Automatic discovery of web servers hosting similar applications

Jan Kohout[⋆†] and Tomáš Pevný[⋆†]
[⋆]Cisco Systems
E-mail: {jkohout,tpevny}@cisco.com

[†]Department of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University in Prague

***Abstract*—Increasingly more popular cloud services have frequently many functional parts, which makes their structure rather complex yet its understanding improves network monitoring for security purposes, traffic routing, etc. Since the structure of third-party services is typically unknown, automated tools for its discovery are of great need. In this work, we propose such tool relying only on high-level statistics of servers' usage, such as volumes and times of interactions with the servers. Without looking into the communication contents, the method works for encrypted channels as well, which is experimentally demonstrated on Dropbox service and Windows Live platform.**

***Index Terms*—clustering, servers fingerprinting, service identification**

I. Introduction

We currently see a trend of moving network services and applications from local networks to the cloud, accessed via a web interface. Although this at one hand simplifies IT infrastructure and its costs, it also brings new challenges with detecting policy violations, misuses, and attacks. This is because network administrators have lost the direct control over the services and typically they have a very little insight into services' internals.

Many contemporary services have rather complex structure, as they are composed of several sub-services fulfilling specific tasks. The knowledge of this structure can simplify services' monitoring and improve security countermeasures against their misuses. However, the structure is not typically publicly known and its analysis by hand often includes reverse engineering of the communication protocol [1], which is time consuming and frequently too costly. Moreover, the rise of popularity of the HTTPS protocol, accelerated by Snowden's affair, makes deep packet inspections difficult and also rules-out any port-based identifications. Also, inferring the service's structure from hostnames similarities or proximities of servers' IP addresses can be misleading, because there is no guarantee that servers with similar hostnames (or with IP addresses from the same subnet) run the same type of service.

This work shows how a service's structure can be discerned solely from high-level information such as number of transmitted bytes and times of connections to servers — all being available in HTTPS traffic. The paper is structured as follows. In Section II, we propose compact "fingerprints" of servers extracted from the traffic. These fingerprints are used by off-the-shelf clustering algorithm to identify groups of similar servers without knowing the number of groups in advance. The discussion about the clustering algorithm is also part of Section II. In Section III, the method is deeply evaluated on the Dropbox service chosen due to known ground truth [1], and on a subset of servers belonging to the Windows Live platform. We review the related work in Section IV. Finally, we conclude the work in Section V.

II. The proposed approach

The goal of identifying groups of servers hosting same applications or their distinct functional part (e.g., Dropbox has dedicated servers for notification of changes and others for serving the content), can be viewed as a clustering problem. Using off-the-shelf clustering methods requires definition of a similarity measure quantifying similarity of two points (servers). Since we thrive for efficiency and scalability, we want a server to be represented by a point in Euclidean $\mathbb{R}^d$ space. This choice allows to use large variety of clustering algorithms.

The next subsection presents how servers can be represented in an Euclidean space $\mathbb{R}^d$ allowing those providing similar services to be close to each other. The choice of the similarity measure and the clustering algorithm is discussed in Subsection II-B.

A. Fingerprinting servers

The representation of a server, called fingerprint, was designed under following constraints. First, we wanted to use information that is easily available without significant effort and privacy concerns which rules out deep packet inspection or decryption of the traffic. Therefore, the data can be obtained from proxy logs provided by systems like Squid [2] or possibly from NetFlow-like[1] traces. Second, the application can use HTTPS meaning that most information from URLs like path, parameters, domain names, and referrers can be missing. Third, we avoid any features specific for a certain class of application, as the method would not be general and could

[1]Using NetFlows would require redefinition of server being IP:port and slight change of features, because flows in the NetFlow format are unidirectional.

not be used without prior knowledge about the service. This left us with relatively high-level information about each web request to the servers of interest, namely:

1) **bytes sent** r_{up} from the client to the server,
2) **bytes received** r_{down} by the client from the server,
3) **duration**: r_{td} (in milliseconds) of the request,
4) **inter-arrival time** r_{ti} (in seconds) elapsed between start of the request and previous request from the same user.

All four quantities are transformed by $\log(1+x)$ to decrease their dynamic range. Although the information about servers' usage might seem limited, similar information about packets were used to classify application level protocols [3]. With respect to the available information, a request r can be reduced to a 4-tuple $r = (r_{\mathrm{up}}, r_{\mathrm{down}}, r_{\mathrm{td}}, r_{\mathrm{ti}})$.

Server's fingerprint relies on an assumption that statistical distribution of requests is similar for servers running the same application. The distribution is captured by a four-dimensional joint soft histogram, which arranged in a vector forms the server's fingerprint. As the joint soft histogram is the keystone of the proposed fingerprint, the rest of this subsection is devoted to it.

Soft histogram: A commonly used approximation of the joint distribution of $r_{\mathrm{up}}, r_{\mathrm{down}}, r_{\mathrm{td}}, r_{\mathrm{ti}}$ from a finite number of samples is through empirical joint histogram with equal bins. Its advantage is that it captures dependencies between all quantities at cost of large number of bins, if the quantization is fine. We believe that this should not be a problem because (a) histograms are typically sparse due to the regularity of servers usage and (b) our goal is clustering, not a classification where we should worry about over-fitting. Moreover, in text document analysis ([4]) sparse and high-dimensional representations are common. The sparsity is demonstrated in Figure 1 showing the cumulative distribution of the number of non-zero items in fingerprints of 11500 servers under the `.com` top level domain. We can see that fingerprints of more than 95% servers have less than 5% of non-zero bins. This also demonstrates the regularity of servers' usage.

Common (hereafter called hard) histograms are constructed by increasing values of bins into which samples fall irrespectively how close they are to the bins' boundaries. This strict quantization makes histograms sensitive to small variations (noise) in the data. In image and signal processing [5], [6] this sensitivity is removed by using so-called soft histograms, where each sample updates two (in a one-dimensional case) closest bins by values proportional to their distance. In the simplest form, sample's contribution to two nearest bins depends linearly on distance to them, which corresponds to the triangular filters used in signal processing [6].

The situation is depicted in Figure 2, where a one-dimensional histogram is updated by a sample $u = 2.6$. Its two nearest bins centred in $\lfloor u \rfloor = 2$ and $\lfloor u \rfloor + 1 = 3$ are updated by $1 - (u - \lfloor u \rfloor) = 0.4$ and $u - \lfloor u \rfloor = 0.6$, respectively.

Soft histograms can be extended to m dimensions with bins centred at integer lattice points $[b_1, ..., b_m] \in \{0, ..., n\}^m$, where n is an upper bound on values to be inserted to the histogram. Updating the m-dimensional soft histogram with a

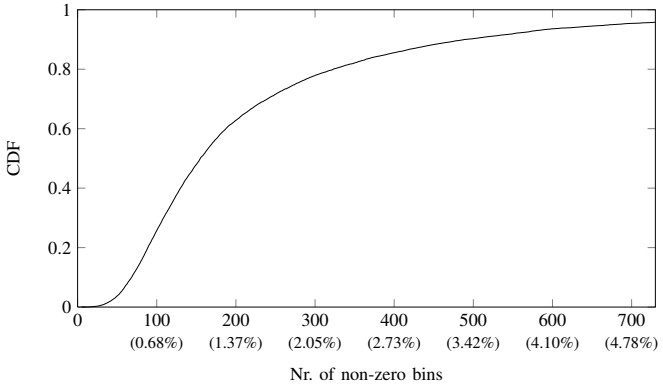

Fig. 1. Cumulative distribution function (CDF) of the number of non-zero bins in fingerprints of totally 11500 servers found under the `.com` TLD. The fingerprints were extracted from 5 days of continuous traffic. The CDF plot demonstrates that fingerprints are typically very sparse — the total number of bins in a fingerprint was set to $11^4 = 14641$, thus more than 95% of fingerprints have less than 5% of non-zero bins.

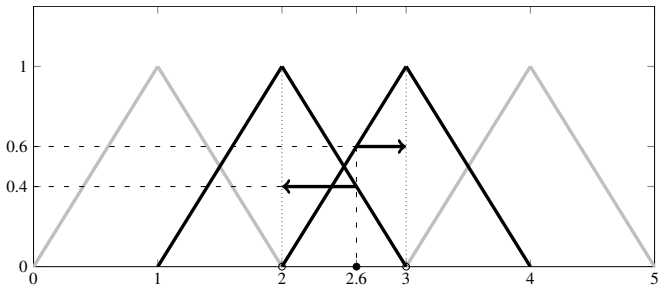

Fig. 2. Example of updating a one-dimensional soft histogram with value 2.6. Filters influencing the contribution are highlighted.

tuple $(u_1, u_2, \ldots, u_m)$ means first calculating indices l_i and contributions v_i to "left" bins as

$$l_i = \lfloor u_i \rfloor, \quad v_i = 1 - (u_i - l_i), \quad i \in \{0, \ldots, m\},$$

and then updating all bins centred in vertices

$$\{(l_1 + i_1, \ldots, l_m + i_m) | (i_1, \ldots, i_m) \in \{0, 1\}^m\}$$

with values $\prod_{j=1}^{m} v_j^{1-i_j}(1 - v_j)^{i_j}$. Finally, the soft histogram is L_1-normalized to approximate the joint probability distribution.

Based on values observed in web requests in our experimental data, we have found $n = 10$ to be sufficient implying the total number of bins to be $11^4 = 14641$. The final server's fingerprint is then the soft histograms arranged in a $(n+1)^m$-dimensional column vector.

B. Fingerprints clustering

Clustering servers' fingerprints follows the usual clustering steps. First, similarities between all fingerprint pairs are calculated, then optionally τ_s smallest similarities are set to zero to accentuate true clusters, and finally the chosen clustering algorithm is applied. The steps are detailed below.

Similarity measure: In experiments presented in Section III we have evaluated two similarity measures between fingerprints x_1 and x_2. s_c is the usual cosine similarity frequently used in document analysis while s_e is based on Euclidean L_2 distance scaled to $[0, 1]$ such that both similarity functions have the same range. The scaling leverages the fact that the upper-bound on L_2 distance between x_1 and x_2 is $\sqrt{2}$, because L_1 norms of the fingerprints are 1 and their items are non-negative. These similarities are formally defined as

$$s_\mathrm{c}(x_1, x_2) = \frac{x_1^T x_2}{||x_1||_2 \cdot ||x_2||_2} \quad (1)$$

$$s_\mathrm{e}(x_1, x_2) = \frac{\sqrt{2} - ||x_1 - x_2||_2}{\sqrt{2}} \quad (2)$$

Discarding low similarities: Optionally, τ_s percent of the lowest similarities are set to zero making the respective servers completely dissimilar. Although this filtering might decrease noise and accentuate the true clusters, it can also discard too much information rendering the true clusters unrecognisable. The impact of filtering on the accuracy of clustering is investigated in Section III.

Clustering algorithm: Any clustering algorithm accepting either feature vectors from $\mathbb{R}^d$ or a similarity matrix can be used. Nevertheless, clustering algorithms are not the same and results can significantly differ. The clustering algorithm of our choice is the Louvain method [7] designed for discovering communities in graphs. The vertices of the graph which is passed to the Louvain clustering represent servers' fingerprints while the similarity measures s_c or s_e determine weights of the edges between the vertices. The filtering described in the previous paragraph has an effect of removing edges. An advantage of the Louvain method is the optimisation of the number of clusters, which is useful for applications when the desired number of clusters is not known beforehand. However, we again emphasize that servers' fingerprints are general and other clustering algorithms can be taken to account, e.g., those based on spectral clustering [8].

III. Evaluation

The efficacy of the proposed method for discovering groups of servers hosting same applications is evaluated in detail on the Dropbox service, where we want to identify its separate functional parts. Dropbox has been chosen because the ground truth exists thanks to [1] and the majority of communication is encrypted by using HTTPS. Thus, it is a good example of a service into which the insight is very limited unless the communication is decrypted, which has been done in [1]. This evaluation is presented in Subsection III-A. In Subsection III-B, we show the results of clustering of servers that belong to the Windows Live platform to demonstrate that our method can be used for analysis of various types of services. Despite that we had no ground truth for these data, the brief analysis of the produced clusters shows promising results as well.

Histograms	Clustering	Similarity	Filtering level τ_s 0%	20%	50%	70%
soft	Louv.	Cosine	0.947	0.947	0.947	0.443
		Euclidean	0.723	0.732	0.732	0.948
	Spect.	Cosine	0.887	0.853	0.860	0.788
		Euclidean	0.793	0.582	0.751	0.746
hard	Louv.	Cosine	0.740	0.945	0.441	0.310
		Euclidean	0.722	0.723	0.723	0.933
	Spect.	Cosine	0.849	0.822	0.138	0.035
		Euclidean	0.786	0.489	0.653	0.699

TABLE I
VALUES OF ADJUSTED RAND INDEX COMPARING SIMILARITY BETWEEN THE GROUND TRUTH CLUSTERING OF THE DROPBOX SERVERS AND THE CLUSTERING PRODUCED BY OUR METHOD FOR DIFFERENT SIMILARITY MEASURES, LEVELS OF FILTERING AND TYPES OF HISTOGRAM (HIGHER IS BETTER, MINIMUM IS -1, MAXIMUM IS 1).

A. Dropbox analysis

For the purposes of the evaluation, we extracted fingerprints of 188 servers under the `dropbox.com` domain from 5 days of continuous web traffic from a larger company with approximately 10000 active users in the network, obtaining 17000 requests per server on average. The data were collected during the year 2013. 95% of servers fell into 4 categories out of 11 identified in [1], namely: `clientX.dropbox.com` (meta data management), `dl-clientX.dropbox.com` (client storage), `dl-debugX.dropbox.com` (exceptions backtraces) and `notifyX.dropbox.com` (notifications about changes), where the letter `X` stands for one or more digits in the hostname. Hereafter we refer to these four groups of hostnames as ground truth groups.

The quality of clustering was measured by Adjusted Rand Index (ARI) ([9], [10]), which is a general measure taking value in the range $[-1, +1]$ evaluating agreement of two clustering solutions (higher value means better match of the solutions). In our case, we always compared outcome of an evaluated clustering method to the ground truth. For details about calculations of ARI we refer the reader to the original publications.

Below we compared the Louvain and the spectral clustering. As the spectral clustering needs the number of clusters to be supplied in advance, it was always set to the correct number.

The main bulk of experimental results is shown in Table I comparing ARI for all combinations of Louvain and spectral clustering, cosine and Euclidean similarities, and 4 different levels of filtering $\tau_s \in \{0\%, 20\%, 50\%, 70\%\}$. Supplemented are results for hard histograms for comparison. The results show superiority of the Louvain clustering over the spectral. Also the cosine similarity (1) gives consistently better results than the Euclidean similarity (2) and it is oblivious to the filtering level τ_s (unless most of the similarities are discarded). The advantage of soft histograms advocated in Subsection II-A over the traditional hard ones is also apparent. We point out that the best combination of the Louvain clustering and cosine

Cluster	ground truth group	Precision	Recall
A	clientX	1.00	1.00
B	dl-clientX	0.86	1.00
C	dl-debugX	1.00	1.00
D	notifyX	1.00	1.00

TABLE II
DOMINANT TYPES OF DROPBOX SERVERS IN THE FOUR LARGEST CLUSTERS.

similarity is virtually parameter free, as the number of clusters is discovered automatically.

To illustrate how well the obtained clusters represent functional parts of Dropbox, Figure 3 shows marginal probability distributions of all four modeled quantities estimated from fingerprints in four largest clusters obtained by Louvain clustering with cosine similarity and $\tau_s = 0\%$. Titles of groups correspond to the most dominant part of Dropbox service in the cluster. We can see that requests to notifyX servers have zero uploaded bytes, nearly constant number of downloaded bytes, they are periodic and of the same (long) duration. This well corresponds with notify servers informing clients about changes by implementing the push mechanism. Contrary, connections to dl-clientX are aperiodic with relatively large amounts of uploaded and downloaded bytes. This is caused by the fact that these servers are used to upload and download content to/from the storage. Servers in dl-debugX are used to send debug data and requests to them have long durations like requests to notifyX, but unlike those they have non-zero uploaded bytes and are aperiodic. The higher volume of transmitted bytes can be caused by the long duration of requests. Finally, requests to clientX have the shortest durations among all groups and in comparison to requests to dl-debugX and dl-clientX the amount of transferred bytes is smaller. According to [1], these servers handle meta-data but the exact protocol is unknown to us.

The titles in Figure 3 were inferred from dominant ground truth labels within. To illustrate the purity of clusters, Table II shows clusters' precisions and recalls. The precision is defined as the ratio of dominant ground truth labels within the cluster to its size. Similarly, the recall is defined as the ratio of the dominant ground truth labels within the cluster to the total number of those labels in the entire dataset. We can see that all clusters are pure and servers with the same functionality are in one cluster. The only exception is cluster B (dl-clientX) containing some services from remaining 5% of servers providing other Dropbox functionality.

The cosine similarity graph of servers is visualized in Figure 4 with the help of Gephi [11] tool and the Force Atlas 2 drawing algorithm [12]. This algorithm attracts nodes with high similarities towards each other while separating pairs of nodes with low similarities. The graph supports conclusions drawn above – despite that marginal distributions of the features for the dl-debugX (blue) and dl-clientX (red) servers are more similar, nodes representing servers from the same ground truth groups are attracted together because of high similarities of servers inside both of the ground truth groups. Therefore, the clustering algorithm is able to separate these servers correctly.

B. Windows Live analysis

The ability of our method to distinguish servers hosting different applications within a service which is different from Dropbox is demonstrated in Figure 5. The figure shows a cosine similarity graph of 310 servers revealed under the live.com second level domain (former Windows Live services). The data set used for this experiment was the same as the one which was used for the Dropbox analysis. Again, the graph was visualised with help of the Force Atlas 2 algorithm and five largest clusters (containing 94% of all servers) discovered by the Louvain method are highlighted with different colors (red, blue, green, purple, and yellow). For these servers, we did not have the exact ground truth like in the case of Dropbox, thus, the detailed analysis of the method's performance could not be performed. However, as shown in Figure 5, the servers form several well-shaped clusters which suggests that these clusters could gather servers running different applications. Indeed, the analysis of hostnames belonging to servers in the five largest clusters revealed that 76% of servers in the blue and in the red cluster have a word "mail" in their hostnames. This is a good evidence that these clusters are likely to represent servers engaged in mail services of the Windows Live platform. On the other hand, 99% of servers in the green cluster and in the purple cluster contain a word "messenger" in their hostnames. This points out that these two clusters gather servers handling the instant messaging service (Windows Live Messenger). Similarly, 67% of servers in the yellow cluster have a "storage" substring in their hostnames. Thus, we can assume that the servers from the yellow cluster are somehow involved in storing users' data.

This brief analysis of the obtained results shows that our method is able to successfully distinguish servers running different applications in general which makes it readily applicable to network traffic analysis in practice.

IV. RELATED WORK

The goal of this work resembles identification of the type of communicating application from the network traffic without inspecting the communication's contents. Ref. [3] builds classifiers for different application layer protocols using only information about packets sizes and timings. Contrary to ours, their work uses TCP layer data and the classifiers are trained to identify previously known protocols. Ref. [13] also classifies application protocol from sizes and inter-arrival times of packets. We differ to them as we do not use any knowledge about what we will cluster and we focus on servers rather than flows. Ref. [14] demonstrates a classifier identifying web mail traffic over HTTPS without content inspection. As the focus is on identification of web mail traffic, the proposed representation is tightly fit to that problem (e.g., assumes proximity between web mail servers and known legacy mail servers in the IP address space) and cannot be easily ported

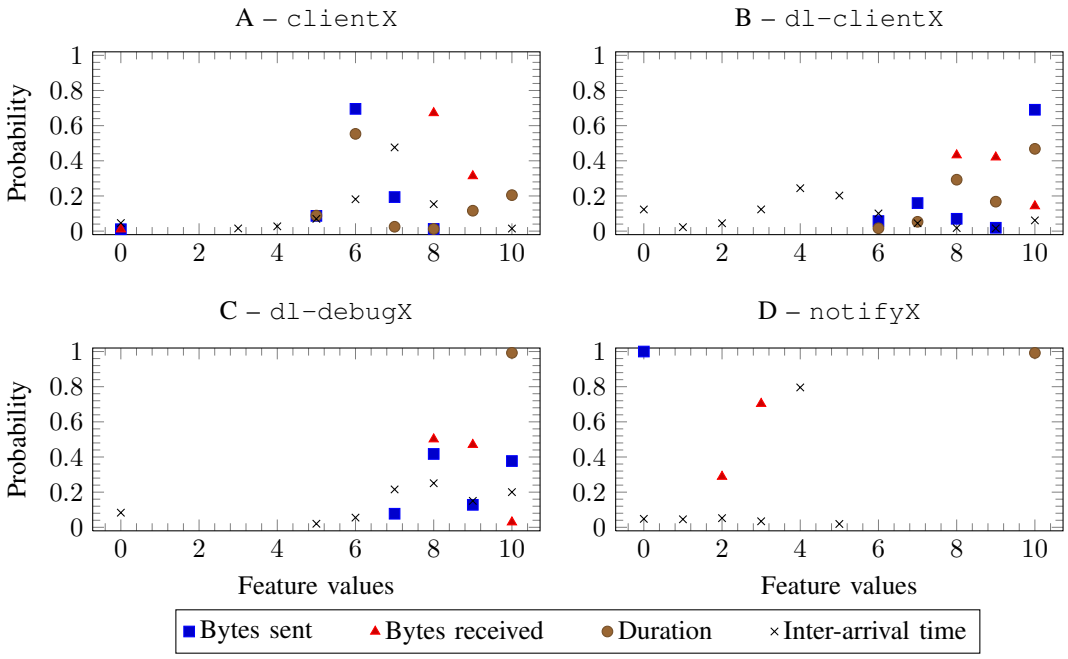

Fig. 3. Marginal probability distributions of the four observed features, estimated from fingerprints of Dropbox servers found in the four largest clusters.

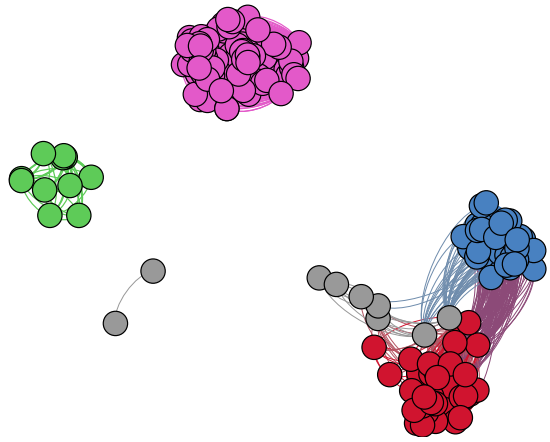

Fig. 4. Cosine similarity graph for the Dropbox servers. Servers from the four ground truth groups are highlighted by different colors: `clientX` servers are purple, `dl-clientX` servers are red, `dl-debugX` are blue, and `notifyX` servers are green. The remaining 5% of servers are grey.

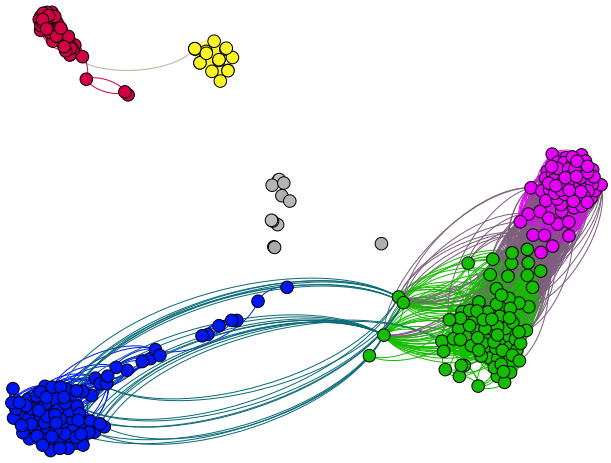

Fig. 5. Cosine similarity graph for the servers under the `live.com` second level domain. Five largest clusters discovered by the Louvain method are highlighted with color, the red cluster and the blue cluster contain servers probably involved in mail-related services while the green cluster and the purple cluster contain servers that handle instant messaging. The yellow cluster gathers servers related to "storage" services. The remaining nodes (6%) are grey.

to other types of traffic. In [15] network hosts are associated with different application types to classify the traffic of these hosts but the method also works at the transport layer.

The core idea of overlapping bins in soft histograms has been applied in computer vision, e.g., in [5], [16] or [17]. However, we are not aware of any application of soft histograms for modelling network traffic.

V. Conclusion

This paper has demonstrated that a highly accurate identification of groups of servers running the same parts of a service is possible without any prior knowledge about the service and without inspecting contents of packets. The core idea of the method is to derive servers' fingerprints from sizes, durations, and times of requests to them. These fingerprints are then clustered to identify groups of servers that are hosting similar applications. With the Louvain clustering, the method is unsupervised, parameterless, and directly applicable to encrypted traffic.

The method's performance was analyzed in detail on the task of identifying separate parts of the Dropbox service for which the ground truth was available. The analysis showed that the method is able to achieve results that highly match the correct solution determined by the ground truth. Furthermore, we demonstrated the proposed method by identifying groups of servers involved in the same functional parts of the former Windows Live platform with promising results, too.

We believe that the presented method can become basis of various tools used for monitoring and analysis of network traffic, e.g., as a part of intrusion detection systems (IDS) involved in detecting attacks and misuse of services. Understanding services' structure might improve behavioral models used in an IDS. In this way, the chance of detecting non-standard usage patterns of a service (which can indicate the malicious activity) can be significantly increased.

VI. Acknowledgement

The work of T. Pevný was also supported by the Czech Science Foundation (GAČR) under the project number 15-08916S.

References

[1] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside Dropbox: Understanding personal cloud storage services," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, 2012.

[2] "Squid," http://www.squid-cache.org.

[3] C. Wright, F. Monrose, and G. M. Masson, "On inferring application protocol behaviors in encrypted network traffic," *Journal of Machine Learning Research*, 2006.

[4] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, 1975.

[5] P.-E. Forssén, "Image analysis using soft histograms," in *Proceedings of the SSAB Symposium on Image Analysis: Norrköping*, 2001.

[6] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing (3rd Ed.): Principles, Algorithms, and Applications*. Prentice-Hall, Inc., 1996.

[7] V. Blondel, J. Guillaume, R. Lambiotte, and E. Mech, "Fast unfolding of communities in large networks," *J. Stat. Mech*, 2008.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2001.

[9] W. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, 1971.

[10] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, 1985.

[11] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009.

[12] M. Jacomy, S. Heymann, T. Venturini, and M. Bastian, "Forceatlas2, a continuous graph layout algorithm for handy network visualization," *Medialab center of research*, 2011.

[13] M. Crotti, M. Dusi, F. Gringoli, and L. Salgarelli, "Traffic classification through simple statistical fingerprinting," *SIGCOMM Comput. Commun. Rev.*, 2007.

[14] D. Schatzmann, W. Mühlbauer, T. Spyropoulos, and X. Dimitropoulos, "Digging into HTTPS: Flow-based classification of webmail traffic," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, 2010.

[15] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel traffic classification in the dark," *SIGCOMM Comput. Commun. Rev.*, 2005.

[16] V. Chandrasekhar, G. Takacs, D. Chen, S. S. Tsai, Y. Reznik, R. Grzeszczuk, and B. Girod, "Compressed histogram of gradients: A low-bitrate descriptor," *Int. J. Comput. Vision*, 2012.

[17] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Computer Vision and Pattern Recognition*, 2008.