# FLOWR: A Self-Learning System for Classifying Mobile Application Traffic

Qiang Xu[#], Thomas Andrews[#], Yong Liao[*], Stanislav Miskovic[*], Z. Morley Mao[#]
Mario Baldi[*], Antonio Nucci[*]
[#]University of Michigan, [*]Narus Inc.

## Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: network monitoring, wireless communication; C.4 [**Performance of Systems**]: measurement techniques

## General Terms

Experimentation, Measurement, Performance

## Keywords

Mobile Application Identification, Traffic Classification

## 1. INTRODUCTION

We aim to devise a method that can identify mobile apps related to each individual traffic flow in the wild. Mobile apps are becoming preferred means of Internet access for a growing user population. Such departure from browser based Internet poses a unique challenge to traffic management tools, still largely incapable of handling mobile apps. Consequently, enterprises and service providers become hindered by being unable to deploy effective mobile policies and security solutions.

Traditionally, desktop applications and networking protocols were identified by signatures derived from transport-layer ports, ip addresses, or domain names [2, 5]. It is not suitable for mobile apps any more. The main reason is that most mobile apps communicate via generic HTTP/HTTPS traffic, thus being a priori indistinguishable from Internet browsing. State-of-the-art solutions attempted to develop signatures via user studies or app emulations [6, 4, 1]. Neither of the two approaches scales due to a number of key challenges:

- **Similarity.** Besides using similar protocols (HTTP/HTTPS), mobiles apps communicate with largely similar IP-/domain-level destinations, Content Delivery Networks (CDNs), and cloud services, which makes them difficult to distinguish.
- **Scalability.** With hundreds of thousands of apps, the identification has to devise very efficient matching algorithms at line speeds. Moreover, the references for matching have to be obtained efficiently. One cannot assume running all existing apps to get the references, or keeping a large state as means of identification.
- **Ground truth.** Being infeasible to exhaustively establish references for app characterization, one loses strong ground truth. Generally, such ground truth would be attainable via app-identifying and a priori unknown features such as developer tags in traffic, proprietary app-identifying URL parameters, hostnames, etc.
- **Coverage.** The issue of coverage is closely related to app matching efficacy: Ideally, one should know all communication states and patterns of each app to perfectly characterize each traffic flow, similarly to a sophisticated app signature generation scheme described in [3]. However, such thorough identification of all apps' flows would require exhaustive and unattainable offline training.

In this paper, we develop a flow recognition system called FLOWR, which identifies in real time the app origins of individual flows in traffic. The system is based on discovering and learning a priori uncertain hints that may exist in the traffic and point to particular apps. Over time, FLOWR automatically learns an increasing number of such hints and evaluates the confidence in their app-identifying capabilities. To optimize our identification capabilities to line speeds, we only look into HTTP header metadata.

We believe there are many reasons for the existence of app-identifying features in the traffic. For example, developers generally like to track adoption of their apps, so they put some reporting capabilities in the apps' communications, e.g., using specific user agent tags. Moreover, we know that ads and analytic services (such as `doubleclick.net` and `flurry.com`) embed explicit app identifiers in HTTP headers for accounting purposes. Nevertheless, our preliminary experiments showed that the percentage of such explicitly identifiable flows is extremely small, while our goal is to enable large flow-identification coverage.

Our methodology consists of three key components: (1) KV tokenization, which parses HTTP traffic and extracts all key-value (KV) pairs that may become app-identifying features, (2) knowledge base, which contains all previously known KV pairs and a confidence measure of their app-identifying capabilities, and (3) flow regression, which measures co-occurrence of flows (and their embedded KV pairs) as a confidence indication of the flows (and their KV pairs) originating from the same apps. To the best of our knowledge, this constitutes the first system to identify apps at flow level in real time and at a scale suitable to large operational networks, such as nationwide mobile networks or large enterprises.


*SIGMETRICS'14,* June 16–20, 2014, Austin, Texas, USA.
ACM 978-1-4503-2789-3/14/06..

## 2. CLASSIFICATION TECHNIQUES

Here, we describe the three key components of the FLOWR system.

### 2.1 Extracting Flow Signatures

We extract KV pairs from HTTP URIs. We only focus on HTTP requests because our preliminary experiments showed that HTTP responses contribute only marginally to app-identification. HTTP URIs have the following well-formatted syntax: `http://host_name/path?query#fragment`. The `query` consists of a sequence of KVs formatted as `k1=v1&k2=v2&...&kN=vN`. For example, KV pairs can be `adkapid=67526`, `age=45`, and `zipcode=90210`.

### 2.2 Seeding the Knowledge Base

We identified three general types of KV pairs based on their app-identifying capabilities: irrelevant KVs (e.g., `age=45`), KVs explicitly referring to app identities (e.g., `packageName=zz.rings.rww2`), and other KVs whose app identifying capabilities are initially ambiguous (e.g., `sdkapid=67526`) and which need to be evaluated further by our algorithm. Ad and analytic services, such as `doubleclick.net`, are particularly rich of explicit KV pairs. Hence, FLOWR relies on them to provide seeds for our knowledge base. Then, by means of flow regression, we spread these knowledge seeds to other observed KV pairs.

### 2.3 Flow Regression - Inferring KV Identities

We devised flow regression as a technique to eliminate irrelevant KVs and infer app identities via scoring ambiguous KVs. Intuitively, if two flows repeatedly co-occur, they are likely to come from the same app. To be confident in flow co-occurrence properties, we observe flows over various users and various disjoint time intervals. As a result, flow regression scores observed flow co-occurrence likelihoods, as well as ties the ambiguous KVs in these flows to app identities of readily app-identifying KVs also found in the flow set.

FLOWR digests co-occurrence likelihoods as follows. In the background, if a `KV` appears frequently close to some flow related to an app `X`, i.e., $\mathtt{P[A = X|KV]} \approx 1$, FLOWR adds `KV` to its signature set that uniquely identifies the app in the knowledge base of app identities. Instead of $\mathtt{P[A = X|KV]} \approx 1$, we found that a more common case is that `KV` co-occurs with `X` with a certain likelihood of less than 1, i.e., $0 \ll \mathtt{P[KV|A = X]} < 1$, which indicates that `KV` may be generated by apps other than `X` as well. Then, as best effort, FLOWR puts `KV` into a signature set that narrows down the set of the flow's candidate apps.

Next, FLOWR generally extracts multiple KV pairs per flow and based on the KV scores the algorithm needs to determine app identity. In our implementation, the best KV determines the identity, i.e., the KV pair with most frequent and most unique co-occurrence with an app ties the entire flow to that app's identity.

## 3. EVALUATION

FLOWR is evaluated using an anonymized traffic trace captured on a major 3G cellular network in the US. Our evaluation demonstrates that FLOWR is able to learn new KVs from a minimal seeding app identification knowledge base. The learned new KVs are consumed by FLOWR to classify an order of magnitude more HTTP flows, whose originating apps could not be identified via the seeding knowledge.

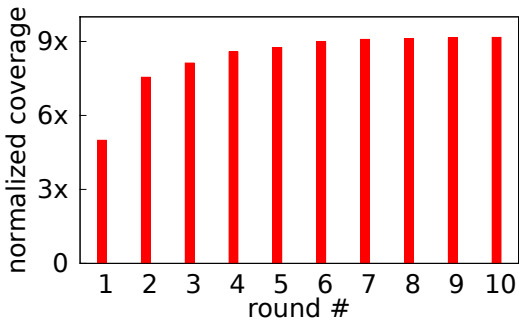


**Figure 1:** Flow coverage.

Our experiment is seeded with only one piece of manually created knowledge. FLOWR is instructed as to which KV in an HTTP message sent to `doubleclick.net`, a popular ad service used by both Android and iOS apps, indicates mobile app identities. We run FLOWR's flow regression on the traffic trace iteratively. In each round, FLOWR considers a new KV as identifying an app only when the estimated false positive rate of using that KV is lower than 1%. FLOWR estimates the false positive rate via a set of carefully tuned and tested heuristics, whose details are omitted here due to space limit. Additional KVs selected in earlier execution rounds are used in later rounds as seeding knowledge. Figure 1 shows the coverage, i.e., the number of classified flows, normalized to the one obtained with the initial seeding knowledge. Note that almost $9\times$ more flows can be classified with only $3 \sim 4$ rounds of flow regression.

## 4. CONCLUSION

In this study, we developed the FLOWR system for determining the app identities of network flows in mobile networks in real time with minimal supervised learning. FLOWR adopts three creative techniques: KV tokenization, ad and analytic services based training, and flow regression. Without consuming exhaustive training effort, flow regression can determine the app identities for the flow signatures produced by KV tokenization through monitoring the co-occurrence events between flow signatures. We believe that FLOWR enables mobile network operators to extract valuable insight from mobile network traffic.

## 5. REFERENCES


[1] Netsense. http://netsense.nd.edu.

[2] Cui, W., Kannan, J., and Wang, H. Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *Proc. USENIX Security* (2007).

[3] Dai, S., Tongaonkar, A., Wang, X., Nucci, A., and Song, D. NetworkProfiler: Towards Automatic Fingerprinting of Android Apps. In *Proc. IEEE INFOCOM* (2013).

[4] Qian, F., Wang, Z., Gerber, A., Mao, Z., Sen, S., and Spatscheck, O. Profiling Resource Usage for Mobile Applications: A Cross-Layer Approach. In *Proc. ACM MobiSys* (2011).

[5] Sen, S., Spatscheck, O., and Wang, D. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *Proc. ACM WWW* (2004).

[6] Wei, X., Gomez, L., Neamtiu, I., and Faloutsos, M. ProfileDroid: Multi-Layer Profiling of Android Applications. In *Proc. ACM MOBICOM* (2012).