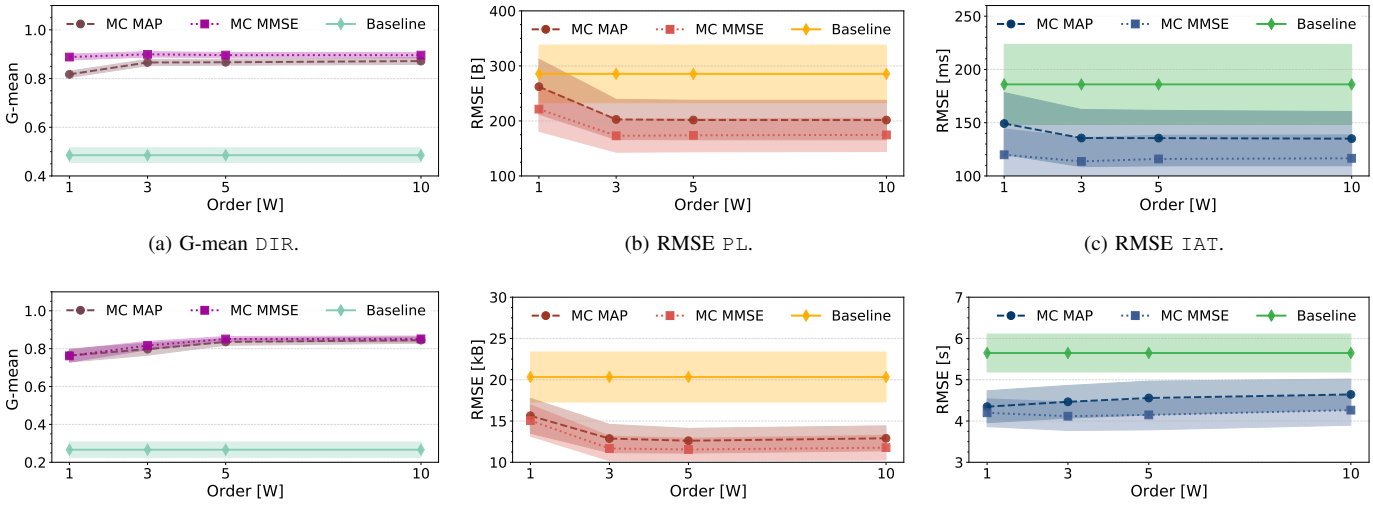
MMSE prediction strategy results in consistently better performance with respect to MAP, due to finer-exploitation of the predictive distribution. Overall, both outperform the reference baseline, proving their practical applicability for the prediction of (encrypted) mobile traffic at packet level. Analogous observations about performance trends when varying the order W can be derived also for message-based analysis, where the RMSE drops by up to -3 kilobytes when passing from $W = 1$ to $W = 3$ (cf. Fig. 12e), while no significant change for IMT is observed (cf. Fig. 12f). Notably, the G-mean for message DIR increases up to $W = 5$, with an absolute performance gain of up to $+0.09$ w.r.t. $W = 1$ (cf. Fig. 12d).

Finally, we have further evaluated the performance obtained by Markov-based approaches (MCs with both MMSE and MAP prediction strategies as well as HMMs) compared against the set of state-of-the-art ML-based regressors introduced in Sec. II-C, namely k-Nearest Neighbors Regressor (k-NNR), Linear Regressor (LR), and Random Forest Regressor (RFR). We considered the feature DIR as the most important variable to be predicted: indeed, even a perfect timing and size prediction, but with the wrong predicted direction, would completely fail most (if not all) of the practical uses of the prediction (resource management, security, accounting, etc.). For instance, router queues are per-interface and therefore the expected resource requirements have to be defined based on the direction of the data streams and related traffic elements. Consequently, among the results of the experimentation[18] we identified the best configuration for each model as the one maximizing the G-mean of DIR.

Accordingly, Tab. I shows the results, reporting details for the analysis at both packet and message level. In addition to the metrics related to DIR, PL (resp. MS), and IAT (resp. IMT), both tables report also the information about the order/memory (S_h/W) of the models as a measure of the related complexity.

From inspection of Tab. I, we can notice that overall Markov-based approaches perform better than ML counterparts in terms of DIR prediction, considering both the analyses at packet and message level, and achieving up to $+0.05$ G-mean in the case of message DIR. The same does not hold for

[18]We considered values for W in $\{1, 3, 5, 10\}$ for all the models.



(a) G-mean DIR. (b) RMSE PL. (c) RMSE IAT.

(d) G-mean DIR. (e) RMSE MS. (f) RMSE IMT.

Figure 12: MC prediction performance vs. order W on Spotify traffic. Values are provided as *avg.* $\pm$ *std.* over 10 folds.

Table II: Large-scale prediction performance summary for MC MMSE, RFR, and Baseline. The **best performance** for each metric—for both packets (first group) and messages (second group)—is highlighted in boldface. Results are in the format *avg. (*$\pm$ *std.)* obtained over the 10 applications.

		S_h/W	G-mean DIR	RMSE PL [B]	RMSE IAT [ms]
Packets	*MC MMSE*	5	**0.87 (±0.08)**	234 (±116)	123.54 (±25.25)
	RFR	5	0.81 (±0.16)	**218 (±106)**	**107.73 (±20.35)**
	Baseline	N/A	0.53 (±0.13)	359 (±155)	212.73 (±30.99)
		S_h/W	G-mean DIR	RMSE MS [kB]	RMSE IMT [s]
Messages	*MC MMSE*	5	**0.83 (±0.08)**	11 (±7)	4.45 (±1.72)
	RFR	5	0.76 (±0.12)	**10 (±5)**	**3.97 (±1.29)**
	Baseline	N/A	0.35 (±0.12)	18 (±14)	5.82 (±2.3)

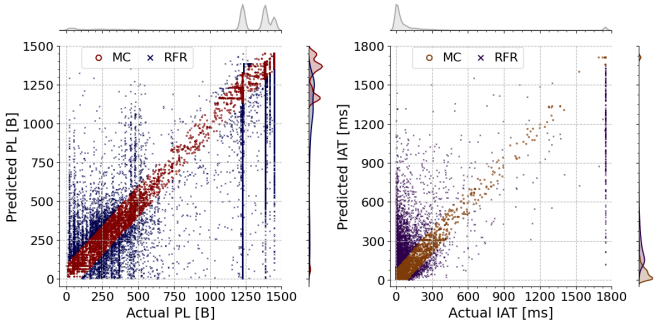
the other metrics: Markov models performs better in terms of PL prediction but not for what concerns MS, IAT, and IMT. In more detail, at packet-level the best results are interestingly obtained by models taking advantage of the same memory size ($W = 3$ for both MC MMSE and RFR). On the other hand, for message DIR prediction MC (both MMSE and MAP) requires a longer window size ($W = 10$ vs. $W = 3$ for the RFR) to obtain a significantly-improved G-mean performance (0.85 vs. 0.80 for the RFR). Focusing on the specific approaches in each group, MMSE MC is the best performing among Markov ones, proving to successfully capitalize the predictive distribution (thus confirming the results in Fig. 12). On the other side, RFR performs always better than the other approaches in the ML group. This is expected due to its ensemble nature which successfully exploits the concept of "bagging".

For the sake of a wider performance evaluation, we show in Tab. II results pertaining to a *large-scale analysis of mobile-app traffic prediction*. In particular, this analysis is performed on 10 different mobile applications by comparing the best Markovian model, i.e. the MMSE-based MC, and the best ML-based model, i.e. the RFR. Specifically, we consider the following applications taken from MIRAGE-2019: Instagram, LinkedIn, Maps, OneDrive, Reddit, Slither.io, SoundCloud, Spotify, Waze, and Wish. The above choice reflects our intention of showing results pertaining to mobile apps not considered in the previous analyses.

The selection of the best model configuration mimics the previous analysis conducted on Spotify: for the same range of W values, we selected that corresponding to the higher (app-averaged) G-mean on DIR. Accordingly, results elect $W = 5$ as the best order, for *both* the approaches and for *both* packets and messages. In detail, MC MMSE confirms a *higher predictive power* in terms of G-mean for both DIR (i.e. $+0.06$ and $+0.07$ for packets and messages, respectively). Differently, by looking at the other features, MC MMSE obtains only slightly-worse (comparable) performance than RFR in terms of PL, IAT, MS, and IMT (-16 B, -15.81 ms, -1 kB and -0.48 s, respectively). As a result, even in a larger-scale evaluation, MC MMSE achieves a satisfactory performance trade-off while retaining the advantages of interpretability due to its white-box modeling nature. Also, MC MMSE allows for (a) updating the model when new samples become available and (b) adapting the model to a shorter memory value $W' < W$ (with impact on complexity and time-to-prediction). This leads to a *reduced overhead* (avoiding additional training cost) w.r.t. (black-box) RFR, where re-training *from scratch* is required in both cases. These further advantages constitute a strong plus in the rapidly-evolving context of mobile traffic.

E. Interpretability of Prediction Results through Markov Analysis

To capitalize the complementary knowledge that MC-MMSE (shortly MC in what follows) is able to provide toward interpretability of ML models, we now (a) deepen the comparison on the error patterns of MC and RFR by inspecting their concordant/discordant predictions, and then (b) interpret the emerging predictive patterns in RFR behavior by leveraging on the meaning of MC learned parameters (i.e. transition probabilities). Without loss of method generality, we focus on the prediction results of Spotify.



(a) RFR misses on PL . (b) RFR misses on IAT.

Figure 13: Actual vs. predicted plot of Spotify predictions in *outcome disagreement*, i.e. for cases when RFR "misses" and MC "hits" (with error thresholds of 100 B for PL and 100 ms for IAT).

First, a comparative analysis on DIR predictions has highlighted that 97.56% (resp. 0.96%) of these are correctly (resp. wrongly) predicted by both MC and RFR. On the other hand, in 0.79% (resp. 0.69%) of the cases the MC (resp. RFR) is able to correctly infer the direction while the RFR (resp. MC) fails. These results highlight the extensive similarity of both approaches in predicting direction, and therefore that when predicting direction RFR mostly behaves simply according to the empirical probabilities. Considering the *absolute error* of MC and RFR in predicting PLs and IATs, we investigate error-patterns using a 100 B (resp. 100 ms) threshold. Indeed, both models successfully predict the 87.75% (resp. 96.15%) of PLs (resp. IATs), and provide wrong predictions for the 5.61% (resp. 2.24%) of PLs (resp. IATs). Differently, looking at the *outcome disagreement* cases, we find that the MC model is able to provide correct predictions for 4.66% (resp. 0.80%) of PLs (resp. IATs) when the RFR fails.

Focusing *only on the outcome disagreement cases* of PL and IAT, in Fig. 13 we investigate the error patterns of RFR (using the aforementioned thresholds). This visualization highlights the cases where RFR fails whereas MC provides acceptable predictions (hereafter *RFR misses* for short), therefore these cases emerge where the departure from the Markov model is counterproductive. Besides the common diagonal pattern (i.e. the locus of *MC hits*), other patterns emerge. Focusing on PL, it is evident that RFR misses mostly occur for large PL values and with specific actual lengths (vertical lines). Concerning IAT instead, RFR misses mostly overestimate the small IATs but with a distinguished pattern of underestimating only the values very close to the maximum (vertical line on the right).

Moreover, to directly compare the two models highlighting the differences in their behaviors, we perform a differential Markovian analysis. To this aim we *distill* small-order Markov Chain models of both the RFR and the MC, and analyze the discrepancies in their predictive behaviors. Accordingly, in Fig. 14 we report the *difference of the distilled transition distributions* obtained for the two methods. In such a representation the values close to zero witness similar distributions, whereas positive (resp. negative) values report higher occurrences for MC (resp. RFR). Considering DIR prediction, Fig. 14a shows that the strongest departure of RFR from the Markovian model is for the "UUU" case (three consecutive upstream packets), where RFR predicts an upstream packet with a probability 11% higher than MC. Concerning PL, both methods result in similar transition probabilities in predicting high PL values when low PL values are the last observed (white bottom-right corner in Fig 14b). Conversely, RFR is more prone to predict low values for PL regardless of the previous value (blue cloud on the left in Fig. 14b), while MC shows two narrow ranges of (long) lengths predicted regardless of the previous length (red vertical lines to the right). Finally, regarding IAT, the two models are vastly equivalent when predicting long intervals, and mostly differ for the prediction of shorter IATs, with different preferences (vertical patterns).

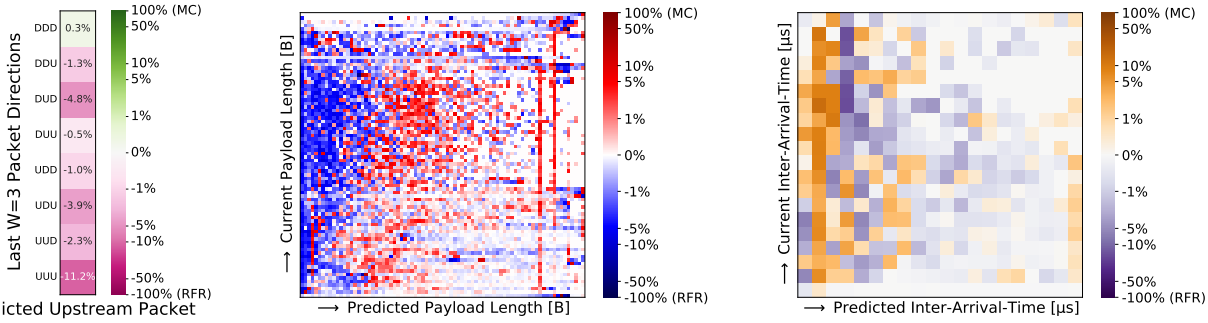
With the tool of differential Markovian analysis introduced above, we further the investigation of RFR errors. Keeping the same distilled models, but *limiting the analysis to RFR misses* (with MC hits), in Fig. 15 we report the difference between the transition distributions conditioned on RFR misses. Considering PL values (Fig. 15a), we can notice how the frequently-predicted (regardless of previous length) vertical lines are kept, i.e. the MC in these cases is correct while the RFR misses. Another pattern of MC hits (missed by RFR) appears when the previous length has close-to-max value (red horizontal line on top). Several cases of specific previous lengths leading to mis-predictions are highlighted as well (horizontal blue segments). Regarding IATs (Fig. 15b), a pattern of RFR errors for longer time intervals is evident, with a strong vertical purple line at the 6^{th} bin (corresponding to the interval 142–200 ms) regardless of the previous IAT.

Such in-depth analyses and insights can be leveraged as a toolset for identifying notable and recurrent error cases, based on the easily-interpretable evolution of input values over time (i.e. transitions) and so provide the guidelines for further improving the design of prediction approaches.

VI. Conclusions and Future Directions

In this paper, we addressed the challenging problem of analyzing the highly complex, dynamic, and encrypted network traffic generated by mobile devices. In detail, we focused on a public human-generated recent dataset, using the methodological toolset of Markov models, that provide an useful interpretability property.

Our contributions cater to the need—especially strong for ML techniques—to have a well-understood and sufficiently diverse dataset for training and evaluation: by employing a family of theoretical approaches to model the (encrypted) network traffic, we provided the ML technique designer and evaluator with a complementary interpretable view and a baseline. We proposed a novel message-reconstruction heuristic to analyze and compare traffic both at the finest (packet-level) and a more application-related (message-level) granularity. We also proposed a rigorous yet synthetically explainable framework for quantifying the heterogeneity of packet traces. The heterogeneity analysis was exploited to highlight and quantify the (dis-)similarity of traffic at different aggregation levels (app version, app, category). By comparing the analyzed approaches with a reference baseline and state-of-the-art ML counterparts, their practical applicability to predicting

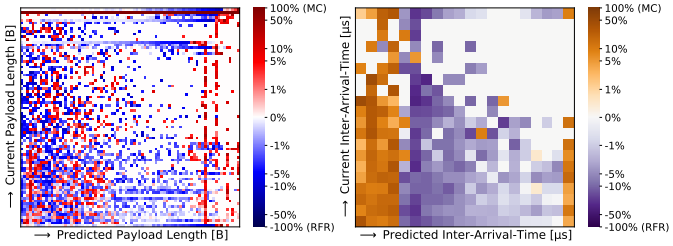


(a) 3^{rd}-order MC distilled from DIR predictions.

(b) 1^{st}-order MC distilled from PL predictions.

(c) 1^{st}-order MC distilled from IAT predictions.

Figure 14: Differential Markov analysis of MC and RFR predictive behaviors. The discrepancy is expressed through the *difference of the distilled 3^{rd}-order transition distributions* for DIR (a), and *1^{st}-order transition distributions* for PL (b) and IAT (c). Positive values represent probabilities higher for MC when compared to RFR ones, negative values vice-versa.



(a) RFR misses on PL

(b) RFR misses on IAT

Figure 15: Differential Markov analysis of MC and RFR predictive behaviors, conditioned on RFR misses. A miss is defined w.r.t. the error thresholds of 100 B for PL and 100 ms for IAT.

(encrypted) mobile traffic at packet- and message-level was validated. Of the two families of models, namely Hidden Markov Models and Markov Chains, the latter showed the best prediction performance for all metrics for both packet- and message-level evaluation. Also, they achieved comparable performance to state-of-the-art ML techniques (e.g., Random Forest Regressor), moreover offering *interpretability tools* to further understanding and enhancement of ML techniques.

In future works, the effectiveness of the considered models will be assessed on other datasets and/or in the context of synthetic traffic generation. We plan to investigate additional and more complex ML methods in our future studies. Also, the use of the proposed models in the aided-design of interpretable ML prediction algorithms is foreseen [31]. As the proposed approaches are designed and applied at the finest granularity (packet level), they are relevant to a wide spectrum of applications (e.g. traffic engineering, routing): thus, further work along these specific lines is envisioned as well.

REFERENCES

[1] G. Aceto and A. Pescapè, "Internet censorship detection: A survey," *Computer Networks*, vol. 83, pp. 381 – 421, 2015.

[2] A. Dainotti, A. Pescapé, and G. Ventre, "A packet-level traffic model of Starcraft," in *2nd IEEE International Workshop on Hot Topics in Peer-to-Peer Systems (HOT-P2P)*, 2005, pp. 33–42.

[3] J. Zhang and I. C. Paschalidis, "Statistical anomaly detection via composite hypothesis testing for Markov models," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 589–602, 2017.

[4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Multi-classification approaches for classifying mobile app traffic," *Journal of Network and Computer Applications*, vol. 103, pp. 131–145, 2018.

[5] F. Jejdling *et al.*, "Ericsson mobility report," *Ericsson AB, Business Area Networks, Tech. Rep. EAB-19*, vol. 7381, November 2019.

[6] G. Maier, F. Schneider, and A. Feldmann, "A first look at mobile handheld device traffic," in *Passive and Active Measurement (PAM)*, 2010, pp. 161–170.

[7] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *10th ACM SIGCOMM conference on Internet Measurement (IMC)*, 2010, pp. 281–287.

[8] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, "NetworkProfiler: Towards automatic fingerprinting of android apps," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2013, pp. 809–817.

[9] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and Distributed System Security Symp. (NDSS)*, 2020.

[10] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Robust smartphone app identification via encrypted network traffic analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 1, pp. 63–78, 2018.

[11] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app traffic capture and ground-truth creation," in *4th International Conference on Computing, Communications and Security (ICCCS)*, Oct. 2019, pp. 1–8.

[12] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," in *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2011, pp. 329–344.

[13] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, pp. 65–76, 2012.

[14] X. Wei, N. C. Valler, H. V. Madhyastha, I. Neamtiu, and M. Faloutsos, "Characterizing the behavior of handheld devices and its implications," *Computer Networks*, vol. 114, pp. 1–12, 2017.

[15] I. Nevat, D. M. Divakaran, S. G. Nagarajan, P. Zhang, L. Su, L. L. Ko, and V. L. L. Thing, "Anomaly detection and attribution in networks with temporally correlated traffic," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 131–144, 2017.

[16] W. Sha, Y. Zhu, M. Chen, and T. Huang, "Statistical learning for anomaly detection in cloud server systems: A multi-order Markov chain framework," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 401–413, 2015.

[17] H. Liu, L. T. Yang, J. Chen, M. Ye, J. Ding, and L. Kuang, "Multivariate multi-order Markov multi-modal prediction with its applications in network traffic management," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 3, pp. 828–841, 2019.

[18] L. Muscariello, M. Mellia, M. Meo, M. A. Marsan, and R. L. Cigno, "Markov models of Internet traffic and a new hierarchical MMPP model," *Elsevier Computer Communications*, vol. 28, no. 16, pp. 1835–1851, 2005.

[19] H. Okamura, T. Dohi, and K. S. Trivedi, "Markovian arrival process parameter estimation with group data," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1326–1339, 2009.

[20] G. Casale, A. Sansottera, and P. Cremonesi, "Compact Markov-modulated models for multiclass trace fitting," *Elsevier European Journal of Operational Research*, vol. 255, no. 3, pp. 822–833, 2016.

[21] S. Colonnese, P. Frossard, S. Rinauro, L. Rossi, and G. Scarano, "Joint source and sending rate modeling in adaptive video streaming," *Signal Processing: Image Communication*, vol. 28, no. 5, pp. 403–416, 2013.

[22] A. Dainotti, A. Pescapé, P. Salvo Rossi, F. Palmieri, and G. Ventre, "Internet traffic modeling by means of hidden Markov models," *Computer Networks*, vol. 52, no. 14, pp. 2645–2662, 2008.

[23] S. Maheshwari, S. Mahapatra, C. S. Kumar, and K. Vasu, "A joint parametric prediction model for wireless internet traffic using hidden Markov model," *Wireless networks*, vol. 19, no. 6, pp. 1171–1185, 2013.

[24] S. Dash, S. Maheshwari, and S. Mahapatra, "Traffic prediction in future mobile networks using hidden Markov model," in *IEICE Smart Wireless Communications (SmartCom)*, 2019.

[25] L. Rossi, J. Chakareski, P. Frossard, and S. Colonnese, "A Poisson hidden Markov model for multiview video traffic," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 547–558, Apr. 2015.

[26] S. Tanwir, D. Nayak, and H. Perros, "Modeling 3D video traffic using a Markov modulated gamma process," in *Int. Conference on Computing, Networking and Communications (ICNC)*, Feb. 2016, pp. 1–6.

[27] Y. Xie, J. Hu, Y. Xiang, S. Yu, S. Tang, and Y. Wang, "Modeling oscillation behavior of network traffic by nested hidden Markov model with variable state-duration," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1807–1817, 2013.

[28] U. K. Sarkar, S. Ramakrishnan, and D. Sarkar, "Modeling full-length video using Markov-modulated Gamma-based framework," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 638–649, 2003.

[29] M. Mouchet, S. Vaton, T. Chonavel, E. Aben, and J. Den Hertog, "Large-scale characterization and segmentation of internet path delays with infinite HMMs," *IEEE Access*, vol. 8, pp. 16 771–16 784, 2020.

[30] A. Razaghpanah, A. A. Niaki, N. Vallina-Rodriguez, S. Sundaresan, J. Amann, and P. Gill, "Studying TLS usage in Android apps," in *13th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2017, pp. 350–362.

[31] H. Hagras, "Toward human-understandable, explainable AI," *IEEE Computer*, vol. 51, no. 9, pp. 28–36, 2018.

[32] A. Dainotti, A. Pescape, and H. Kim, "Traffic classification through joint distributions of packet-level statistics," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2011, pp. 1–6.

[33] D. H. Hagos, P. E. Engelstad, A. Yazidi, and Ø. Kure, "General TCP state inference model from passive measurements using machine learning techniques," *IEEE Access*, vol. 6, pp. 28 372–28 387, 2018.

[34] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video QoE from encrypted traffic," in *ACM Internet Measurement Conference (IMC)*, 2016, p. 513–526.

[35] K. Park and H. Kim, "Encryption is not enough: Inferring user activities on KakaoTalk with traffic analysis," in *Springer International Workshop on Information Security Applications (WISA)*, 2016, pp. 254–265.

[36] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Security*, 2017.

[37] C. Liu, Z. Cao, G. Xiong, G. Gou, S.-M. Yiu, and L. He, "MaMPF: Encrypted traffic classification based on multi-attribute Markov probability fingerprints," in *IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–10.

[38] S. Alcock and R. Nelson, "Passive detection of TCP congestion events," in *IEEE 18th Int. Conf. on Telecommunications*, 2011, pp. 499–504.

[39] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing Internet Latency: A Survey of Techniques and Their Merits," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2149–2196, 2016.

[40] M. B. Fjordvald and C. Nedelcu, *Nginx HTTP Server: Harness the power of NGINX to make the most of your infrastructure and serve pages faster than ever before*. Packt Publishing Ltd, 2018.

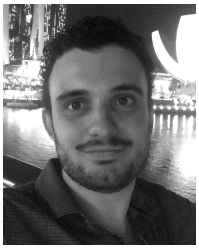
[41] A. Basu, A. Mandal, and L. Pardo, "Hypothesis testing for two discrete populations based on the Hellinger distance," *Elsevier Statistics & Probability Letters*, vol. 80, no. 3-4, pp. 206–214, 2010.

[42] J. Beirlant, E. J. Dudewicz, L. Györfi, and E. C. Van der Meulen, "Nonparametric entropy estimation: An overview," *International Journal of Mathematical and Statistical Sciences*, vol. 6, no. 1, pp. 17–39, 1997.

Giuseppe Aceto is an Assistant Professor at University of Napoli Federico II, where he received his PhD in Telecommunication Engineering. His research concerns network performance and censorship, both in traditional networks and SDN, and ICTs applied to health. He received the best paper award at IEEE ISCC 2010, and 2018 Best Journal Paper Award by IEEE CSIM.

Giampaolo Bovenzi is a PhD Student at DIETI, University of Napoli Federico II, since November 2018. He achieved MS degree (summa cum laude) at the same University in October 2018. His research interests focus on (anonymized and encrypted) traffic classification, network security (with focus on IoT), and blockchain.

Domenico Ciuonzo (S'11-M'14-SM'16) is an Assistant Professor at University of Napoli Federico II. He holds a PhD from University of Campania L. Vanvitelli (IT) and, from 2011, he has held several visiting researcher appointments. Since 2014 he is editor of several IEEE, IET and ELSEVIER journals. His research interests include data fusion, wireless sensor networks, internet of things, network analytics and machine learning.



Antonio Montieri (GSM'18) is a Postdoctoral Researcher at DIETI of the University of Napoli Federico II. He has received his Ph.D. degree in Information Technology and Electrical Engineering in April 2020 from the same University. His work concerns network measurements, (encrypted and mobile) traffic classification, traffic modeling and prediction, and monitoring of cloud network performance. Antonio has co-authored 25 papers in international journals and conference proceedings.

Valerio Persico is an Assistant Professor at DIETI, University of Napoli Federico II, where he received the PhD in Computer and Automation Engineering in 2016. His work concerns network measurements, cloud-network monitoring, and Internet path tracing and topology discovery. He has co-authored more than 30 papers within international journals and conference proceedings.

Antonio Pescapé (SM'09) is a Full Professor of computer engineering at the University of Napoli Federico II. His work focuses on measurement, monitoring, and analysis of the Internet. He has co-authored more than 200 conference and journal papers, he is the recipient of a number of research awards. Also, he has served as an independent reviewer/evaluator of research projects/project proposals co-funded by a number of governments and agencies.