From Encrypted Video Traces to Viewport Classification

Othmane Belmoukadam, Chadi Barakat

From Encrypted Video Traces to Viewport Classification

Othmane Belmoukadam
Université Côte d'Azur, Inria, France
othmane.belmoukadam@inria.fr

Chadi Barakat
Université Côte d'Azur, Inria, France
chadi.barakat@inria.fr

***Abstract*—The Internet has changed drastically in recent years, multiple novel applications and services have emerged, all about consuming digital content. In parallel, users are no longer satisfied by the Internet's best effort service, instead, they expect a seamless service of high quality from the side of the network. This has increased the pressure on Internet service providers (ISP) in their effort to efficiently engineer their traffic and improve their end-users' experience. Content providers from their side, and to further protect the content of their customers, have shifted towards end-to-end encryption (e.g., TLS/SSL), which has complicated even further the task of ISPs in handling the traffic in their network. The challenge is notable for video streaming traffic which is driving the Internet traffic growth, and which imposes tight constraints on the quality of service provided by the network depending on the content of the video stream and the equipment on the end-user premises. Video streaming relies on the dynamic adaptive streaming over HTTP (DASH) protocol which takes into consideration the underlying network conditions (e.g., delay, loss rate, and throughput) and the viewport capacity (e.g., screen resolution) to improve the experience of the end user in the limit of available resources. Nevertheless, knowing the reality of the encrypted video traffic is of great help to ISPs as it allows taking appropriate network management actions. In this work, we propose an experimental framework able to infer fine-grained video flow information such as chunk sizes from encrypted YouTube video traces. We also present a novel technique to separate video and audio chunks from encrypted traces based on Gaussian Mixture Models (GMM). We evaluate our technique with real chunk sizes (Audio/Video) collected through the browser using the Chrome Web Request API [1]. Then, we leverage these results and our dataset to train a model able to predict the class of viewport (either *SD* or *HD*) per video session with an average 92% accuracy and 85% F1 score.**

***Index Terms*—Video streaming, video chunk size, viewport resolution, YouTube encrypted traces, machine learning.**

I. Introduction

Video traffic is the major contributor to the global Internet traffic and the main source of pressure on the Internet infrastructure. By 2023, video traffic will account for 73% of the global mobile data traffic [2]. Besides major video streaming platforms such as YouTube and Netflix, it also includes sharing video through social networks, or offering direct video connectivity (e.g., Messenger). Video transmission over HTTP has changed over the years, it started with videos downloaded completely by the clients before being played out, to videos progressively streamed to the clients at a fixed resolution. Recently, adaptive streaming over HTTP has been widely adopted to automatically tune the streamed video resolution as a function of the available network resources. For instance, Dynamic Adaptive Streaming over HTTP (DASH) protocol [3], [4] allows adapting the video quality to the available bandwidth and the client terminal characteristics (e.g., viewport resolution). In plain, DASH divides the video into segments (e.g., 2 - 10 seconds) with each segment available in different quality versions. Details on the video availability at the server are stored in the Media Presentation Description (MPD) template which describes the video segments in terms of coding standard and bit rate and is shared with the client at the beginning of every video session. The choice between the different video representations is done by the DASH client taking into consideration the network state and the terminal capacity with as objective the smoothest possible playout without excess bandwidth usage.

In light of this rapid growth, Internet service providers feel more pressure to optimize their network and meet the expectations of their end users. They give high importance to video traffic engineering which requires the ability to infer the context of the video streaming such as the characteristics of the terminal on which the video is played out and the resolution of the streamed video. However, this is getting more difficult because of the end-to-end encryption adopted by major video streaming platforms (e.g., YouTube, Netflix and Amazon) [5]. For example, to prioritize or load balance traffic efficiently, ISPs need information on end-user QoE rather than just capturing the network Quality of Service (QoS). But, video QoE is dependent on the content itself (the video bitrate and resolution) and on the application level QoS metrics such as start-up delay, duration of stalls and resolution switches [6]–[8]. It also depends on the resolution of the viewport on which the video is played out [9]. All this information is unfortunately hard to obtain from encrypted video traffic, making their inference an important challenge to solve.

Previous studies manage to highlight some key QoE metrics such as stalls, average resolution, and representation fluctuations from measurements on the encrypted traffic and by using machine learning [10]. These studies mainly rely on the fact that application level metrics are proved to be tightly correlated to the network level QoS [11]–[13]. However, they overlook an important information regarding the resolution of

This work is supported by the French National Research Agency under grant BottleNet no. ANR-15-CE25-0013 and by Inria within the Project Lab BetterNet.

the viewport on which the video is played out and its impact on the user QoE. Cermak et al. [9] answered partially the question concerning the bandwidth needs for acceptable video experience on a set of screen resolutions. Still, downloading any resolution exceeding the screen resolution (or viewport) will be automatically downsized, hence resulting in a waste of bandwidth [14]. However, the literature is missing a solution to infer this screen resolution from passive captures of encrypted video traffic.

In this paper, we present a dedicated methodology able to perform viewport classification from YouTube encrypted video traces. To that aim, we leverage video chunk sizes and inband network-level traffic features such as throughput and download/upload packet inter-arrival time to train machine learning models able to distinguish *HD*/*SD* viewports. More specifically, our contributions are the following:

- We present a controlled experimental framework to perform video streaming experiments at large scale and to collect YouTube video metadata. We leverage the Chrome Web Request API to read the HTTP clear text messages [1] and obtain ground truth on the video streams.
- We stream up to 5K unique YouTube videos, collect encrypted traces and HTTP clear messages, and show that chunk sizes and inband network-level traffic features carry an interesting signature of the viewport resolution.
- We propose a novel approach to separate video and audio chunks from encrypted video traces based on a Gaussian Mixture Model (GMM). Then, we validate our work on inferring video chunk sizes by comparing similarities and differences with respect to the real video chunk size distribution derived from HTTP clear messages.
- We train different machine learning algorithms to classify the viewport resolution. We prove the pertinence of this classification taking as input chunk statistics and inband network-level traffic features that can be derived from the encrypted video traffic.

Overall, the paper is organized as follows. In Section III, we present the structure of our experimental framework. In Section IV, we present our methodology to extract video chunk sizes from YouTube encrypted traces. Then, in Sections V and VI, we highlight the viewport signature carried by a set of inband network traffic features and chunk size statistics, and train and evaluate a classifier able to predict the viewport class (*SD*/*HD*) from encrypted traces. Last, we conclude and present our future work in Section VII and VIII, respectively.

II. State of the art

Traffic identification from encrypted traces is a very active field of study. Methods based on Deep Packet Inspection (DPI) offer solutions to inspect and take actions based on the content of the packets (known as "payload") rather than just the packet header. Machine learning (ML) is widely exploited in the DPI field, as multiple ML based solutions have been proposed over last years [15]–[17]. ML techniques proved their efficiency, learning from big data and using statistical properties of the traffic flow. While constituting a promising approach, ML techniques require a lot of training and might struggle in terms of processing complexity if run in real-time. Another well-known DPI technology is OpenDPI, which is freely available and includes the latest DPI technology combined with other techniques making it one of the most accurate classifiers nowadays [18]. Khalife et al [19] attempt to reduce the OpenDPI computational overhead by examining different sampling techniques. Two sampling techniques are applied and compared *(i)* per-packet payload sampling, and *(ii)* per-flow packet sampling. Enhancing DPI performance is as active as inventing new DPI technologies, so several approaches have been proposed including behavioural [20], statistical [21], port based [22] and DFI (Deep Flow Identification) based approaches [23]. Other approaches apply software based optimization focused on enhancing DPI algorithms, e.g., [24], while other approaches rely on hardware based optimisation [25].

In terms of inferring video QoE, and given its tight correlation to application level QoS metrics [6]–[8], researchers leverage encrypted traffic by passively monitoring the network and capturing traffic statistics that are then transformed to video QoE indicators using machine learning. For instance, one can find work on inferring video interruptions, video quality and quality variations by observing network-level traffic statistics [10]. Others use a large number of video clips to identify specific Netflix videos leveraging only the information provided by the TCP/IP headers [26]. Dimopoulos et al [10] propose to use the size of video chunks as input to machine learning, however, their method requires access to end-user devices to collect real values about these chunks from the devices they control, instead of inferring them from the encrypted packet traces. They also provide the first heuristic to automatically extract chunk size information from encrypted traffic based on identifying long inactivity periods along with the video streaming session. Silhouette, a video classification method, uses Application Data Units (ADUs) and network statistics to detect video flows [27]. It leverages downlink/uplink packet characteristics to identify chunk requests and corresponding information sent by the server (chunk size), however, it only incorporates static thresholds making it unable to differentiate between video and audio chunks.

All the previous studies incorporate techniques targeting video flow identification and the inference of application level quality of service. However, in the light of advanced and diverse equipments, their limitation is in overlooking the end-user display which is an important factor determining the end-user QoE. In a previous contribution [28], we take into consideration the relationship that exists between screen resolution, video resolution and end-user QoE, and we propose a resource allocation problem that maximizes the overall QoE over a set of users sharing the same bottleneck link. In the present work, we propose to complete the puzzle by proposing techniques to infer the end-user viewport characteristics from the encrypted traffic, which together with the information on the video flow as the streaming resolution and the application

level QoS, can provide the ISP a fine-grained estimation of the user QoE for a more efficient video traffic management. To the best of our knowledge, this is the first attempt to infer the end-user viewport class (*SD* or *HD*) using inband network features and chunk sizes from encrypted video traces. Moreover, we believe our approach to separate video and audio chunks from encrypted traces is novel and prove its efficiency when compared to ground truth collected from HTTP clear messages through the Chrome web API [1].

III. Experimental setup

We play different YouTube videos using multiple viewports, under different network conditions emulated using Linux traffic control (*tc*). Each experiment consists of a unique YouTube video, browser viewport, and enforced network bandwidth. For every video session, we leverage the Chrome Web Request API to read the HTTP clear text messages for ground truth about the requested chunks and the application level quality of service. Moreover, we dump the encrypted client-server traffic to pcap files using *tcpdump*.

A. Overall experimental framework

Our overall experimental setup, described in Fig. 1, consists of a local *mainController* running on MacBook Air machine of 8 GB RAM. Videos are visualized on a Dell screen 27' of 2560 x 1440 resolution. The local *mainController* stores the YouTube video catalog and the viewport list, and provides a random combination of video ID and viewport for every new experiment as illustrated in Fig. 1. We consider a list of default standard viewports such as the current YouTube small media player mode (400x225) along with other default *SD* viewports (e.g., 240x144, 640x360 and 850x480). These latter viewports represent the current player dimensions adopted by streaming platforms for several watching modes. We also account for larger viewports by considering the standard 1280x720 and 1920x1080 (*HD*). As video resolution pattern is a function of both network conditions and terminal display capacity, we study the viewport importance while degrading the network bandwidth. To that aim, we use Linux traffic control *tc* and enforce different bandwidth settings such as 3, 6, 9, 15 and 20 Mbps. We also stream with no bandwidth limitation on Ethernet access to emulate the best case scenario.

B. YouTube catalogue

We use an open source YouTube catalogue [29] to identify the videos to stream. The catalog was built using the YouTube API where YouTube was searched with specific keywords obtained from Google Top Trends website. The authors of [30] rely on Google's *getvideoinfo* API to return the video metadata for each video identifier in the catalog. The dataset includes around 1 Million unique video identifiers.

IV. Analysis of video streaming traffic

In adaptive video streaming, the client decides on the resolution of the next chunk to download based on underlying network conditions and viewport characteristics. Each viewport is supposed to exhibit a different video resolution pattern,

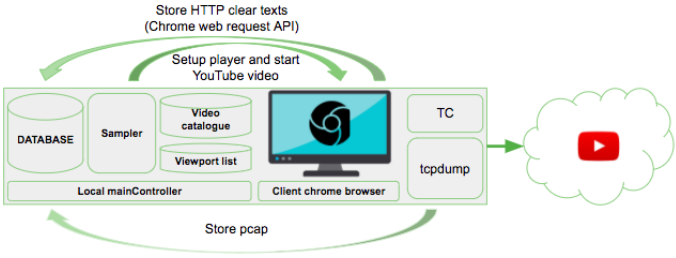


Fig. 1: Experimental framework description

which also depends on the available network resources; the pattern of chunk sizes is the main illustration of such specific behavior. However, as most of the video traffic is encrypted, such information on the viewport is not visible to any entity between the client and the server. Our intuition is to exploit the specificity of the chunk size pattern to infer the viewport resolution from encrypted traffic. The problem is that when the bandwidth starts going scarce either due to congestion or in-network shaping, clients will automatically be forced to request lower video resolutions, thus reducing the effect of the viewport and increasing the difficulty of viewport resolution inference. To highlight all these aspects, we investigate the extent to which screens are considered in video transmission while varying the network bandwidth.

A. Inferring chunk sizes

Overall, we stream up to 5K YouTube unique videos in series. In general, chunks of a video are fetched using separate HTTP requests. We propose a method to infer the chunk sizes from YouTube encrypted video traces. In parallel, we extract real chunk sizes from the clear text HTTP traces accessed from within the Chrome browser using the Chrome Web Request API [1].

As highlighted in [27], [30], for each video streaming session (source and destination already known), large sized uplink packets correspond to chunk requests while small packets correspond to transport level acknowledgments by TCP/QUIC. At first, we use the source IP of our host and the list of destination IPs to isolate the different flows corresponding to every video session (CDNs identified by the URLs ending with googlevideo.com). The CDN identifiers can be collected from the HTTP clear text messages and corresponding IP addresses can be resolved. Then, we look at the size of each uplink packet and instead of using thresholds as depicted in [16], we use K-means clustering to segregate the uplink packet sizes into two clusters; the first cluster represents the request packets and the second cluster represents the acknowledgment packets. Once the uplink request packets are identified, we sum the data downloaded between any two consecutive request packets, with this sum representing the downloaded chunk size corresponding to the first request between the two. Overall, we leverage clustering for the sake of generality and to make sure our approach can be reused with other type of ACKs mainly in the context of other transport protocols.

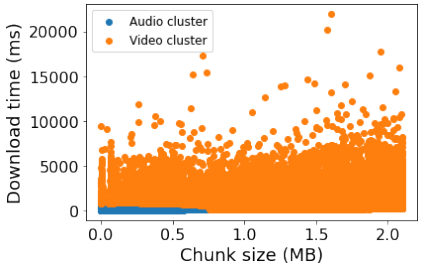


Fig. 2: Audio/video clusters as produced by GMM

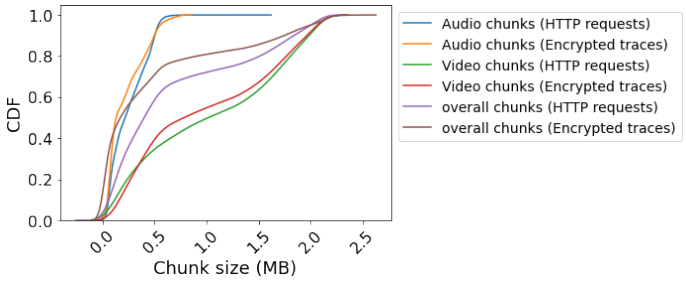


Fig. 3: Chunk size CDF

Up to this point, and as the case for other existing methodologies, the calculated chunk sizes mix between audio and video chunks, whereas we are only interested in the video part. In fact, the chunks located between consecutive request packets can be of either of the two types, audio or video, and so need to be separated. To overcome this limitation, we leverage Gaussian Mixture Models (GMM) applied to the chunk size. The GMM clustering algorithm is based on the maximum likelihood principle, able to find clusters of points in a dataset that share some common characteristics. Unlike K-means, the GMM belongs to the soft clustering subset of unsupervised algorithms, it provides probabilities that tell how much a data point is associated with a specific cluster. Another key property of GMM is that clusters do not need to be topologically separated as with K-means, they can overlap and still be identified as long as they follow some Gaussian property for the distribution of their points. In general, video chunks should have larger sizes than audio chunks, we use this property to identify the two components of the inferred chunks (two Gaussian distributions). Each distribution has three unique values, mean γ, covariance $\sum$ that defines its spread around the mean, and a probability π that defines how big or small one cluster is compared to the other one, the sum of probabilities of the two clusters is naturally equal to 1.

For our case, we fit a GMM of two components with the chunk sizes inferred according to the above methodology. To get a clear visual illustration, we plot in Fig. 2 the two clusters rendered by the GMM method, over a 2D space of chunk sizes (MB) and download time (s). In plain, the audio cluster is defined with a chunk size less than 750 KByte and no more than 2 seconds download time. The video chunks however can be of larger sizes and larger download times compared to the audio chunks.

Thanks to the above method, we are able to identify the chunks and infer their sizes from the encrypted traces (audio+video), and further to separate them between video and audio chunks. However, we still need to test the accuracy of this method by comparing its output to some ground truth. Even though we could have obtained the real label of each chunk of the encrypted traces by matching it to what has been seen in the browser, this information will not be available to an operator, so we decided to make sure our GMM method provides video chunk sizes that respect the distribution of the size of real video chunks as seen in the browser.

Overall, we leverage the HTTP request messages collected from within the browser. These requests include the *itag*, *range* and *mime* (Multi-purpose Internet Mail Extensions) parameters, which can be then used to infer the corresponding resolution, the codec and the size of the chunk using open source documentation [31], [32]. Fig. 3 compares the chunk sizes as estimated by our method from the encrypted traffic traces and the chunk sizes obtained from the clear text HTTP traces for the same video sessions. The overall distribution of the encrypted chunk sizes extracted using our method exhibits the same shape as the one obtained with HTTP requests. Further, the two distributions produced by our method for audio and video are very close to those of HTTP requests. We can also notice how the video chunks, understandably, have larger sizes than the audio chunks. This helps better characterizing the video chunks within a trace of encrypted video traffic, with this result particularly useful in our case to further understand the interplay between viewport, network resources and chunk resolution pattern.

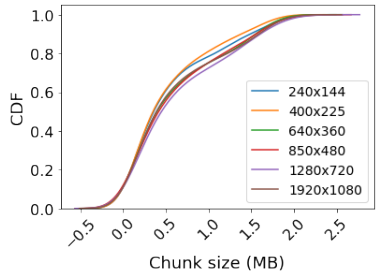
B. *Video resolution pattern*

For DASH, the client automatically switches between video resolutions according to the viewport and underlying network performance. The video resolution pattern as requested from the server is thus determined by the network conditions captured by the DASH client, and normally has to take into consideration the viewport size, which is defined as the number of pixels, both vertically and horizontally, on which the video is displayed. It is indisputable that the network conditions, for instance the bandwidth, will reduce the impact of the screen in scenarios of bandwidth shortage as DASH will end up downloading chunks of lower resolution than the viewport capacity. In this section, we present experimental results supporting these statements and highlight in particular the reduction of the effect of the viewport as the available bandwidth decreases.

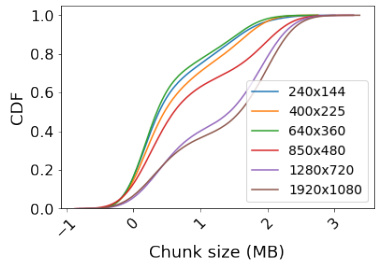
We artificially change the available bandwidth (as highlighted in Sec. III), and stream for each value hundreds of YouTube video sessions using different viewports. Each time, we use random sampling to select the video ID and the viewport. We plot in Fig.4 the CDF of the video chunk size per viewport for three bandwidth settings: 3Mbps, 15Mbps, and no control. As expected, the video resolution pattern is driven by the network bandwidth and the viewport size. In

Fig. 4(a), the bandwidth is limited to 3Mbps, all viewports thus exhibit the same pattern by selecting the same video resolution, which therefore results in the same cumulative distribution of chunk sizes. However, in Fig. 4(b) we set the bandwidth to 15Mbps, the effect of the viewport starts appearing as the distribution of chunk sizes defers from one screen to another. However, and even at this high bandwidth, the two large viewports 1280x720 and 1920x1080 illustrate close distributions, which can be explained by the same reason of bandwidth shortage. Finally, when no restriction imposed on the bandwidth, full high definition (1920x1080) viewport starts differentiating itself from the others. We further notice in Fig. 4(c) that 40% of chunk requests on small screens (e.g., 240x144, 400x225 and 640x360) correspond to a chunk size less 200 KBytes compared to 300 KBytes for medium screens. For HD screens, chunk sizes are bigger with 40% of chunks coming from the HD viewport to be less than 1 MBytes and less than 1.6 MBytes for FHD viewport.

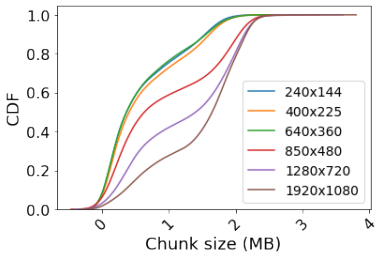
To illustrate further this result, we plot the achieved network throughput as measured over the encrypted traces and compare it to the available bandwidth for different viewports. For each video, we get the CDN URL from HTTP logs, use the DNS Lookup of CDN URL and identify the video flow corresponding to the CDN IP from the traffic traces. Then, we leverage the downlink packets' timestamps and a bin size of 1 second to return a vector of throughput values per video session. The vector is then used to derive throughput statistics (e.g., average, percentiles) per video session. In Fig. 5, we use the average throughput per video session to plot the CDF for different viewports with different bandwidth settings. We notice that with an enforced bandwidth of 3 Mbps (Fig. 5(a)), all viewports end-up experiencing the same throughput which correlates with chunk size results. Moreover, regardless of the available bandwidth, a subset of viewports form one cluster exhibiting the same throughput pattern (e.g., 240x144, 400x225 and 640x360).



(a) Chunk size per viewport with a 3Mbps bandwidth



(b) Chunk size per viewport with a 15Mbps bandwidth



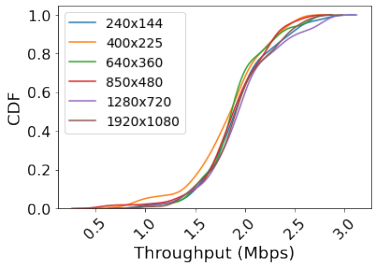
(c) Chunk sizes per viewport with unlimited bandwidth

Fig. 4: Network and viewport impact on chunk sizes

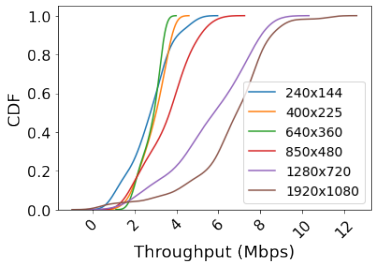
V. Traffic correlation to viewport

For each video session, we get an array of video chunk sizes over which we calculate different statistical features that we plan to use for viewport classification. We study here the correlation between this array and the viewport. Our feature set contains the maximum, the standard deviation, the $10th$ to the $90th$ percentiles (in steps of 10) of the chunk size array. This forms a set of 12 features describing statistically the evolution of chunk sizes over a video session. In addition to chunk size related statistics, we also consider the same statistical features, but this time over the downlink throughput (in bps, averaged over bins of 1s), and the uplink and downlink packet interarrival times (in seconds). With these features, we believe that we can get a fine-grained understanding of the DASH transmission process and capture any effect of viewport resolution. Overall, according to feature analysis, viewports such as 240x144, 640x360 and 850x480 are more likely to exhibit close chunk size and throughput distributions forming one viewport class (*SD*). On the other hand, the 1280x720 and 1920x1080 represent another cluster, called *HD* showing similar properties. To take advantage of this overlapping, we relax the viewport classification problem and consider a binary classification of the viewport to either *SD* or *HD*.

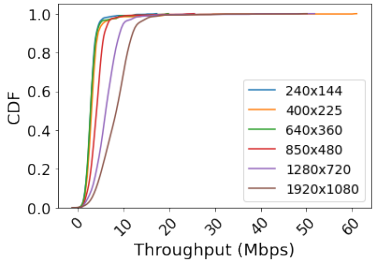
Before building our classifier, we start by studying here the correlation between our feature set and the target variable (the viewport class). In Fig. 6 we point to most relevant features by ranking them according to their Pearson correlation coefficient with the viewport class and showing only those having a correlation coefficient at least equal to 0.4. In this figure, the x_th_csize represents the x_th percentile of the video chunk size over a video session and y_th_dltp stands for the y_th percentile of the downlink throughput. Overall, the chunk size percentiles show a more important correlation with the viewport, especially when it comes to lower percentiles. Downlink throughput percentiles come in second place with a correlation coefficient of more than 0.4. Overall, the chunk sizes incorporate the strongest bond to the viewport, simply as the video resolution pattern is not only influenced by the available network resources, but also by the user display



(a) Throughput per viewport for a 3Mbps bandwidth



(b) Throughput per viewport for a 15Mbps bandwidth



(c) Throughput per viewport for a unlimited bandwidth

Fig. 5: Throughput per viewport for multiple network settings

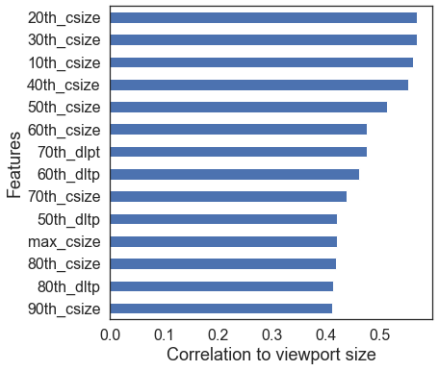


Fig. 6: Features correlation to viewport class

capacity.

To shed further light on the previous results, we show boxplots of the most important traffic features w.r.t. the two viewport classes. We plot in Fig. 7 the distribution of the $20th$ chunk size percentile for all video sessions and for both viewport classes. Overall, we notice a small overlapping portion,

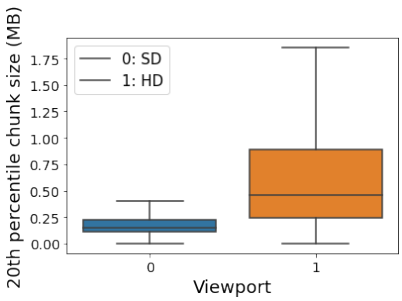


Fig. 7: $20th$ chunk size percentile (most relevant chunk size percentile in Fig. 6)

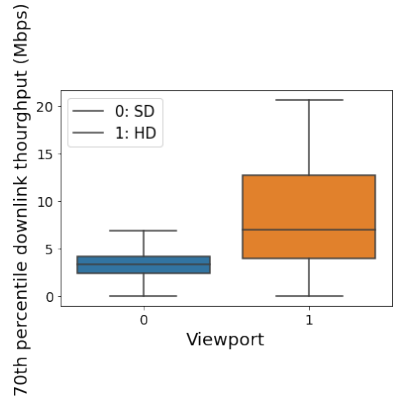


Fig. 8: $70th$ downlink throughput percentile (most relevant throughput percentile in Fig. 6)

the smaller the overlap the easier to differentiate between *SD* and *HD* viewports. In plain, 50% of video sessions have their $20th$ chunk size percentile less or equal than 230 KBytes, whereas high definition viewports score almost twice the value for the same percentile. In terms of downlink throughput, we plot in Fig. 8 the distribution of the $70th$ download throughput percentile for all video sessions as it scores 0.46 in terms of the correlation coefficient. In general, and as expected, larger screens are characterized by higher throughput values. Moreover, the boxplots show that half of our video sessions have a $70th$ download throughput percentile around 7 Mbps compared to 4 Mbps for small definition viewports. All these results point to the presence of a correlation pattern between encrypted traffic features and viewport resolution at the client, a pattern that we will exploit next to build our classifier of viewport resolution class.

VI. Viewport Classification by Machine Learning

In this section, we discuss the performance of the ML model built using our dataset. In plain, we try to predict the viewport category (*SD* or *HD*) using inband network features and chunk size stats ($F_{inband+chunk}$) extracted from YouTube encrypted video traces. Our goal is to provide the ISP with a mean to infer fine-grained video flow insights, in particular, approximate the viewport class, and this is despite the end-to-end encryption of the video flows. Such inference can help

the ISP to get an idea about the bandwidth requirements of her customers and their level of Quality of Experience (QoE) with the obtained network service, which can also help taking network management decisions (e.g., resource allocation, priority queuing) to improve such QoE [9] [28].

We build a dataset matching $F_{inband+chunk}$ to viewport class and use it to train different supervised ML classification algorithms. We randomly pick videos from the catalog available in [29], then stream them under different network conditions emulated locally using the Linux traffic control utility (*tc*). Each experiment consists of enforcing the bandwidth, playing out the selected video under the enforced QoS, collecting HTTP clear messages using the Chrome Web Request API and dumping the traffic in *pcap* files using tcpdump. The *pcap* files are used to calculate the feature set $F_{inband+chunk}$.

A. Classification accuracy vs bandwidth

As we have seen before, the effect of the viewport is maximum in a bandwidth unlimited scenario. As bandwidth decreases, the different viewports converge to the same video resolutions and therefore we expect any viewport inference model to get less accurate as both classes (*SD*/*HD*) start overlapping. To assess the extent of such limitation, we test our model in different scenarios each featuring a different bandwidth configuration. We use random forest (using python Scikit-Learn library [33]) as in our case it showed the best results compared to other classifiers such as support vector, decision tree and multi-layer perceptron (MLP). To find the best tuning, we apply at first a random search of best hyper-parameters values, then after reducing the space of search, we apply a grid search to get a fine-grained fitting of major parameters, this methodology is widely adopted in the machine learning community [34].

Fig. 9 highlights the accuracy of two classification models trained with our dataset. In plain, for each bandwidth setting on the x-axis, we highlight two models related to the random forest approach trained on two datasets, the blue model is trained with video samples conducted with one specific enforced bandwidth (the corresponding x-axis value), the orange one is a model trained with the aggregate set of video sessions obtained over all enforced bandwidth values. It follows that the blue model varies from one x-axis value to another one, where the orange model is the same over all x-axis values. On average, we validate on a test set of 200 videos specific to each bandwidth scenario as depicted on the x-axis in the figure. In general, regardless of the training set, the model accuracy is coherent with our hypothesis as the accuracy increases w.r.t. the enforced bandwidth. For example, in case of enforced bandwidth of 3Mbps, both models show a low performance with a median accuracy of 62%. This is expected as for such low bandwidth, viewports show similar distributions for most important features (see Fig. 3). One can expect an even lower accuracy if the exact viewport resolution is to be predicted for such a low bandwidth value. Starting from 6 Mbps, both models start showing a median accuracy exceeding 80%, with

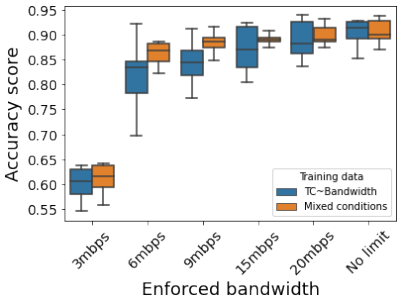


Fig. 9: Model accuracy vs enforced bandwidth

	Precision	Recall	F1
HD	0.87	0.85	0.86
SD	0.89	0.93	0.91

TABLE I: Random forest sample (precision/recall)

the model based on mixed conditions showing better accuracy (in terms of both average and variance) than the model specific to the enforced bandwidth value, which is a good property of the orange model given its generality over different bandwidth scenarios. We recall that the best performance for both models is reached when no limitation is imposed on the network bandwidth, the median accuracy, in this case, is around 92%.

The previous results present a general evaluation of the model, yet, we need to evaluate the model per viewport class. Here, one can use metrics like *precision* and *recall* or simply the F1 score which is an average of both. To that aim, we benchmark well known supervised machine learning algorithms, fit them on our dataset (the no bandwidth limitation case) and compare them per class using the F1 score. We plot in Fig. 10 the k-fold (k =10) validation result for well known machine learning algorithms. According to this validation, the dataset is split into k folds, and at each of the k iterations, a new fold is used as a validation set while the k - 1 remaining folds form the training set. Average results are then calculated over the k iterations. In plain, random forest seems to be the most relevant algorithm, as it shows an average F1 scores of 85%, while other algorithms such as linear discriminant analysis and decision tree classifier come second and third with average F1 score of 80% and 78% respectively. The lowest performance is recorded for the multi-layer perceptron and linear regression classifiers with 72% and 68% F1 scores respectively. Moreover, we show in Table I the precision and recall values of a tree sample produced by our random forest model, our model classifies correctly 85% of the total video sessions issued from *HD* viewports and 93% of the sessions related to *SD* viewports. When our model labels a video session as *HD*, it is correct in 87% of the cases and in 89% of the cases for *SD*.

B. Real time viewport classification

Up to this point, the statistics we use to train and test our model take into consideration the entire video session.

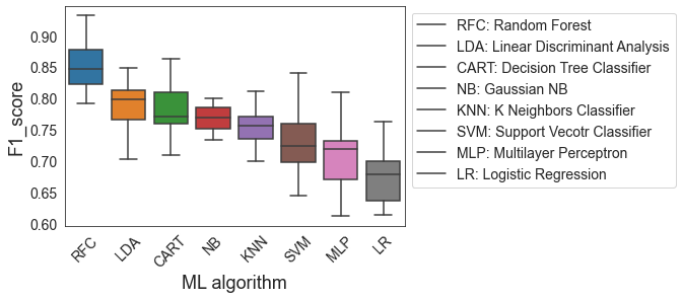


Fig. 10: ML algorithm comparison (no bandwidth limitation)

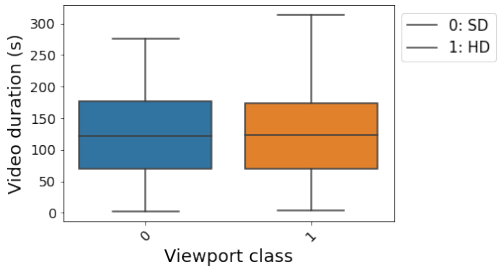


Fig. 11: Video duration distribution (seconds)

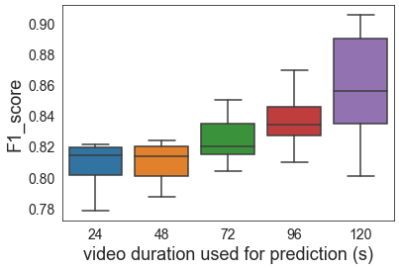


Fig. 12: Model accuracy vs video proportion considered

This requires waiting until the end of the session to collect the features and predict the viewport, which can limit the usability of the method in practice by preventing from taking quick and real time traffic engineering actions. For that, one needs to perform the classification as soon as the video starts playing out, thus allowing for mechanisms such as weighted fair queuing and load balancing to take place. So here, we study the goodness of our model for viewport prediction on the fly, which instead of using as input aggregated statistics on the entire video session, calculates features on the early part of the session.

We stream a total of 104 hours (4 days) and 70 hours (almost 3 days) of random YouTube videos using different *SD* and *HD* viewports respectively. We highlight in Fig. 11 the video duration distribution per viewport class. As expected, the two distributions look the same, with half of the videos requested from *SD* and *HD* viewports having a median duration of 120 seconds. We split this dataset into training and test sets. In the training set, we compute the features by considering the entire video session. On the other hand, for the test set, we use a specific proportion of the video starting from its beginning and test over it. For instance, an input on the first 20% will consider a feature set $F_{inband+chunk}$ calculated over the first 20% of the video, and so on. To represent the proportions in seconds and give them a practical meaning, we consider the median video duration (120s) as reference duration, so proportions of 20%, 40%, and 60% would correspond to the first 24, 48, and 72 seconds of a video session.

We plot in Fig. 12 the F1 score w.r.t. the proportion used as input for the test. With no surprise, the larger the proportion of video session considered, the higher the accuracy of the model is. This makes sense as the model gets more and more relevant input as compared to those used for the training. Most importantly, the model still works with few seconds as input and provide good classification accuracy exceeding 80% on average. This confirms that the first few seconds of a video session do also carry an important signature of viewport class, with for example the first 24 seconds (assuming a median duration of 120 seconds) allowing a median F1 score of 80% and 78% in the worst case. We recall that considering the full video data leads to 85% median F1 score. We shall thus confirm the feasibility of our approach to perform pseudo real-time viewport classification, yet, the best performance is ensured if 100% of video session related information is used.

VII. Conclusion

In this paper, we present our methodology for building viewport prediction models from YouTube encrypted video traces using controlled experimentation and machine learning. The models presented predicts *HD* and *SD* viewports from statistical features calculated over the encrypted video packets, fully or partially. Such information on the viewport can help the ISP plan better traffic engineering actions for a more efficient network management and QoE optimization. Our methodology starts by inferring chunk sizes, then relies on Gaussian Mixture Models to separate video chunks from audio chunks. Statistics on video chunks are then used to train machine learning models for viewport classification. These models show prediction accuracy that improves with the available network bandwidth, and can go up to 92% in its median. The median F1 score can go up to 85%. Limiting the classification to the first few seconds of the video decreases its accuracy, but still leads to acceptable levels of F1 score. Our experimental study can be reused for other video content provider as long as documentation exists to interpret the HTTP clear text messages (to get ground truth on chunk sizes).

VIII. Future work

As future work, we plan to extend our study to cover other popular streaming platforms (e.g., Netflix) and viewport sizes (e.g., UHD). Moreover, we aim at reusing our results to perform optimal resource allocation at the edge of the network leveraging terminal characteristics and aiming at maximizing the Quality of Experience of end-users.

REFERENCES

[1] Google, "Chrome Web Request Extension," https://developer.chrome.com/extensions/webRequest, 2020.

[2] Ericsson, "Ericsson Mobility Report, June 2018," https://www.ericsson.com/assets/local/mobility-report/documents/2018/ericsson-mobility-report-june-2018.pdf, 2018.

[3] T. Stockhammer, "Dynamic adaptive streaming over http –: Standards and design principles," in *ACM conference on Multimedia systems (MMSys)*, 2011.

[4] C. Müller and C. Timmerer, "A vlc media player plugin enabling dynamic adaptive streaming over http," in *ACM international conference on Multimedia*, 2011.

[5] sandvine, "The Global Internet Phenomena Report, October 2018," https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf, 2018.

[6] R. R. Pastrana-Vidal, J. C. Gicquel, C. Colomes, and H. Cherifi, "Sporadic frame dropping impact on quality perception," in *Proc. SPIE 5292, Human Vision and Electronic Imaging IX*, 2004.

[7] Y. Qi and M. Dai, "The effect of frame freezing and frame skipping on video quality," in *Proc. IEEE Int. Conf. Intell. Inform. Hiding Multimedia Signal Process.*, 2006.

[8] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. D. Veciana, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies,," in *IEEE J. Sel. Topics Signal Process.*, 2012.

[9] G. Cermak, M. Pinson, and S. Wolf, "The relationship among video quality, screen resolution, and bit rate," in *IEEE Transactions on Broadcasting*, 2011.

[10] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *ACM Internet Measurement Conference*, 2016.

[11] T. Hoßfeld, M. Seufert, M. Hirth, T. Zinner, P. Tran-Gia, and R. Schatz, "Quantification of youtube qoe via crowdsourcing," in *IEEE International Symposium on Multimedia*, 2011.

[12] M. Khokhar, T. Ehlinger, and C. Barakat., "From network traffic measurements to qoe for internet video," in *IFIP Networking Conference*, YouTube catalogue link, 2019.

[13] O. Belmoukadam, T. Spetebroot, and C. Barakat, "Acqua: A user friendly platform for lightweight network monitoring and qoe forecasting," in *3rd International Workshop on Quality of Experience Management*, 2019.

[14] O. Belmoukadam, M. Khokhar, and C. Barakat., "On excess bandwidth usage of video streaming: when video resolution mismatches browser viewport," in *11th IEEE International Conference on Networks of the Future*, 2020.

[15] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive ip traffic," in *IEEE/ACM Transactions on Networking*, 2012.

[16] R. Thupae, B. Isong, N. Gasela, and A. Abu-Mahfouz, "Machine learning techniques for traffic identification and classification in sdwsn: A survey," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, 2018.

[17] J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, "Internet traffic classification by aggregating correlated naive bayes predictions," in *IEEE Transactions on Information Forensics and Security*, 2013.

[18] Opendpi, "," http://www.opendpi.org/, 2012.

[19] J. Khalife, A. Hajjar, and J. E. Díaz-Verdejo, "Performance of opendpi in identifying sampled network traffic," in *Journal of Networks*, 2013.

[20] LiJuan Zhang, DongMing Li, Jing Shi, and JunNan Wang, "P2p-based weighted behavioral characteristics of deep packet inspection algorithm," in *International Conference on Computer, Mechatronics, Control and Electronic Engineering*, 2010.

[21] F. Dehghani, N. Movahhedinia, M. R. Khayyambashi, and S. Kianian, "Real-time traffic classification based on statistical and payload content features," in *2nd International Workshop on Intelligent Systems and Applications*, 2010.

[22] G. Aceto, A. Dainotti, W. de Donato, and A. Pescape, "Portload: Taking the best of two worlds in traffic classification," in *INFOCOM IEEE Conference on Computer Communications Workshops*, 2010.

[23] C. Wang, X. Zhou, F. You, and H. Chen, "Design of p2p traffic identification based on dpi and dfi," in *International Symposium on Computer Network and Multimedia Technology*, 2009.

[24] G. La Mantia, D. Rossi, A. Finamore, M. Mellia, and M. Meo, "Stochastic packet inspection for tcp traffic," in *IEEE International Conference on Communications*, 2010.

[25] A. Rao and P. Udupa, "A hardware accelerated system for deep packet inspection," in *ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE)*, 2010.

[26] A. Reed and M. Kranch, "Identifying https-protected netflix videos in real-time," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, 2017.

[27] F. Li, J. Chung, and M. Claypool, "Silhouette: Identifying youtube video flows from encrypted traffic," in *28th ACM SIGMM Workshop*, 2018.

[28] O. Belmoukadam, M. Khokhar, and C. Barakat, "On accounting for screen resolution in adaptive video streaming: a qoe driven bandwidth sharing framework," in *CNSM*, 2019.

[29] YouTube, "Video Catalogue," https://drive.google.com/open?id=1tu0sBInt8xJ9Zn32IDlh6DW_ju2FhEou, 2020.

[30] M. Khokhar, T. Ehlinger, and C. Barakat., "From network traffic measurements to qoe for internet video," in *IFIP Networking Conference*, YouTube catalogue link, 2019.

[31] YouTube, "Itag documentation," https://www.genyt.xyz/formats-resolution-youtube-videos.html, 2020.

[32] Git, "Itag catalogue," https://gist.github.com/sidneys/7095afe4da4ae58694d128b1034e01e2, 2020.

[33] Python, "Random Forest classifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2020.

[34] M. blog, "Hyperparameter Tuning the Random Forest in Python," https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74, 2020.