YouTube QoE Estimation Based on the Analysis of Encrypted Network Traffic Using Machine Learning

Irena Orsolic, Dario Pevec, Mirko Suznjevic and Lea Skorin-Kapov
University of Zagreb, Faculty of Electrical Engineering and Computing
10000 Zagreb, Croatia
Email: {irena.orsolic, dario.pevec, mirko.suznjevic, lea.skorin-kapov}@fer.hr

***Abstract*—The widespread use of encryption in the delivery of Over-The-Top video streaming services poses challenges for network operators looking to monitor service performance and detect potential customer perceived Quality of Experience (QoE) degradations. While monitoring solutions deployed on client devices provide insight into application layer KPIs (e.g., video quality levels, buffer underruns, stalling duration) which can be further mapped to user QoE, network providers commonly rely primarily on passive traffic monitoring solutions deployed within their network to obtain insight into user perceived degradations and their potential causes. In this paper we present a methodology for the estimation of end users' QoE when watching YouTube videos which is based only on statistical properties of encrypted network traffic. We have developed a system called YouQ which includes tools for monitoring and analysis of application-layer KPIs and corresponding traffic traces, and the subsequent use of this data for the development of machine learning models for QoE estimation based on traffic features. To test this approach, we have collected a dataset of 1060 different YouTube video traces using 39 different bandwidth scenarios. All video traces are annotated with application-layer KPIs and classified into one of three QoE classes. The dataset was used to test various machine learning algorithms, and results showed that up to 84% QoE classification accuracy could be achieved using only features extracted from encrypted traffic.**

I. Introduction

Studies haves shown that 67% of global Internet traffic in 2014 was video, with that share expected to grow to 80% by 2019 [1]. One of the most prominent video streaming services is YouTube, delivered via Google's Content Delivery Network (CDN). While Over The Top (OTT) providers like YouTube offer services directly to end users, Internet Service Providers (ISPs) are responsible for transmitting IP traffic and generally lack control over the services. From an ISP's point of view, there is a lack of insight into the performance or quality parameters of YouTube streams passing through their network, as YouTube's traffic is encrypted. The inability to monitor service performance at an application level poses a threat to providers, as they are potentially unable to detect problems and act accordingly. Poor performance further imposes the risk of losing customers, as customers often tend to blame ISPs for poor QoE.

One approach for ISPs to tackle these challenges is to estimate QoE based on network traffic characteristics at the transport layer. The research problem can be generalized as a network traffic classification problem in which the traffic of one service is being classified into different behavioural classes. There has been a lot of research on the subject of network traffic classification using machine learning (ML) techniques. In [2], the authors explain the main problems of Internet traffic identification and potential solutions. They also present the trends in this field and provide an overview of techniques for traffic analysis. Similarly, in [3], the authors discuss the motivation for the application of ML to Internet traffic classification and give a comprehensive survey of relevant studies. An approach called BLINC was developed to classify network flows based on application category (web, p2p, streaming, gaming, etc.) [4]. In [5], the authors describe a methodology for classifying network traffic into four application classes: interactive, bulk-data transfer, streaming, and transactional. Another example is [6] which describes Bayesian classification of network traffic into the following categories: bulk, database, interactive, mail, services, www, p2p, attack, games, and multimedia. Our approach will be based on applying ML techniques to classify YouTube video sessions into QoE classes.

Because YouTube is compliant with the MPEG-DASH standard, video quality is dynamically adapted based on client-detected network conditions and buffer state. Hence, the resolution of delivered videos is not necessarily constant, making it no longer possible to accurately estimate QoE from methods which rely on measured average download throughput and assumed video encoding bit rate, such as discussed in [7]. The approach in [8] proposes the Prometheus system, which relies on ML techniques to relate passive in-network measurements to application's QoE. It measures QoE of video-on-demand and VoIP applications without requiring knowledge about specific application services. Promising results with over 80% accuracy are reported when network traffic features were used to classify application QoE into two classes ("good quality" and "bad quality") using data collected in the core network of a large cellular operator. For algorithm training purposes, network traffic is annotated with application layer QoE metrics measured on mobile devices.

The aim of this paper is to propose a methodology and instrumentation for estimating YouTube QoE based solely on the analysis of encrypted network traffic. We have developed a system called **YouQ**, which includes a solution for developing a training dataset based on monitored client-side application-layer KPIs (including quality level of video playback, number and duration of stalling events, etc.) and network video traces

annotated with QoE classes. As opposed to [8], we have used a classification based on three QoE classes, described further in Section II. Relevant traffic features are extracted and machine learning techniques are used to estimate QoE per video session. The system has been tested by collecting a large dataset involving 1060 different videos streamed over a laboratory WiFi network under various bandwidth conditions.

The paper is organized as follows. The laboratory setup and test methodology are described in Section II. Section III reports on the characteristics of the collected dataset and the QoE classification accuracy for different ML algorithms. Conclusions and future work are discussed in Section IV.

II. Methodology

This section describes components of the developed YouQ system and the employed test methodology. Furthermore, a description is given of the QoE classes used to classify YouTube video streaming instances based on calculated application-layer KPIs.

A. Laboratory testbed

The laboratory testbed is depicted in Figure 1. YouTube traffic between an Android smartphone (Samsung S6 with Android version 5.1.1) and YouTube servers is transmitted over an IEEE 802.11n wireless network and then routed through a PC running IMUNES (www.imunes.net), a general purpose IP network emulation/simulation tool enabling a test administrator to set up different bandwidth limitations and schedule bandwidth changes [9]. Traffic is further sent through Albedo's Net.Shark device (a portable network tap used for aggregating and mirroring network traffic) where it is replicated and sent to a PC designated for network traffic capturing. The PC running IMUNES also has an OS layer, accessed by the YouQ client application to run a bandwidth scheduling script according to defined experiments.

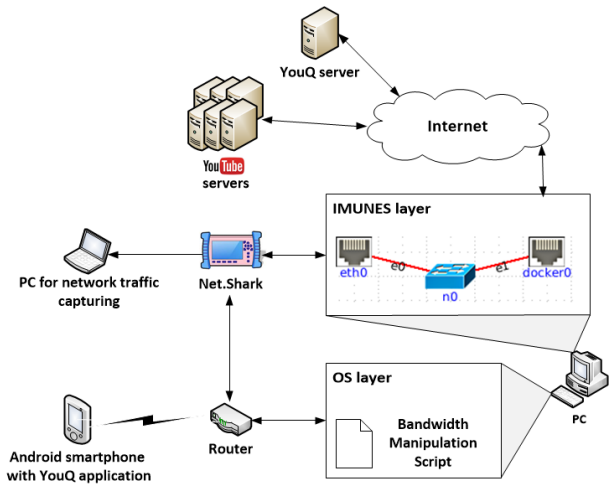


Fig. 1. Laboratory testbed.

B. YouQ system - developed tools

The system consists of a YouQ Android application and a YouQ server. The YouQ server is comprised of: 1) a YouTubeDatabase application for creating a large pool of video IDs and storing their metadata (this data is collected via the YouTube Data API); 2) an FTP server for storing measurement results; 3) a YouQ web application for displaying experiment results; 4) a database server for storing application level KPIs and video metadata; 5) scripts for data processing, and 6) an application for extracting network features. The most important components are described in further detail below.

YouQ Android application is run on the client and monitors application-level data that is used to calculate various QoE-related KPIs (number of stalling events, average stalling duration, percentage of playback time spent on certain quality level, etc.). The application enables a test administrator to select a target number of videos to be played during an experiment (e.g., run experiment with 100 different videos). Filtering options enable the administrator to request that the duration of each played video falls within a set interval (e.g., play only videos that last between 60 and 120 s), that the videos have been viewed a certain minimum number of times, and that the videos are offered in a certain playback quality (high definition or standard definition). Video IDs to be played during the experiment are retrieved from the aforementioned database, which we populated with over 2 million YouTube video ID entries. Throughout the entire experiment during which the chosen number of videos are played (note that all played videos are different), the application collects data using the YouTube IFrame API, into three log files, as described in Table I. Existing solutions for client-side monitoring of YouTube QoE include YoMoApp [10], a similar Android application which uses the YouTube mobile website and HTML5 API to passively monitor KPIs. Our goal was to enable automatic data collection from a large number of filtered YouTube videos without user interaction.

TABLE I
DATA COLLECTED WITH YOUQ ANDROID APPLICATION

Log	Description
Event log	YouTube events: "Buffering", "Playing", "Paused", "Ended", "hd1080" (quality level playback has switched to), "hd720", "large" (480p), "medium" (320p), "small" (240p), "tiny" (144p)
Buffer log	Amount of video buffered in every second of watch time.
URL log	URLs from all HTTP requests towards YouTube servers.

Data processing scripts are a part of the YouQ server. They are used to process captured network traffic and logs from the YouQ Android application. The output of these scripts are files with selected fields from captured network traffic for each video from the experiment and files with application KPIs for every video in each experiment. Scripts are also used to populate a database with information from the application level for the purpose of data visualization.

Application for extracting network traffic features is developed for extraction of relevant traffic features from captured network traffic traces. Following an analysis of related work on feature extraction [11]–[13] and also our numerous observations of YouTube adaptation behavior, we have comprised a set of 54 features, which can be divided into six categories: packet length statistics, size of transferred data in 5s intervals, packet count statistics, interarrival time statistics, throughput statistics, and TCP flags count. Features were extracted for all captured network traffic that belongs to a single video, for each contacted YouTube server, and established TCP flow, by inspecting headers of each packet. The set of features was further reduced to 33 primarily based on redundancy. In addition to calculation and extraction of traffic features, the application calculates the QoE class that each video belongs to, based on application level KPIs and the approach described in Section II-C. This application prepares a training dataset for use by ML algorithms.

C. QoE classification

Previous studies have shown that QoE in HTTP-based adaptive video streaming services depends on: percentage of playback time spent on each quality level, number of stalling events, average stalling duration, overhead time, and initial buffering [14]–[16]. In this work, initial buffering has been taken into account only through overhead time, since none of the videos had a significant initial buffering duration. Although models have been derived based on subjective studies which estimate QoE from the aforementioned KPIs, there is currently a lack of models which simultaneously map the impact of multiple influence factors onto QoE. For the purpose of relating multiple KPIs with QoE, a simplified model has been created using as a basis existing models and findings with respect to the impact of various QoE influence factors [14]–[18]. Considering the difference in model dimensionality, the classes in this work cannot be expressed as a range of MOS values provided by related works. We note that this classification is proposed only for proof-of-concept purposes, and that the aim of this paper is not to propose new QoE models for adaptive video streaming.

We define the following three QoE classes to be used for classification purposes: "high", "medium", and "low". An instance of a video streaming session is evaluated by two functions which quantify **measured degradations**, and based on this classify the session to the "high" or "low" QoE class. If it belongs to neither, it is classified as the "medium" QoE class. The first function, which checks if the QoE class of an instance i is "high", is defined as:

$$checkHigh(i) = \begin{cases} nPHQ(i) + nSD(i), & SC = 1 \\ 1, & otherwise \end{cases}$$

where nPHQ is the normalized value of the percentage of playback time spent on high quality (*hd1080*, *hd720*, *large*), nSD is the normalized value of stalling duration, and SC is stalling count. Normalization functions are defined as follows (PHQ is percentage of playback spent on high quality and SD is stalling duration):

$$nPHQ(i) = \begin{cases} 0, & PHQ(i) > 90\% \\ 9 - 0.1 * PHQ(i), & PHQ(i) \in [80\%, 90\%] \\ 1, & PHQ < 80\% \end{cases}$$

$$nSD(i) = \begin{cases} 0, & SD(i) < 3s \\ 0.5 * SD(i) - 1.5, & SD(i) \in [3s, 5s] \\ 1, & SD(i) > 5s \end{cases}$$

If the output value of the function $checkHigh(i)$ is less then 1, degradation is considered not significant, and instance i is classified as "high" QoE. Otherwise, it is checked by the function $checkLow(i)$ and if it does not belong there either, the instance is classified as "medium".

The function used to check whether or not a video session should be categorized as "low" is as follows:

$checkLow(i) = nPLQ(i) + nSC(i) + nOR(i) + nASD(i)$

PLQ refers to percentage of playback time spent on low quality (*small* and lower), OR refers to overhead ratio (overhead time and video duration ratio), SC is stalling count, and ASD is average stalling duration. These influence factors are normalized as follows:

$$nPLQ(i) = \begin{cases} 0, & PLQ(i) < 50\% \\ 0.1 * PLQ(i) - 5, & PLQ(i) \in [50\%, 60\%] \\ 1, & PLQ(i) > 60\% \end{cases}$$

$$nSC(i) = \begin{cases} 0, & SC(i) < 4 \\ 0.5 * SC(i) - 2, & SC(i) \in [4, 6] \\ 1, & SC(i) > 6 \end{cases}$$

$$nOR(i) = \begin{cases} 0, & OR(i) < 0.08 \\ 0.25 * OR(i) - 2, & OR(i) \in [0.08, 0.12] \\ 1, & OR(i) > 0.12 \end{cases}$$

$$nASD(i) = \begin{cases} 0, & ASD(i) < 3s \\ 0.5 * ASD(i) - 1.5, & ASD(i) \in [3s, 5s] \\ 1, & ASD(i) > 5s \end{cases}$$

If the function $checkLow(i)$ returns a value greater than or equal to 1, degradation is considered significant and the session is classified as "low" QoE. If any of the four normalization functions returns 1, session i is labelled with "low" QoE. If the sum of output values of these four functions is less than 1, the session is classified as "medium".

D. Measurement procedure

Figure 2 shows the steps a test administrator has to take when conducting an experiment. It also shows the resulting output of every phase. Following setup of a desired bandwidth envelope, network traffic capturing is started and the experiment is initiated via the YouQ Android application. An administrator can choose the number of videos to be played during the experiment, and optionally can specify the previously mentioned filters to be applied when retrieving YouTube video IDs. After all the videos are played, both application and network level data can be uploaded to the YouQ server for processing.

To collect data, 39 experiments with different bandwidth limitations were conducted (each experiment involved multiple videos being played). The bandwidth envelopes were

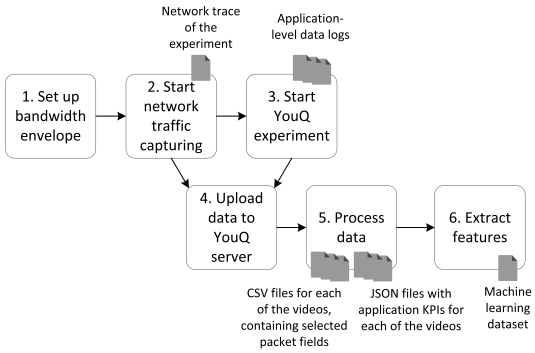


Fig. 2. Experiment actions.

derived based on extensive prior analysis of YouTube buffering behaviour and adaptation logic (further details are out of scope for this paper). The goal was to cover a large number of different adaptation and playback scenarios differing in played quality levels, stalling number and durations, etc. Each experiment is defined by its bandwidth envelope and video duration range requested in the YouQ Android application. The number of videos per experiment and their durations were chosen so as to balance the overall number of videos per QoE class. Table II gives an overview of experiments with their bandwidth envelopes, number of videos played in the experiment, and selected video duration. If bandwidth limitation was not constant during the experiment, the moment of change is given on the arrow. The number on the arrow corresponds to the number of seconds passed from the initial buffering event of the currently playing video until a new bandwidth limitation was set using IMUNES. For example, the experiment with ID 16 was run as follows: at the time when video playback was initiated, available bandwidth was limited to 3 Mbps. After 60 seconds, this limitation was dropped to 1 Mbps, and after another 60 seconds it was again dropped to 0.5 Mbps.

III. Results

The collected dataset contains information corresponding to the streaming of 1060 videos obtained across 39 experiments. The dataset, besides application KPIs and network traffic features, contains video metadata, such as video duration, view, like, and dislike count. This data was collected for future analysis of how content popularity affects QoE, considering delivery efficiency is improved by replicating popular content.

The percentage of instances labelled with each of the defined QoE classes per experiment is presented with Figure 3. A total of 333 instances were labelled as "low", 310 as "medium", and 417 as "high".

Figure 4 shows overall stalling statistics: number of videos with certain stalling count, average stalling duration, and number of stalling events per experiment. It can be observed that in most of the videos no stalling event occurred. In most of the videos with stalling occurrence, average stalling duration was short. It is also interesting to note that most of the stalling

TABLE II
Experiment scenarios

No. of experiment	Bandwidth envelope $[Mbps]$	Number of videos	Video duration $[s]$
1	5	9	600-650
2	3	9	600-650
3	1	9	600-650
4	5	16	180-600
5	3	9	180-600
6	1	18	180-600
7	$1 \xrightarrow{60} 5$	14	180-600
8	$1 \xrightarrow{120} 5$	18	180-600
9	$1 \xrightarrow{240} 5$	5	480-600
10	$0.5 \xrightarrow{60} 3$	17	180-600
11	$0.5 \xrightarrow{120} 3$	16	180-600
12	$0.5 \xrightarrow{240} 3$	4	480-600
13	$3 \xrightarrow{60} 0.5$	14	180-600
14	0.5	16	150-180
15	$1 \xrightarrow{60} 0.5$	18	180-600
16	$3 \xrightarrow{60} 1 \xrightarrow{120} 0.5$	16	180-360
17	$0.5 \xrightarrow{60} 1 \xrightarrow{120} 3$	13	180-360
18	$0.5 \xrightarrow{60} 10 \xrightarrow{120} 0.5$	20	360-600
19	$0.5 \xrightarrow{90} 2$	26	180-200
20	$0.5 \xrightarrow{60} 1 \xrightarrow{120} 0.5 \xrightarrow{180} 1 \xrightarrow{240} 0.5$	5	300-360
21	0.3	27	180-200
22	$10 \xrightarrow{120} 1$	26	300-600
23	$0.3 \xrightarrow{30} 1 \xrightarrow{90} 0.5 \xrightarrow{150} 8 \xrightarrow{165} 1 \xrightarrow{225} 0.4$	18	240-600
24	7	42	240-600
25	0.7	18	120-420
26	1.5	17	120-420
27	0.8	19	120-420
28	2	17	120-130
29	4	44	120-420
30	3.5	47	120-420
31	2.7	48	60-180
32	7	25	120-300
33	1.3	43	120-300
34	0.6	43	120-300
35	0.5	45	120-300
36	0.4	56	60-120
37	0.55	89	60-120
38	$0.85 \xrightarrow{45} 0.75$	89	60-120
39	0.35	75	60-120

events happened in experiments with low overall bandwidth, with drastic bandwidth changes, and in experiments with an increase in available bandwidth. The last observation can be attributed to the fact that YouTube, when switching to higher quality level, presumably discards all of the video content buffered in lower quality level and starts downloading and playing higher quality video immediately.

In Figure 5a it can be observed that overhead time was shorter than 10s in more than 90% of the cases. Figure 5b shows the number of videos that at some point played in a certain quality level, while Figure 5c shows the distribution of percentage of video duration played in a certain quality level. It can be observed that a large number of videos were not

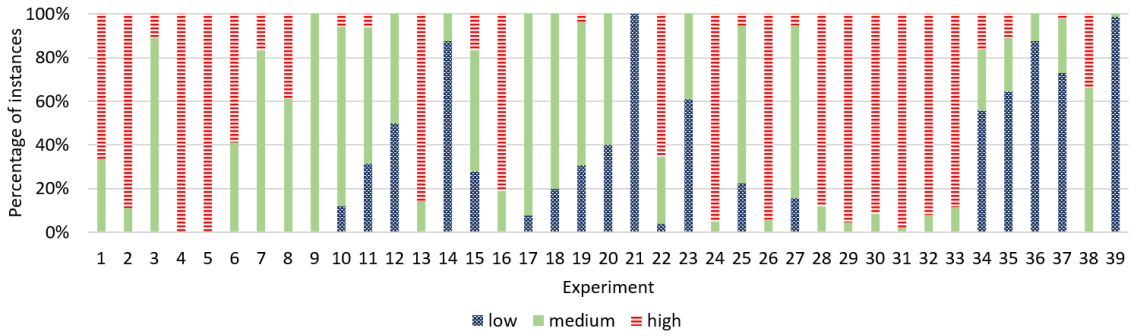
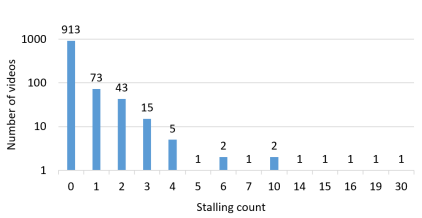
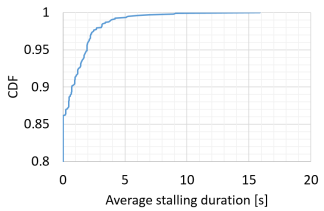


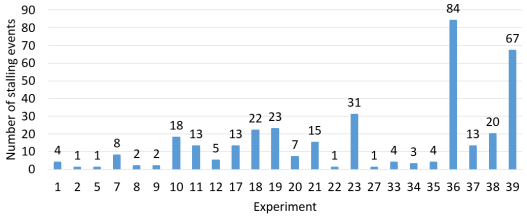
Fig. 3. Percentage of instances in each class, per experiment.



(a) Number of videos with certain stalling count.

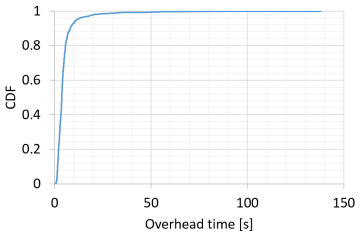


(b) Average stalling duration.

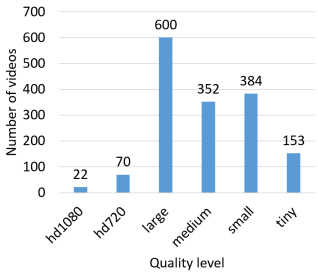


(c) Number of stalling events per experiment. Experiments with 0 stalling events are not included.

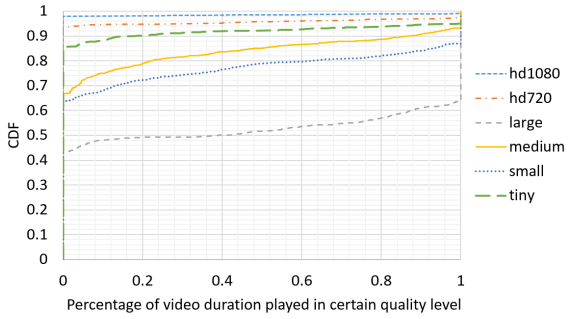
Fig. 4. Stalling statistics.



(a) Overhead time.



(b) Number of videos that played in certain quality level at some point.



(c) Percentage of video duration played in certain quality level.

Fig. 5. Overhead time and quality level statistics.

played in *hd1080* or *hd720* because these quality levels were either not available, or, even if they were available, were not retrieved by the YouTube client algorithm. This may be due to the fact that the videos were played on a mobile device and YouTube assumes that quality level *large* is good enough to achieve satisfactory QoE on a small screen. All the videos from all the experiments were available at least in quality level *large*, if not higher.

The collected dataset, with each video instance described with 33 network traffic features and labelled with a QoE class, was prepared as input for the WEKA Java library, which includes implementations of all tested ML algorithms [19]. To reduce computational cost and eliminate irrelevant features, features were subset by using Wrapper methods in WEKA. For each of the five tested algorithms, features were selected and models built by using selected features only. 10-fold cross-validation was used for testing all the models. Table III provides an overview of used algorithms, features selected by Wrapper methods for each algorithm, and classification accuracy of the model. Models were built using the full dataset and a reduced dataset, which contains only the data collected in experiments with constant bandwidth limitations (Exp. 1-6, 14, 21, 24-37 and 39). The reduced dataset focuses on more realistic network conditions, with no drastic bandwidth fluctuations. On the full dataset, the model built by Random Forest algorithm had the highest accuracy (80.18%), and on the reduced dataset the most accurate model was built by using Naïve Bayes algorithm (83.94%). Further analysis determined that most of the classification errors corresponded to videos that were either static or had high resolution and few stalling

TABLE III
CLASSIFICATION RESULTS

Algorithm	Selected features	Accuracy	
		Full dataset	Reduced dataset
OneR	*throughputMedian*	74.62%	83.8%
Naïve Bayes	*avgPacketSize*, *averageInterarrivalTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *push*, *interarrivalTimeThroughTimeMedian*, *initialThroughput2*	77.35%	83.94%
SMO	*maximalSizeThroughTime*, *minimalInterarrivalTime*, *sizeThroughTimeMedian*, *maxThroughputThroughTime*, *dupack*, *effectiveThrouhput*	77.35%	80.97%
J48	*minimalInterarrivalTime*, *avgInterarrivalTimeThroughTime*, *sizeThroughTimeStdDev*, *interarrivalTimeThroughTimeMedian*, *throughputMedian*	78.20%	83.26%
Random Forest	*avgPacketSize*, *minimalSizeThroughTime*, *push*, *initialThroughput10*, *minThroughputThroughTime*, *interarrivalTimeThroughTimeMedian*, *dupack*, *effectiveThrouhput*	80.18%	83.53%

events. Binary QoE classification ("low" and "high") was also tried and the accuracy of the models was raised up to 89%.

IV. CONCLUSION

The instrumented tools and applied test methodology prove the feasibility of QoE classification of adaptive YouTube video streaming based solely on encrypted network traffic. We note that the proposed QoE classes are not based on an official recommendation or standard, but are rather derived based on previous studies that primarily model QoE in terms of a limited number of influence factors. Pending the availability of future multidimensional models, this classification can be redefined accordingly. We also note that the primary factor contributing to QoE was quality level, while stalling events were rarely observed. Furthermore, while we used a large number of different bandwidth conditions, a dataset comprising fewer fluctuations and fewer sharp drops or increases would likely lead to better classification accuracy. Future work will focus on further analysis of why prediction errors happen, with the aim being to incorporate that knowledge into classification models. Moreover, ongoing work involves application of the test methodology and tools to network traces collected in an operational mobile network, followed by an analysis of classification accuracy using various ML techniques.

ACKNOWLEDGMENT

This work has been conducted in the scope of the project "Survey and analysis of monitoring solutions for YouTube network traffic and application layer KPIs" funded by Ericsson Nikola Tesla, Croatia. This work has also been supported in part by the Croatian Science Foundation under the project UIP-2014-09-5605 (Q-MANIC).

REFERENCES

[1] "The Zettabyte Era: Trends and Analysis," Cisco, Tech. Rep., 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.pdf

[2] A. Callado, C. Kamienski, G. Szabó, B. P. Gerö, J. Kelner, S. Fernandes, and D. Sadok, "A survey on Internet traffic identification," *Comm. Surveys & Tutorials, IEEE*, vol. 11, no. 3, pp. 37–52, 2009.

[3] T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2008.

[4] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: multilevel traffic classification in the dark," in *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4. ACM, 2005, pp. 229–240.

[5] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification," in *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. ACM, 2004, pp. 135–148.

[6] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, no. 1. ACM, 2005, pp. 50–60.

[7] P. Casas, A. D'Alconzo, P. Fiadino, A. Bar, A. Finamore, and T. Zseby, "When YouTube Does not WorkAnalysis of QoE-Relevant Degradation in Google CDN Traffic," *Network and Service Management, IEEE Transactions on*, vol. 11, no. 4, pp. 441–457, 2014.

[8] V. Aggarwal, E. Halepovic, J. Pang, S. Venkataraman, and H. Yan, "Prometheus: toward quality-of-experience estimation for mobile apps from passive network measurements," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014, p. 18.

[9] M. Zec and M. Mikuc, "Operating system support for integrated network emulation in imunes," in *Workshop on Operating System and Architectural Support for the on demand IT Infrastructure (1; 2004)*, 2004.

[10] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "YoMoApp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *Proc. of EuCNC 2015*. IEEE, 2015, pp. 239–243.

[11] Q. A. Chen, H. Luo, S. Rosen, Z. M. Mao, K. Iyer, J. Hui, K. Sontineni, and K. Lau, "QoE doctor: Diagnosing Mobile App QoE with Automated UI Control and Cross-layer Analysis," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 151–164.

[12] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," 2005.

[13] R. Schatz, T. Hoßfeld, and P. Casas, "Passive youtube QoE monitoring for ISPs," in *2012 6th IEEE Intl Conf. IMIS*, 2012, pp. 358–364.

[14] P. Casas, M. Seufert, and R. Schatz, "YOUQMON: A system for online monitoring of YouTube QoE in operational 3G networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 2, pp. 44–46, 2013.

[15] D. Ghadiyaram, A. C. Bovik, H. Yeganeh, R. Kordasiewicz, and M. Gallant, "Study of the effects of stalling events on the Quality of Experience of mobile streaming videos," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*. IEEE, 2014, pp. 989–993.

[16] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on Quality of Experience of HTTP adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[17] T. Hoßfeld, R. Schatz, E. Biersack, and L. Plissonneau, "Internet video delivery in YouTube: From traffic measurements to Quality of Experience," in *Data Traffic Monitoring and Analysis*. Springer, 2013, pp. 264–301.

[18] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: between the devil and the deep blue sea," in *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. IEEE, 2012, pp. 1–6.

[19] "Weka 3: Data Mining Software in Java." [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/