Online Classification of RTC Traffic

Gianluca Perna[†], 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*—Real-time communication (RTC) platforms have become increasingly popular in the last decade, together with the spread of broadband Internet access. They are nowadays a fundamental means for connecting people and supporting the economy, which relies more and more on forms of remote working. In this context, it is particularly important to act at the network level to ensure adequate Quality of Experience (QoE) to users, where proper traffic management policies are essential to prioritize RTC traffic. This, in turn, requires in-network devices to identify RTC streams and the type of content they carry.**

In this paper, we propose a machine learning-based application to classify, in real-time, the media streams generated by RTC applications encapsulated in Secure Real Time Protocol (SRTP) flows. Using carefully tuned features extracted from packet characteristics, we train a model to classify streams into an ample set of classes, including media type (audio/video), video quality and redundant streams. To validate our approach, we use traffic from more than 88 hours of multi-party meeting calls made using the Cisco Webex Teams application. We reach an overall accuracy of 97% with a light-weight decision tree model, which makes decisions using only 1 second of traffic.

I. Introduction

In recent years, real-time communication (RTC) platforms for video calls and virtual meetings have become a fundamental pillar of leisure and business, helping people to stay connected and companies to save substantial travel costs. Their value was especially manifested during the months of self-isolation due to the COVID-19 pandemic, where online conferencing made it feasible for a lot of businesses to continue operations using remote working, with big economical benefits. This has been made possible by the Internet becoming ubiquitous and available bandwidth continuously growing [1]. During the early 2000s, Skype opened the business for RTC applications, in a scenario where most of the users were connected via cable modems, offering in general poor bandwidth and high latency. Nowadays, the market offers countless competing platforms for video calls, benefiting from the widespread adoption of broadband access. Each of them employs different technical solutions and network protocols despite the recent efforts for standardization, with WebRTC as the notable example[1].

In this context, it is fundamental to maximize the Quality of Experience (QoE) of users at the network level to avoid impairments, service misbehavior and, in turn, users' churn. The QoE depends on many factors, such as the quality of the connection of the participants, the network architecture and the in-network management. Classification of RTC traffic is the first and most important step towards effective traffic management, allowing in-network devices to monitor users' perceived QoE, and, if classification is made in real-time, to take proper actions to cope with possible deterioration. The widespread encryption of the Internet [2], with HTTPS carrying more than 90% of web traffic, has made it difficult for routers and middleboxes to separate traffic based on port numbers or mere deep packet inspection (DPI), which is the traditional way [3]. Indeed, most of the RTC platforms still rely on the RTP protocol [4] to stream the multimedia content in its encrypted version, *Secure* RTP (SRTP). It employs in-clear packet headers, but encrypts the media payload, making it hard to guess the type of content it carries. This calls for novel techniques, based on machine learning (ML), to re-obtain visibility on application traffic and help decision-making at routers. In real-time communication this could amount to distinguishing top priority flows from possibly less critical data exchange – e.g., audio as more important than video, presenter's media as more valuable than audience's.

In this paper, we propose a novel ML-based application for classifying, in real-time, the RTC media flows carried by RTP streams. Our approach is based on a few, yet well-chosen features, derived from the statistical properties of the traffic, which allow us to classify RTP streams to the type of content they carry. Our application identifies not only audio or video streams but also other properties of the media such as the video resolution or the use of Forward Error Correction (FEC). Our solution works with minimal delay, making a decision on a stream with as little as 1 second of traffic. It is designed to work as a software module to be plugged in network devices (e.g., routers), enabling fine-grained traffic management.

We build our study on the Cisco Webex Teams RTC application, which allows multi-party meetings with audio, video and screen sharing[2]. Using more than 88 hours of network traffic captured during real calls, we evaluate the impact of the feature choice and different classification algorithms. After a careful feature selection and using a light-weight decision tree classifier, we obtain an overall accuracy of 97%, with no big differences across classes. This work is the first step towards a comprehensive traffic management system for RTC based on ML, which enables application-level visibility at the

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

[1]https://webrtc.org/

[2]https://www.webex.com/team-collaboration.html

network control plane, providing highly-detailed information about the ongoing RTC sessions, the employed applications and the perceived QoE.

II. Dataset

In this section, we briefly describe the dataset we use throughout the paper. We focus on the Webex Teams RTC application, which allows calls between multiple participants with audio, video and screen sharing media. It is available as a standalone application for PC and mobile devices and it can also be used via browsers that support the WebRTC API set. Webex Teams uses the Selective Forwarding Unit (SFU) approach, in which participants send their multimedia content to a centralized server. The server then forwards the data, deciding which stream to send to each participant. The media streams are encapsulated using the SRTP protocol [4].[3] Each client opens a single UDP flow towards the server, in which multiple streams are multiplexed via a unique Synchronization Source Identifier (SSRC).

Using Webex Teams, we capture real calls made under different conditions, with a diverse number of participants (from two to ten), multimedia content (audio, video, screen sharing) and user equipment (PC, tablet or phone). The calls are made in a real environment in which the participants are connected on different networks, from various countries and employ different classes of devices, from Windows PCs to iPhones and Android phones. During each call, a participant captures the exchanged traffic and saves it in a `pcap` file.

In our classification problem, we target RTP streams defined as the tuple: source IP address, source port, destination IP address, destination port and RTP SSRC. In other words, we target a single stream, each carrying a particular multimedia content. We divide the streams into 7 classes: Audio, Low Quality (LQ) Video - 180p, Medium Quality (MQ) Video - 360p, High Quality (HQ) Video - 720p, Screen Sharing, FEC audio and FEC video. Indeed, Webex Teams uses FEC to mitigate packet losses, sending streams containing redundant information to be used by the receiver in case some packets are lost or contain errors. We observe FEC streams both for audio and video, and we are interested in identifying them as separate classes.

To obtain the ground truth, which maps each RTP stream to the content type, we employ the application logs of Webex Teams, that are automatically generated during each call. They contain per-second details on each stream, such as the type of media (audio, video or screen sharing), the video resolution, the number of frames per second etc. During each call, the participant who captures the traffic also stores the log file, that we later use to extract the ground truth. Note that we cannot use the RTP Payload Type field for this, since it is assigned dynamically.

In total, we collected approximately 88 hours of media traffic, exchanged during 27 calls. Each of them presents a different mixture of the above classes and contains the traffic generated by all participants as captured from the perspective of a single individual. Ten calls have only two participants, two are made with three participants and fifteen include more than three participants. In Table I we provide a breakdown of the dataset. For each class, we report the amount of data we collected, in seconds. The most represented classes are Audio, FEC audio and LQ video. While for audio this is somehow expected, the prevalence of LQ video is caused by the video thumbnails used by Webex Teams for showing inactive participants during calls with more than three participants. The least represented class is Screen Sharing, but the overall imbalance of the dataset is still limited, with the ratio between the support of the most and the least represented class being less than 6.

[3]In the remainder, we use the term *RTP* regardless it is used in its encrypted version *SRTP*, which maintains all the original headers in clear.

TABLE I: Dataset summary

Class	No. of seconds	
	Training	Testing
Audio	214 359	103 098
Video LQ	203 275	98 855
Video MQ	45 291	16 817
Video HQ	68 947	34 324
Screen Sharing	41 178	11 249
FEC Audio	146 705	51 393
FEC Video	50 341	3 486

III. Methodology

Our goal is to classify the RTP streams that we observe on the network to one of the seven classes reported in the previous section. We aim at achieving this in real-time – i.e., making a decision based solely on the traffic observed in a short time interval. Thus, our classification target is an RTP stream as observed during a short time bin (from 200 ms to 4 s). We recognize RTP with straightforward deep packet inspection matching the protocol headers. We then separate multiple media streams via their SSRC. We are not interested in the control traffic for e.g., session setup or login, and, as such, we neglect it. A single RTP stream typically results in several entries (one per each time bin), that we shall classify. We follow the typical machine learning approach. We first extract meaningful features from the data, guided by domain knowledge on network traffic and the RTP protocol. Then, we perform a two-step feature selection process, by first filtering out highly correlated features and then performing a recursive feature elimination. Finally, we train a machine learning classifier and evaluate its performance on an independent test set. The feature selection and algorithm training are done offline, while the system is designed to calculate features and classify new samples in real-time. The time it takes is equivalent to the chosen time bin plus the feature computation and algorithm run, whose execution time is negligible. Our code is written in Python and exploits the scikit-learn library [5] for machine learning.

Train/test methodology. To avoid overfitting and obtain robust results, we split our set of calls in a training set and a test set. As such, we evaluate the performance on data which are never used at training-time. Out of 27 calls, we use data from 22 calls for training and data from the remaining 5 calls for testing. We verify that each class is well-represented in both sets, and the test set accounts for approximately 30% of the overall duration of the calls. We perform feature selection and hyper-parameter tuning of the algorithms on the training set and we evaluate classification performance on the test set. As a performance indicator, we use the macro average (a simple mean) of the F1-scores of each class.[4] For some analysis, we also consider the accuracy as a concise index for the overall performance, since the classes are not strongly imbalanced.[5]

Feature extraction. We extract features from the raw packets, separately for each RTP stream and time bin. Our features are built upon the fields of the RTP protocol and take into account its operation. We sketch our approach in Figure 1. We consider five groups of features, reported in the central column of the figure, in bold. They include packet characteristics (length, timing, volume) and the RTP timestamp field which indicates the instant at which the content is generated. RTP has a few other fields, which indicate mainly header extensions, and we do not include them, as they are very application-specific. Since two of the selected quantities represent time, we only consider their relative variation across packets, as the absolute values are useless in our context. For packet size, we use both absolute and relative values. We extract these values for all packets and compute various statistical indexes to build the final features, such as range, mean, standard deviation, percentiles, third and fourth moments, etc. Since we observe that the same values often recur over packets, we also include features to measure the number of unique values, the percentage of occurrences of the most common value (mode) and the ratio between the minimum value and the range. Finally, we also consider the traffic volume in terms of the number of packets and bits seen in the time bin.

Note that our classifier is designed to work in real-time, so we build features that can be computed on-the-fly considering only the packets observed in a time bin. Intuitively, the smaller the time bin is, the faster the stream is classified. However, with larger time bins, the features are more significant, since they are computed on a larger set of packets. In Section IV, we explore this trade-off. Finally, note that we also avoid features that require the association of multiple flows to keep our design simple and scalable.

Feature selection. In total, we extract 95 features, derived from the four empirical distributions and traffic volume of the RTP stream. To remove those that are redundant and shrink the overall number of features, we perform a two-step feature selection process:

1) *Correlation analysis*: We perform a first selection of the features by measuring the correlation between each pair of them. Whenever we find a Pearson correlation coefficient higher than 0.9, we keep only one of the two features at random. Roughly half of the features are eliminated in this step.
2) *Recursive Feature Elimination using the ExtraTree algorithm*: We use the Recursive Feature Elimination (RFE) approach [6] to further refine our list of features, maintaining only those that are most useful for our classification problem. Using RFE, we train an Extra-Tree classifier on the training set and rank the features by importance, as provided by the algorithm. We then eliminate the one with the lowest value. This procedure is recursively repeated until we reach the minimum number of features and the best performance. Note that tree-based feature ranking is known to be biased in the case of groups of correlated features [7]. As such, our first step (correlation analysis) is of key importance to make RFE work properly.

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

[5]The *accuracy* is the share of correct predictions over the total.

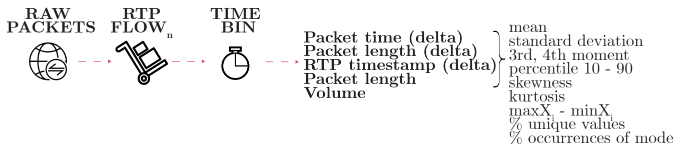


Fig. 1: Features derived from packets.

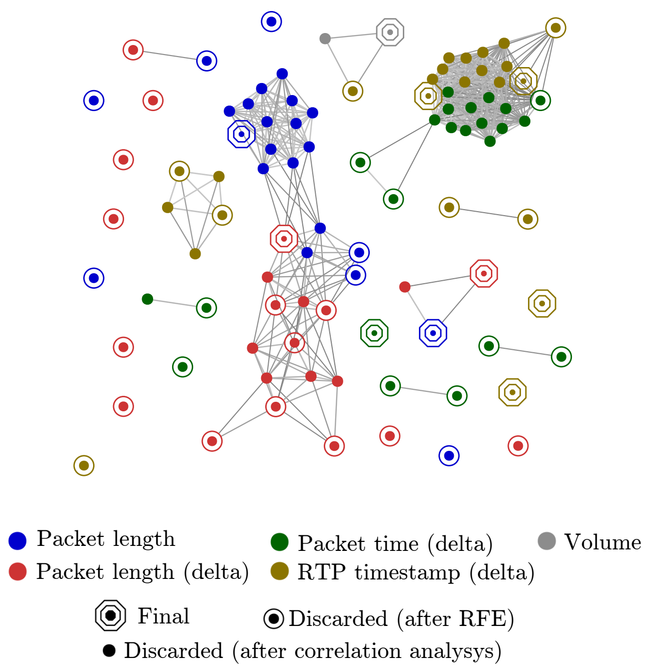


Fig. 2: Graph representing the correlation between features. The color indicates the feature set, the shape whether the feature is kept after feature selection, the distance represents the correlation.

We illustrate the whole process graphically with Figure 2, which shows the initial 95 features in the form of a graph. Each node represents a feature, and the length of edges is (roughly) inversely proportional to the correlation among pairs

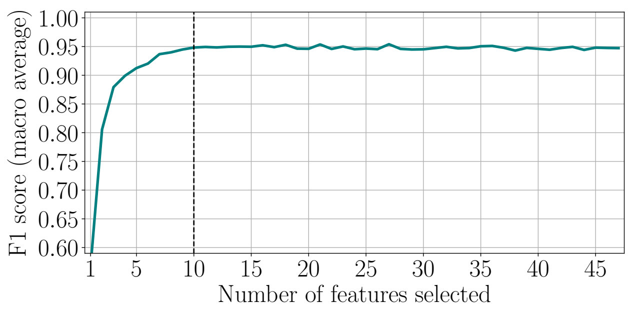


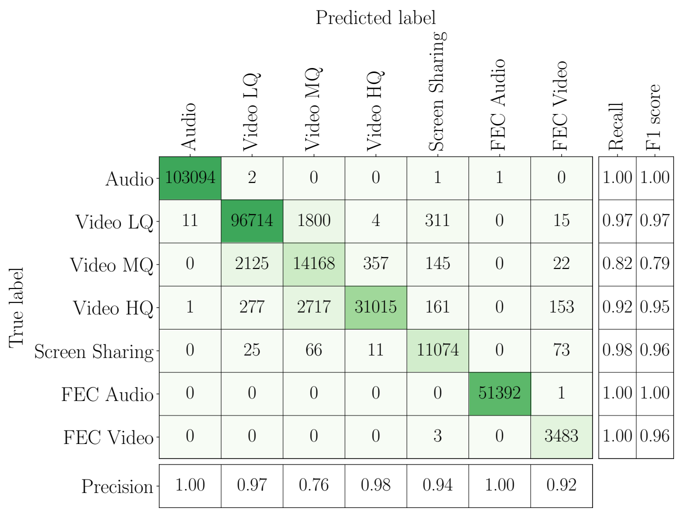
Fig. 3: Mean accuracy when varying the number of features

in absolute value – i.e., highly correlated features reside close to each other. For the sake of visualization, we show only edges for which the correlation is higher than 0.5 (in absolute value). The feature sets are represented with different colors, while the shape of each node indicates whether a feature is maintained or discarded at one of the selection steps: a circle means the feature has been discarded after the correlation analysis, a double circle indicates feature discarded with RFE, and an octagon means it passed both steps and has been kept in the final phase.

We first notice that the correlation analysis step maintains all features which are poorly correlated with other ones – all nodes without edges are either double circles or octagons. On the contrary, among groups of highly correlated features, only a few samples are kept. For example, the dense community on the top right of the figure groups the percentiles of the packet inter-arrival time and RTP timestamp delta. We keep only a few of them.

The first step of the feature selection shrinks our set from 95 to 45 features. We then run RFE to find only those that prove to be useful for our classification problem. We start by training an ExtraTree classifier on 45 features, running a 3-fold cross-validation to evaluate how accurate the obtained model is. Then, we eliminate the feature ranked as least important and repeat this operation until we notice that the classification performance starts decreasing. In Figure 3, we show how the average F1 score varies when removing an increasing number of features. When we use all 45 features, we obtain approximately 95% accuracy. The performance is almost stable (with minimal variations) until we reach 10 – i.e., we eliminate 35 features. Then, the accuracy starts decreasing consistently. Analyzing the curve, we decide to set the final number of features to 10. Interestingly, we notice that each feature group is represented in the set of the final features (there is an octagon for each group of features with the same color in Figure 2). Among the final features, we find the packet length (40th percentile), the range of the RTP timestamp delta, the range of packet length delta and the number of packets. Intuitively, for each characteristic of the packets, we keep one statistical property of its distribution.

Multi-class classification. Using the features that we obtain from the previous steps, we try different classification algorithms to find the one that properly trades-off performance and



True label \ Predicted label	Audio	Video LQ	Video MQ	Video HQ	Screen Sharing	FEC Audio	FEC Video	Recall	F1 score
Audio	103094	2	0	0	1	1	0	1.00	1.00
Video LQ	11	96714	1800	4	311	0	15	0.97	0.97
Video MQ	0	2125	14168	357	145	0	22	0.82	0.79
Video HQ	1	277	2717	31015	161	0	153	0.92	0.95
Screen Sharing	0	25	66	11	11074	0	73	0.98	0.96
FEC Audio	0	0	0	0	0	51392	1	1.00	1.00
FEC Video	0	0	0	0	3	0	3483	1.00	0.96
Precision	1.00	0.97	0.76	0.98	0.94	1.00	0.92		

Fig. 4: Confusion matrix when using a Decision Tree classifier with 1 s time bins.

simplicity. The algorithms we consider are: tree-based classifiers [Decision Tree (DT) and Random Forest (RF)], k-Nearest Neighbors (k-NN), which classifies points based on proximity to other data points and Gaussian Naïve Bayes (GNB) as a generative probability model. For each one, we perform hyper-parameter tuning with a 3-fold cross-validation, using uniquely the training set. We then assess their performance on the separate test set, using the macro-averaged F1-score as a performance indicator. Indeed, in Section IV, we show to what extent the algorithm choice impacts the classification results.

IV. Experimental Results

In this section, we present the experimental results for the entire classification problem. We first discuss the overall classification performance and then focus on the impact of the time bin duration and algorithm choice. Finally, we quantify the amount of training data required to achieve good performance. All the presented results are obtained training classification models on the training set and evaluating their performance on the independent test set.

We try different classification algorithms and finally opt to use a Decision Tree classifier which provides good performance and a simple model. Running hyper-parameter tuning, we find that best results are achieved when using the Gini index as a purity measure and posing a minimum of 45 entries to allow a node split, to avoid overfitting. In Figure 4 we show the confusion matrix obtained on the test set using a 1 s time bin. 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 precision in the bottom row. Looking at the figure, we observe that 6 out of 7 classes have an F1-Score above 0.95, and as such, high precision and recall. Audio and FEC audio are the classes with the best permanence, and only a handful of samples are misclassified, suggesting that audio streams are in general

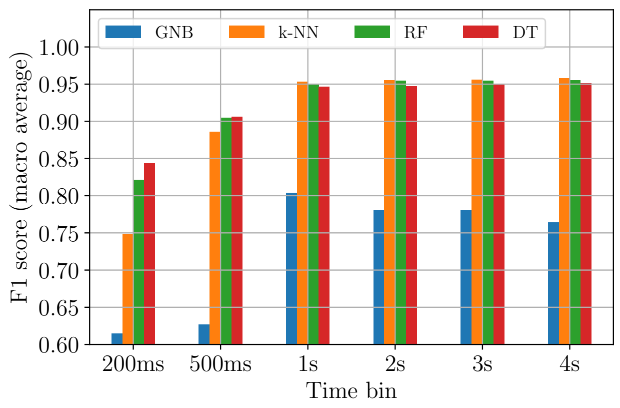


Fig. 5: Performance of the four algorithms for different time bins.

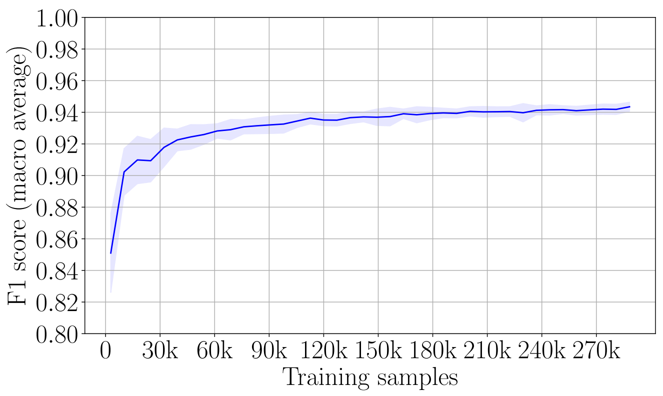


Fig. 6: Learning curve: Relationship between the number of training samples and the score.

easy to isolate. The class with the worst performance is video LQ, with an F1 score of 0.79. Indeed, the confusion matrix reveals that the three different video qualities are sometimes confused. Although this is a flaw of our classification model, we tolerate this behavior given the similar nature of the three classes. Moreover, Cisco Webex (and most RTC applications) employs variable bitrate video codes that result in diverse network traffic. Overall, 97.3% of the samples are correctly classified, and the average F1 score is 0.94.

We now illustrate the impact of the time bin duration on the classification performance. Indeed, we are interested in classifying a stream as quickly as possible, without sacrificing accuracy. Figure 5 shows how the performance varies with different time bin durations, from 200 ms to 4 s. We provide results when using 4 classification algorithms, and the y-axis reports the F1-score we obtain, averaged over the classes. We notice that, in general, we get better results with larger time bins. This is no surprise, as the features are calculated over more significant sets of packets. For example, in 200 ms of an audio stream, typically only 10 packets are generated. However, this effect stabilizes for values larger than 1 s, meaning that such a time frame is enough to gather precise information about a stream. Looking at the figure, we can also compare the performance of different classification algorithms. We notice no significant differences, except for Gaussian Naïve Bayes which shows somewhat worse performance, likely rooted in the simplicity of the model – note that the lowest F1 score is 0.62. This confirms that our careful feature engineering and selection make the results robust to the choice of algorithm. We finally opt to use a Decision Tree for its simplicity, interpretability and speed. Random Forest leads to similar results but requires the use of several trees in parallel, 100 in our case. k-NN performs similarly as well, but requires the model to store the entire training set with significant memory consumption. Using a decision tree, instead, the model has size of only a few kB.

Finally, we investigate how much training data we need to obtain good performance. To this end, we train a classification model multiple times, increasing the number of samples from the training set every time. In this experiment, we use a Decision Tree classifier with time bins of 1 s. Figure 6 shows the classification performance versus the size of the training set. For a fair comparison, we always use the same number of entries per each class. Indeed, for each value n on the x-axis, we sample n entries equally distributed among the different classes. As such, the x scale is limited by the support of less represented class of the training set (Screen Sharing), multiplied by the number of classes.[6] The y-axis reports the macro average of the F1 scores we obtain when classifying the test set. For each value on the x-axis, we perform 50 runs and plot the average (blue line) and the standard deviation (blue area) of the metric. Watching the figure, we notice that the performance improves quickly with the training set size. With only 60 k samples, the F1 score is already above 0.92. This suggests that the features we extract and the nature of the problem do not require a large dataset to obtain a reliable model. Indeed, 60 k samples correspond to approximately 16 hours of media streams, less than 2.5 hours per class. With a larger dataset, the F1-score increases further, but without drastic improvements. When using the maximum number of training samples, the performance reaches an average F1 score of 0.94, reproducing what we detail in Figure 4. We conclude that a training set including a few hours of call is enough to train a robust model.

V. Related Work

Network traffic classification has been extensively studied since the introduction of the Internet [3]. Due to widespread encryption and proprietary use of protocols, traditional approaches based on DPI and port numbers fall short, and the current research tends to leverage statistical traffic characteristics and machine learning techniques [8]. Recent efforts aim at identifying the web services [9] or mobile applications [10] behind network traffic, predicting the QoE of web [11], video [12] or smartphone [13] users.

[6]Since our training set includes approximately 40 k seconds of screen sharing, the x-axis ends with 280 k samples.

Regarding RTC traffic, many works propose techniques for identification among other traffic categories. The authors of [14] use a stochastic characterization of Skype traffic to obtain an ML-based model to be used for classification. In [15], UDP flows are classified with SVM models using statistical signatures of the payload, to various classes, including Skype and RTP-based traffic. The authors of [16] use statistical properties of RTP to differentiate between voice and data traffic. The authors of [17] propose a method to detect WebRTC sessions at run-time based on statistical pattern recognition. Finally, some approaches target application signaling mechanisms, for identifying e.g., Skype traffic through in-clear headers exchanged during session setup [18].

Fewer works focus on the classification of the media streams carried by RTP flows. In [19], the authors use packet length as a discriminator between audio, video streaming, browser and chat traffic. As a model, they opt for an interpretable Decision Tree. Authors of [20] target three Variable Bit Rate (VBR) audio codecs, using packet size and packet payload entropy as features and comparing a k-NN, C4.5 and Naïve Bayes algorithms. The authors of [21] identify the codecs used for compression of audio/video, again based on statistical features and RTP Payload Type. Our work goes in this direction, aiming at unveiling the nature of media streams. Differently from previous works, we classify streams into a rich set of classes including media type (audio and video), video quality and redundant data (FEC). Moreover, we are the first to explicitly target real-time applications with a 1 second (or shorter) classification delay, while the past approaches base their decision on the characteristics of the entire flow.

VI. Conclusion and Future Work

In this paper, we presented an ML-based system for classifying, in real-time, the media streams generated by RTC applications. Given a media stream carried inside the RTP protocol, we can distinguish seven different classes of media, including different video qualities and redundant data used to mitigate losses (i.e., FEC). We carefully engineered features based on packet characteristics and designed the system to work with a minimal set of features using a light yet accurate tree-based model. We used Cisco Webex Teams as a case study and showed that we obtain high classification accuracy with only 1 s of classification delay.

This work is only a first step towards a thorough system designed to gather fine-grained information from the RTC traffic at the network level. We aim at making it possible for the network control plane to make decisions on traffic with the awareness of RTC traffic and its characteristics. We would first identify and then provide detailed information about all RTC sessions on the network, like the type of applications in use, number of concurrent meetings going on, and finally the QoE perceived by the users.

References

[1] M. Trevisan, D. Giordano, I. Drago, M. M. Munafò, and M. Mellia, "Five years at the edge: Watching internet from the isp network," *IEEE/ACM Trans. on Networking*, vol. 28, no. 2, pp. 561–574, 2020.

[2] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The cost of the" s" in https," in *Proc. of the 10th ACM International on Conf. on emerging Networking Experiments and Technologies*, pp. 133–140, 2014.

[3] M. Finsterbusch, C. Richter, E. Rocha, J. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2014.

[4] R. Frederick, S. L. Casner, V. Jacobson, and H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications." RFC 1889, Jan. 1996.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[7] L. Toloşi and T. Lengauer, "Classification with correlated features: unreliability of feature ranking and solutions," *Bioinformatics*, vol. 27, no. 14, pp. 1986–1994, 2011.

[8] T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE communications surveys & tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

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

[10] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–8, IEEE, 2018.

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

[12] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A machine learning approach to classifying youtube qoe based on encrypted network traffic," *Multimedia tools and applications*, vol. 76, no. 21, pp. 22267–22301, 2017.

[13] P. Casas, A. D'Alconzo, F. Wamser, M. Seufert, B. Gardlo, A. Schwind, P. Tran-Gia, and R. Schatz, "Predicting qoe in cellular networks using machine learning and in-smartphone measurements," in *Ninth International Conf. on Quality of Multimedia Experience*, pp. 1–6, IEEE, 2017.

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

[15] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "Kiss: Stochastic packet inspection classifier for udp traffic," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1505–1515, 2010.

[16] A. S. Buyukkayhan, A. Kavak, and E. Yaprak, "Differentiating voice and data traffic using statistical properties," in *2013 International Conference on Electronics, Computer and Computation (ICECCO)*, pp. 76–79, 2013.

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

[18] T. Sinam, I. T. Singh, P. Lamabam, N. N. Devi, and S. Nandi, "A technique for classification of voip flows in udp media streams using voip signalling traffic," in *2014 IEEE International Advance Computing Conference (IACC)*, pp. 354–359, 2014.

[19] M. C. S, S. H, and T. E. Somu, "Network traffic classification by packet length signature extraction," in *2019 IEEE International WIE Conference on Electrical and Computer Engineering*, pp. 1–4, 2019.

[20] P. Choudhury, K. R. Prasanna Kumar, G. Athithan, and S. Nandi, "Analysis of vbr coded voip for traffic classification," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 90–95, 2013.

[21] P. Matousek, O. Rysavy, and M. Kmet, "Fast rtp detection and codecs classification in internet traffic," *Journal of Digital Forensics, Security and Law*, 01 2014.