What's my App?: ML-based classification of RTC applications

What's my App? ML-based classification of RTC applications

Dena Markudova†, Martino Trevisan†, Paolo Garza†,
Michela Meo†, Maurizio M. Munafò†, Giovanna Carofiglio ‡,
†Politecnico di Torino, ‡Cisco Systems Inc.
first.last@polito.it, gcarofig@cisco.com

ABSTRACT

With the spread of broadband Internet, Real-Time Communication (RTC) platforms have become increasingly popular and have transformed the way people communicate. Thus, it is fundamental that the network adopts traffic management policies that ensure appropriate Quality of Experience to users of RTC applications. A key step for this is the identification of the applications behind RTC traffic, which in turn allows to allocate adequate resources and make decisions based on the specific application's requirements.

In this paper, we introduce a machine learning-based system for identifying the traffic of RTC applications. It builds on the domains contacted before starting a call and leverages techniques from Natural Language Processing (NLP) to build meaningful features. Our system works in real-time and is robust to the peculiarities of the RTP implementations of different applications, since it uses only control traffic. Experimental results show that our approach classifies 5 well-known meeting applications with an F1 score of 0.89.

Keywords

RTC applications; Classification; Machine Learning; VoIP

1. INTRODUCTION

In recent years, the Internet has gained unprecedented popularity, reaching billions of users and providing a reliable means for a large plethora of services. Moreover, in the last decade, we witnessed the spread of social networks and chat applications [15], which revolutionized the way people communicate. The extensiveness and performance of today's networks also fostered the adoption of RTC platforms, which allow for audio and video calls, thus avoiding the cost of the traditional telephony. Recently, dozens of new RTC platforms have appeared [11], competing in a world where video calls are part of the normal business and leisure routine. Nowadays, they are massively adopted and companies pay subscriptions for premium plans.

In this context, it is not trivial for in-network devices to correctly recognize the RTC applications behind the traffic, which is fundamental for the proper network operation. Indeed, correct classification of such traffic is very important for adequate traffic engineering, in both ISP networks to ensure users' Quality of Experience (QoE) and corporate scenarios, in which, for instance, accredited services must be prioritized and the other segregated. The spread of encryption [9] makes it difficult to use classical approaches based on mere Deep Packet Inspection (DPI), which fall short in the case the RTC applications adopt TLS, DTLS or other secure protocols, which is very often the case.

To regain visibility on network traffic, the research community has turned towards Machine Learning (ML) for traffic classification [10]. These efforts target specific fields of applications, such as mobile apps [1], web traffic [13], network performance [2] or users' QoE [3, 12]. DNS has also been used extensively to mark traffic [4, 7]. Regarding RTC traffic, many works propose techniques for identification among other traffic categories [5, 6], while in this work we aim to go a step further.

In this paper, we propose a novel methodology to unveil the applications behind RTC traffic. We base our approach on the domains applications contact prior to set up a call, that we use as a signature to classify RTC streams in live traffic.[1] We employ NLP techniques to model how these domains appear in the traffic and use them to extract meaningful features that we then feed to Machine Learning (ML) classifiers. Exploiting a large dataset containing more than 230 packet traces, we evaluate the performance of our methodology when classifying the traffic of 5 RTC applications. Our results show that it is possible to obtain good performance, reaching an F1 score as high as 0.89. Moreover, our approach is based solely on the domains clients resolve *prior* to start an RTC call, allowing the system to classify new streams with virtually zero delay. Indeed, our system does not need any feature coming from the RTC stream itself, making it robust to, e.g., applications resorting to peer-to-peer communication between clients.

2. SYSTEM ARCHITECTURE

Our goal is to identify the meeting software that is behind an RTC stream when observing live network traffic. In other words, whenever we find a client starting an RTC session, we want to classify it as *generated* by, e.g., application A. For us, an *RTC session* is a media stream of an RTC application, carried, in our cases, always over the RTP protocol.

Rather than extracting features directly from the RTC streams or leveraging the server-side IP address, we target the domains the client resolves *prior* to start the session. We opt for this approach for a twofold reason. First, nowadays large companies rely on Content Delivery Networks to serve

[1]We use the term *domain* throughout the paper, meaning Fully Qualified Domain Name.

Workshop on AI in Networks and Distributed Systems (WAIN) 2020
Milan, Italy

Table 1: Dataset overview

Application	Webex Teams	Microsoft Teams	Skype	Jitsi	Zoom
No. training	27	47	35	35	30
No. testing	12	17	11	12	13

their content and on Cloud Providers to host their infrastructure, making simple approaches based on the enumeration of the server IP addresses or ranges ineffective. Second, RTC applications are known to rely on peer-to-peer communication between participants when possible. This again makes it difficult to leverage IP address and ports numbers, typically randomized by NATs. Also, the behavioral features of the streams are subject to extreme variability due to the diverse possible network conditions, discouraging their use for classification.

As such, we want to feed our classifier the domains resolved by the client before starting the RTC session. In this way, we target the control traffic of the application, that is typically exchanged over the TLS protocol. More specifically, we extract the Server Name Indication contained in the Client Hello messages, revealing in-clear the domain name of the server. This said, whenever we encounter an RTC stream in the traffic, we collect the *bag* of domains the client resolved in the previous ΔT seconds.

Training Methodology. We train our system on a collection of packet traces, each one containing a single RTC call using a known application – i.e., we know the ground truth. In Figure 1, we show an overview of the training steps. We collect the domains contacted by the client in the ΔT seconds before the call begins. Then, we prepare the dataset to be used to train a ML-classifier – i.e., we *vectorize* the data. This means we turn the textual domains into numeric features, and, as such, each domain results in a column of the dataset, which assumes the usual tabular format. At this stage, we merge the data of all packet traces in a single dataset, where each row represents an RTC call, and a value v greater than 0 in a cell means the domain corresponding to the column has been observed v times.

Our goal is to find, for each application, the domains that are more useful for its classification, because they are used, for example, for signaling, session setup, login, etc. To this end, we rely on NLP techniques, which have been proved to be useful to find the terms that better characterize each document within a corpus. In our system, we opt for the $tf-idf$ analysis [8], which is traditionally used to describe *documents* in a collection with the most representative terms. Given a particular term and a document, the $tf-idf$ is computed as the product of the frequency of the term in the given document (tf) and the inverse of the frequency at which the term appears in distinct documents (idf). While tf estimates how well the given term describes the document, idf captures the term's capacity to discriminate the document from others. In our context, we use $tf-idf$ to identify the domains that better characterize an application. The intuition is that if a domain appears in the majority of the traces, it is less important than a domain appearing in only a few of them. Using the $tf-idf$, we process the feature matrix so that each cell value v is replaced by the results of $tf-idf$ on the dataset.

To help the $tf-idf$ get rid of noise, we apply a threshold

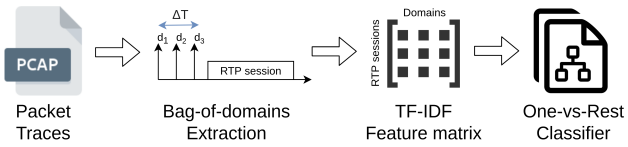

Figure 1: Scheme of our training methodology.

that each domain must reach to be included. We want to avoid very rare domains, appearing in one or a few traces, that obtain, by chance, high $tf-idf$ scores due to a high idf (because they are infrequent) and a high tf (because they have appeared many times in a trace). To this end, we set a threshold $MinFreq$, below which a domain is discarded. If a domain appears in a fraction of packet traces smaller than $MinFreq$, we flag it as noise and neglect it.

Once we obtain the feature matrix, we use classifiers to decide between the RTC applications. Remember that we suppose our training set includes a ground truth, indicating for each row (a session), the employed application. We use the one-vs.-rest strategy, which consists of fitting one binary classifier per class. We opt for this schema to improve robustness if classifiers are used in the wild, with potentially new unknown applications. By using one-vs.-rest, we force each classifier to focus on a single application, leaving all the rest in the *Other* class. This also improves the interpretability, since it allows us to gain knowledge about each class by looking at its specific classifier.

RTC Stream Classification. We use the obtained classifiers to identify the applications behind live network traffic. At classification time, we build the bag of domains whenever an RTC session begins. We then build the $tf-idf$ vector from the bag, which means using as tf the frequency at which domains have appeared and as idf the values we computed at training time. If a domain was not seen at training time, and, as such, we have no idf value, it is discarded. On the obtained feature vector, we run the classification models previously trained. We try five different algorithms: tree-based classifiers [Decision Tree (DT) and Random Forest (RF)], k-Nearest Neighbors (kNN), which classifies points based on proximity to other data points, Support Vector Machine (SVM), which instead constructs a hyperplane in high dimensional space to separate the data points and Gaussian Naïve Bayes (GNB) as a generative probability model. We then compare their performance under different conditions. Moreover, in Section 4.1, we explore the impact of the system parameters (ΔT and $MinFreq$ among all) on the classification performance. The system is easily parallelizable, since it works on a per-client basis.

3. DATASET

We rely on a dataset collected by a set of 15 volunteers, which recorded packet traces whenever they made a call during the first six months of 2020. Important to our analysis, the volunteers begin capturing the traffic on their equipment *before* starting the RTC application. In total, we collect 239 traffic traces from five meeting applications: Webex Teams, Microsoft Teams, Skype, Jitsi and Zoom. Note that it is hard to do automatic collection of such data, because it is not possible to rely on well-known browser automation tools like Selenium when using native applications running on PCs. Moreover, collecting data in the wild improves the

Table 2: Examples of highly discriminative domains.

Application	Domains
Webex Teams	ciscospark.com webex.com
Microsoft Teams	area.microsoft.com teams.microsoft.com
Skype	area.microsoft.com skype.com
Jitsi	meet.jit.si
Zoom	zoom.us

robustness of our results, since the packet traces were collected under a diverse set of operating systems, user devices, application versions, etc. Out of 239 calls, we use 174 for training and 65 for testing. We explicitly use all the traces of a single individual either for training or for testing, with the goal of avoiding overfitting to specific features of a client. Due to the nature of our problem, one call translates to one data point for the classifiers. We provide a breakdown of the dataset in Table 1. Recalling that we have to resort to a non-automated data collection, we consider to have a fair amount of data in terms of number of calls. More importantly, results show that our system is very accurate even with a low number of data points.

We process the packet traces using Tstat [14] to obtain the bag of domains the client contacted before starting the RTC stream. For all applications, it is straightforward to identify the media streams as they all rely on RTP. Only for Zoom, we had to strip a custom encapsulation header using a simple command-line tool.[2]

4. RESULTS

4.1 Classification Performance

We now discuss the performance of our system in terms of its ability to classify the application generating RTC streams. We assess the performance using our test set composed only of packet traces collected by individuals that do not appear in the training set. We try different algorithms and finally opt for a Random Forest classifier, which gives us the highest F1 score.[3] However, we also evaluate the impact of the classification algorithm and parameters in Section 4.2.

In Figure 2, we show the confusion matrix obtained using the best configuration of parameters. By definition, a confusion matrix C is such that C_{i_j} is equal to the number of observations known to be in group i and predicted to be in group j. The diagonal represents the number of correctly classified samples. We also show the per-class recall and F1 score in the last two columns, as well as the precision in the bottom row. Looking at the figure, we observe that all classes exhibit an F1-score higher than 0.8 and 3 out of 5, a score higher than 0.9. Webex Teams is the only class with slightly lower precision (0.67), meaning the other applications are being confused with it, especially Jitsi. This is also why Jitsi exhibits a lower Recall (0.75) than the other classes. The overall macro-averaged F1 score is 0.89.[4] The Random Forest classifiers used to reach this score consist of as little as 100 trees and a maximum depth of 20.

[2]`https://github.com/marty90/rtc_pcap_cleaners`

[3]The *F1-Score* is the harmonic mean between the *Precision* and *Recall* of a class.

[4]The macro average is the mean of the scores of each class.

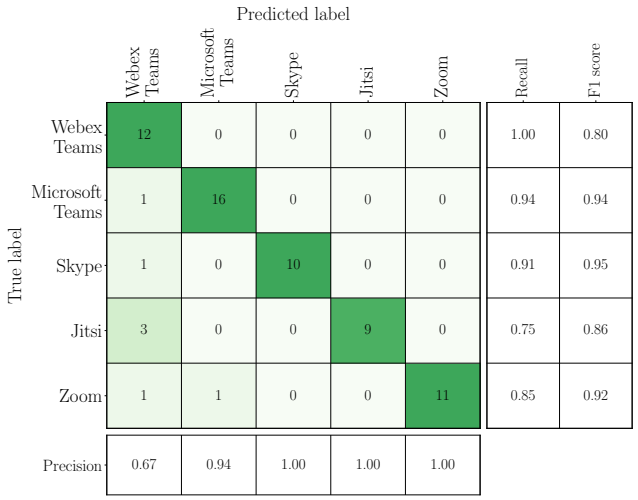

Figure 2: Confusion matrix with the best parameter choice.

Since we adopted the one-vs.-all classification schema, we can observe, for each application, which features (i.e., domains) are important for each class. Table 2 provides examples of the domains for which the Random Forest classifier indicates the high feature importance. To ease visualization, we truncate the domains at the second level. We qualitatively note how our approach is able to identify the domains per application. For example, Webex relies on `webex.com` and, not trivial, on `ciscospark.com`, the former name of the application. In other words, with our approach based on domains, we obtain *interpretable* results, especially if compared with other techniques based on packet-level features.

4.2 Parameter Tuning

We now briefly illustrate the impact of system parameters and different algorithms on the classification performance.

We first investigate the impact of ΔT – i.e., the duration of the period before the stream begins, that we use to collect the bag of domains. We vary it between 5s and 30s and show the results in Figure 3. We also use the figure to show the impact of different classification algorithms in terms of macro-averaged F1 score, represented on the y-axis. Note that all algorithms undergo hyperparameter tuning, using a 3-fold cross-validation on the training set, which however leads to small improvements. In general, the larger ΔT is, the more domains we collect and the better the final performance is. Indeed, the F1-score for 15s and above is altogether higher than the one for 5s or 10s. The performance also varies with different classification algorithms. kNN shows generally worse performance for higher ΔT, working better with a low number of features. DT and SVM achieve high F1 scores in general. GNB exhibits excellent results, showing the best performance of all for 15s or 20s, but the best result in absolute terms is achieved with RF, for $\Delta T = 25s$.

We now analyze the impact of the number of features, that we tune by setting $MinFreq$, the fraction of traces where we must see a domain to consider it valid. Intuitively, a large $MinFreq$ allows only few domains to appear as columns in the feature matrix, while small values allow also infrequent ones to be considered. We vary the threshold from 0.02 (appeared in a minimum of 5 traffic traces in our case) to 0.2 (48 traces minimum). We then use Figure 4 to show how the

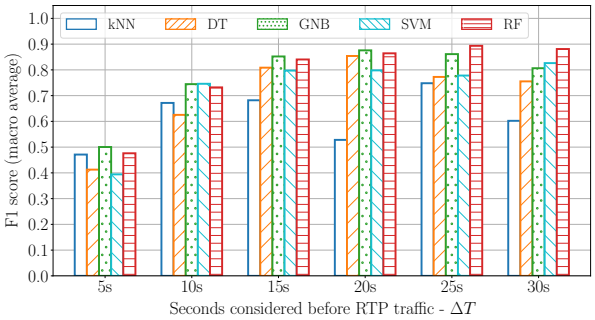

Figure 3: Performance of the five algorithms for different time bins.

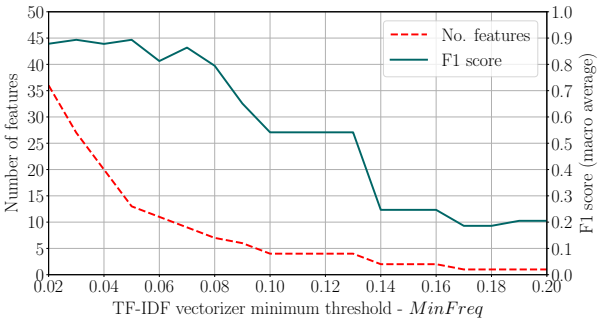

Figure 4: Change of performance and number of features when varying the cut-off.

number of features (left y-axis) and the overall performance (right y-axis) vary. Looking at the figure from right to left, we notice an increase in both number of features and F1-score as we decrease $MinFreq$, since we let more domains in. However, the two curves follow a different slope, with the F1 score exhibiting two main jumps. With $MinFreq$ 0.13 through 0.10, we use as little as 4 features, and the F1 score jumps at 0.54. Then, with $MinFreq = 0.07$, we have 9 features and an F1 score of 0.86, with small increases when further reducing $MinFreq$. Indeed, with very low $MinFreq$, we manually observe that the domains that we include are not related to the RTC applications but refer to background jobs of the workstations or parallel activity of users. To sum up, these results show that our approach requires a relatively low number of features to achieve good performance – in this case 9 features are enough for distinguishing up to five applications. Nonetheless, it is robust to noise and domains that may appear in the traffic by chance.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a ML-based system for classifying RTC applications. Given the domain names resolved prior to the start of a call, our system successfully distinguishes among 5 popular meeting applications with an F1 score of 0.89. It leverages NLP techniques and one-vs.-rest classifiers to build meaningful features and be robust to noisy data. Our approach can work in real-time and classifies flows with practically zero delay.

This paper presents only an initial work towards a meticulous system designed to make the network control plane aware of RTC traffic and manage the network accordingly. Our future challenges include the identification of RTC sessions and the exposure of various characteristics, from the type of application being used to the users' QoE.

Acknowledgements

This work has been supported by the SmartData@PoliTO center on Big Data and Data Science and Cisco Systems Inc.

6. REFERENCES

[1] Aceto, G., Ciuonzo, D., Montieri, A., and Pescapé, A. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Trans. on Network and Service Management 16*, 2 (2019), 445–458.

[2] Apiletti, D., Baralis, E., Cerquitelli, T., Garza, P., Giordano, D., Mellia, M., and Venturini, L. Selina: A self-learning insightful network analyzer. *IEEE Transactions on Network and Service Management 13*, 3 (2016), 696–710.

[3] Balachandran, A., Aggarwal, V., Halepovic, E., Pang, J., Seshan, S., Venkataraman, S., and Yan, H. Modeling web quality-of-experience on cellular networks. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (2014), pp. 213–224.

[4] Bermudez, I. N., Mellia, M., Munafo, M. M., Keralapura, R., and Nucci, A. Dns to the rescue: discerning content and services in a tangled web. In *Proceedings of the 2012 Internet Measurement Conference* (2012), pp. 413–426.

[5] Bonfiglio, D., Mellia, M., Meo, M., Rossi, D., and Tofanelli, P. Revealing skype traffic: when randomness plays with you. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications* (2007), pp. 37–48.

[6] Di Mauro, M., and Longo, M. Revealing encrypted webrtc traffic via machine learning tools. In *2015 12th International Joint Conference on e-Business and Telecommunications* (2015), vol. 04, pp. 259–266.

[7] Foremski, P., Callegari, C., and Pagano, M. Dns-class: immediate classification of ip flows using dns. *International Journal of Network Management 24*, 4 (2014), 272–288.

[8] Jones, K. S. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* (1972).

[9] Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M., Papagiannaki, K., and Steenkiste, P. The cost of the" s" in https. In *ACM International on Conf. on emerging Networking Experiments and Technologies* (2014).

[10] Nguyen, T. T., and Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Comm. surveys & tutorials 10*, 4 (2008), 56–76.

[11] Nisticò, A., Markudova, D., Trevisan, M., Meo, M., and Carofiglio, G. A comparative study of rtc applications. *To appear in the Proceedings of the 22nd IEEE International Symposium on Multimedia* (2020).

[12] Trevisan, M., Drago, I., and Mellia, M. Pain: A passive web performance indicator for isps. *Computer Networks 149* (2019), 115–126.

[13] Trevisan, M., Drago, I., Mellia, M., Song, H. H., and Baldi, M. What: A big data approach for accounting of modern web services. In *2016 IEEE Int. Conf. on Big Data (Big Data)* (2016), IEEE, pp. 2740–2745.

[14] Trevisan, M., Finamore, A., Mellia, M., Munafò, M., and Rossi, D. Traffic analysis with off-the-shelf hardware: Challenges and lessons learned. *IEEE Communications Magazine 55*, 3 (2017), 163–169.

[15] Trevisan, M., Giordano, D., Drago, I., Munafò, M. M., and Mellia, M. Five years at the edge: Watching internet from the isp network. *IEEE/ACM Transactions on Networking 28*, 2 (2020), 561–574.