EFFECTOR: DASH QoE and QoS Evaluation Framework For EnCrypTed videO tRaffic

Raza Ul Mustafa, Md Tariqul Islam, Christian Rothenberg
University of Campinas (UNICAMP)
{razaul,tariqsaj,chesteve}@dca.fee.unicamp.br

Pedro Henrique Gomes
Ericsson Research
pedro.henrique.gomes@ericsson.com

***Abstract*—The exponential increase in Dynamic Adaptive Streaming over HTTP (DASH) based video traffic with end-to-end encryption poses many challenges for Mobile Network Operators (MNOs). To improve user-perceived video quality, MNOs must be aware of the end user's Quality of Experience (QoE) by exploring network level Quality of Service (QoS). On the contrary, the network edge facility provides proximity for performing an intelligent operation to maintain smooth QoE from the network level QoS measurement. Therefore, in this work, we propose EFFECTOR, a framework to showcase lightweight in-band QoS features measurement technique at edge nodes from encrypted DASH video traffic. EFFECTOR uses an emulated environment with real 4G and 5G drive test traces to generate video traffic. Moreover, this work provides a massive dataset for analyzing the impact of 4G and 5G technology on video quality in the form of Interactive Jupyter Notebooks. The proposed framework is ideal for investigating QoS extracted from the network's edge and finding its relations with QoE to ensure better video quality for end-users.**

***Index Terms*—Quality of Experience (QoE), DASH, Quality of Service (QoS), Network Edge.**

I. Introduction

Recently 5G has been rolling out to deliver higher data speeds, 1-millisecond end-to-end over-the-air latency, and support more users and devices when compared to 4G [1]. 5G and Beyond (5GB) networks are expected to be equipped with the Edge Computing (e.g., Multi-Access Edge Computing) paradigm, which brings the computing of traffic and services closer to the end-users, moving from a centralized cloud to the network edge of the network. The main intuition behind edge computing is to reduce latency, provide real-time response and network conditions, lower network bandwidth load, and provide better performance to end-users in real time.

On the other end, MPEG-DASH (Dynamic Adaptive Streaming over HTTP) video traffic is supposed to increase 73% by 2023 [2]. Therefore, it has become a critical issue for Mobile Network Operator (MNO) stakeholders to manage such traffic demand and provide a satisfactory experience to their end-users, known as Quality of Experience (QoE). Thus, understanding and monitoring the Key Performance Indicators (KPIs) that impact users' perceived experience and service quality has become a challenging and trending topic for MNOs to ensure better QoE.

In DASH, the choice of the Adaptive Bitrate Streaming (ABS) algorithm plays a significant role in end-user satisfaction [3]. Based on the current network conditions, ABS algorithms select each segment with a more suitable quality. The objective of any ABS is to provide maximum QoE to video users. Each algorithm exhibits its own functionality in different video streaming scenarios. Thus, there is a need to be aware of network conditions, i.e., traffic patterns, to optimize video delivery quality.

In the past, several approaches have been proposed to measure the KPIs aimed at delivering acceptable video service quality. Most solutions require end-user awareness, which is not viable from the MNOs' perspective. However, Deep Packet Inspection (DPI), in spite of being a widely used solution to estimate the KPIs directly from network traffic, is not a convenient solution anymore for the operators due to the adoption of end-to-end video streaming-encryption over TCP (HTTPs) and QUIC transport protocols. Therefore, we can only rely on the information available in the IP packets header, such as Packet Time and Packet Size, to extract the Quality of Service (QoS) KPIs and their pattern to infer QoE. In this case, edge computing can be an ideal location for evaluating the perceived video service quality (QoE) of target end-users, as it is closer to them and thus reduces the potential for errors caused by cross traffic interference. This proximity allows for more accurate measurement of QoS KPIs [4].

In this work, we present a framework called EFFECTOR, which offers a tailored evaluation approach for measuring the QoE and QoS in specific network regions, such as edge facilities. It takes into account factors such as network topology, capacity (from 4G and 5G network traces), end-user characteristics (e.g. individual or group) and the type of service being evaluated (e.g. DASH). The QoS features are mainly derived from two basic KPIs: packet arrival time and packet size. To the end, we calculated a total of 30+ downlinks QoS features based on different statistics of each KPI in different use-cases of 4G and 5G network conditions using three state-of-art ABS i) Conventional – Throughput, BBA – Buffered, Elastic – Hybrid [5]. Furthermore, EFFECTOR is equipped to evaluate both TCP (HTTPS) and QUIC protocols in combination with various types of DASH videos, including segments ranging from 2-10 seconds, with different encoding schemes. We provide a complete explanation of the different types of settings in the later parts of the paper.

The main contributions of this paper are as follows:

- EFFECTOR provides 30+ QoS features per 1-second time window and demonstrates correlation with QoE using real 4G and 5G use cases. The framework allows flexibility in time windows, video contents, drive test traces, and host numbers. Further details on time window and QoS features can be found in Section III-A and III-B, respectively.
- We also provide a total of 14 QoE features generated from each experiment's video playout performance log, which include different objective QoE metrics and QoE model outputs. We explain each metric in Section III-C.
- Moreover, we derived three unique QoS features from Inter Packet Gap (IPG) named Exponentially Weighted Moving Average (EMA), Double Exponentially Weighted Moving Average (DEMA) and Cumulative Sum (CUSUM).
- Finally, this work provides a massive generated dataset of QoS and QoE KPIs from the framework's video traffic using 4G and 5G drive test traces. Interactive Jupyter Notebooks are provided for analyzing preprocessed datasets obtained over TCP (HTTPS), and QUIC transport, including QoS features from encrypted network traffic (1-second granularity) and per-segment QoE metrics from video log files.

The rest of the paper is structured as follows: Section II presents background and related work. In Section III we provide QoS and QoE features extraction approach. Next, in Section IV we provide experimental setup for the production of the dataset followed by 4G vs. 5G comparison in Section V. Next, in Section VI we provide QoS features correlation with the QoE. Finally, Section VII concludes our paper and discusses some future work.

II. Background & Related Work

In DASH, the video is encoded in different bitrates and resolutions. The client-side ABS algorithm is responsible for selecting the video segment based on the current network condition to avoid freeze (stall) and provide maximum perceived quality to video users. Typical segment duration is usually (2-10)-seconds and the information of each segment bitrate is stored in a Media Representation File (MPD). Once the player downloads the MPD file, the ABS algorithm can adjust quality by selecting the most appropriate segment for each video over time.

The ABS algorithms are divided into three major categories i) rate-based [6], buffer-based [3] and hybrid-based [7]. In rate-based, a decision is made on the delivery rate of the previously downloaded segments. Buffer-based algorithms monitor the state of the playback buffer, while in hybrid, both the playback buffer and delivery rate are considered for the choice of the next segment. However, due to data transfer privacy concerns, content providers recently delivered their DASH service with encryption. Currently, contents are served over HTTPS (TCP) and QUIC transport protocols.

Several in-network probing schemes have been carried out in the literature for end-to-end encrypted video traffic to extract entire sessions and time window-based real-time QoS and QoE KPIs, as well as their mapping. Authors in [8] showed the entire video session generated QoS KPIs for making QoS-to-QoE correlation. Contrarily, Authors in [9], [10] showed QoS to QoE mapping by adopting a 10-second granularity KPIs extraction. In turn, authors in [11], [12] showed window-based QoS feature extraction for inferring QoE with 1-second granularity. Most of the mentioned works were followed through a particular streaming service (e.g., YouTube) adaptation logic. In contrast, this work is based on an emulation-based DASH video service, which allows the use of different adaptation logic (i.e., ABS algorithm) for video segment selection, similar to previous works [13], [14], [5]. In previous work [13], we showed temporal network level QoS KPIs extraction and QoE estimation in the scope of the edge domain. Likewise, in work [14], [5], we explored Machine Learning (ML) for video service assurance through QoE estimation on a per-segment basis QoS feature set from an unencrypted testbed traffic. Meanwhile, this work is an extension of [14], by supporting end-to-end encrypted video traffic and providing a different lightweight, fine-grained QoS feature extraction [15]. Mapping QoS to QoE is essential in the context of encrypted traffic, as it allows for the delivery of higher-quality video content through a better understanding of how limited network resources affect QoE. To fill this gap, we provide a flexible framework for passive network traffic monitoring of DASH-supported video service at the network edge, generating a large dataset of QoS and QoE KPIs for various combinations of network capacity, transport protocol, ABS algorithm, and video content.

III. QoS & QoE Features Extraction Approach

A. Time Window

The passive probing scheme of EFFECTOR continuously extracts features from the encrypted traces in a stream-like manner, as shown in Figure 1. It is worthwhile noting that the extracted features are based on packet-level statistics, and their computation does not require a chunk-detection mechanism. It leverages current techniques, since video content providers change their delivery methods over time, making chunk-level statistics prediction a non-trivial task [16]. Thus, finding chunk-level features in encrypted network traffic is a research challenge that is surpassed in out proposal.

Herein, we consider a window-based QoS features extraction method where we split the entire QoS metrics collection approach into small windows. As shown in Figure 1, the acquisition of the whole DASH video streaming session QoS is achievable by concatenating each window QoS.

B. QoS Features Engineering

In Table I, we provide QoS feature statistics extracted from encrypted network traffic based on the pattern of packet arrival time and their corresponding volume. IPG stands for Inter Packet Gap, and IAT represents 1-second window Inter

Arrival Time, which denotes the time difference between the last and first packet in the window. We assess the average and standard deviation of the IPG value of a window for all packets and the packets whose lengths are greater than 100B. The intuition behind considering packet length greater than 100B is that the TCP stream is full of tiny volume packets (e.g., SYN, ACK, RST); however, we also use the same threshold for QUIC. We followed the same approach to count the total number of packets. In addition, we evaluated average throughput (bits per second) and the distribution of the 10th to 90th percentile (in steps of 10) for throughput and packet size in each time window.

In Algorithm 1, the first five lines initialize variables such as the video streaming session – `Session (n-minutes)`, current time – `Ctime`, QoS features extraction time – `Step`, Total number of packets – `Tpackets`, and Total packets with size greater than 100B – `Tpackets_GT100`. Following, in `While` loop, we store `Packet_size` and `Packet_time` in two separate `arrays` for later use. Next, on line 9, we convert `Packet_size` into bits and assign all the bits to variable `BITS`. From lines 10-13, we check `Packet_size`, If it is greater than 100B, we store size and time in arrays named `Array_GT100_Size`, `Array_GT100_Time` respectively. On line 14, we increment the `Step` of a time window.

Then, on lines 16-18, we extract (10 to 90)-th percentile of packet size from the `Array_Size`, which is an array of packet sizes. We calculate (10 to 90)-th percentile throughput distribution in time intervals on lines 19-21. On line 22, we divide all numbers of bits by window size. Subsequently, we find total number of packets on line 23 by taking `Count` of array `Array_Size` followed by taking `Count` of packet sizes greater than 100B on line 24. We take IAT of a window on line 25, which is the difference between the last packet and the first packet of a window. Next, on line 26, we find the average of IPG followed by the IPG average of packets whose length is greater than 100B on line 27. Lastly, on lines 28-29, we store the standard deviation of the packet sizes, followed by the standard deviation of IPG on lines 30-31. At this point, we calculate a total of 28 QoS features.

In addition, we calculate the Exponentially Weighted Moving Average (EMA), Double Exponentially Weighted Moving Average (DEMA), and Cumulative Sum (CUSUM) from IPG. Usually, EMA and DEMA are technical indicators used to identify a potential uptrend or downtrend in time series data. We use these metrics to find where the continuity of data packets changes from the mean value. As mentioned earlier, we store each packet's time in the array named `Array_Time`. In the following step, we derive IPG from the packet's time. We use all IPGs to a function that recursively computes EMA value and returns to function. Algorithm 2 calculates EMA where we initialize $\alpha = 0.99$, since α value is between 0 and 1. The parameter α decides how important the current observation is in calculating EMA. Now, we present the DEMA calculation in Algorithm 3. The function takes an input of IPG as an array and returns a value of DEMA for a time window. Finally, Algorithm 4 denotes CUSUM calculation which is a Cumulative Sum of IPGs for a time interval. In the end, with these additional features, we have a total of 31 QoS features.

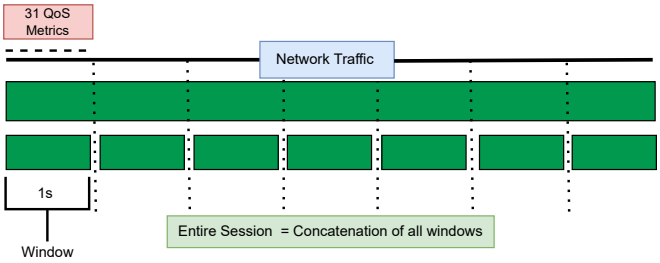


Fig. 1: Demonstration of 1-second Time Windows

TABLE I: Summary Statistics of QoS Features Derived from Packet Time (PT) and Packet Size (PS)

QoS Features	PT	PS
IAT	✓	
IPG, Average and StD (w/ GT 100B) of IPG	✓	
Total Packets (w/ GT 100B)		✓
Average Throughput	✓	✓
10th to 90th Percentile of Throughput	✓	✓
StD (w/ GT 100B) of Packet size		✓
10th to 90th Percentile of Packet size		✓
EMA, DEMA, CUSUM of IPG	✓	

TABLE II: QoE Features Extracted from Player Logs

QoE Features
Arrival, Delivery, Stall, Representation rate, Delivery rate Actual bitrate, Segment size, Buffer, Resolutions P.1203, Yin, Yu, Clae, Duanmu

C. QoE Features

Per-segment QoE features provided in the framework are shown in Table II. Specifically, `Arrival` feature denotes the arrival time of a segment in millisecond (msec). `Delivery` denotes time spent to deliver the current segment in msec. `Stall` means pause of streaming in msec followed by `Representation rate` of the downloaded segment in Kbit/sec. Then, `Delivery rate` means the delivery rate of the network in Kbit/sec (segment size divided by the time for delivery). `Actual bitrate` signifies segment size divided by the segment duration in Kbit/sec, and `Segment size` represents segment volume in bytes. Next, `Buffer` features indicate buffer status in second after the current segment was just downloaded, followed by segment resolution by means of `Resolutions` features. Finally, five QoE models named `P.1203`, `Yin`, `Yu`, `Clae`, and `Duanmu` [16], [17] for each segments.

Algorithm 1 QoS Feature Engineering

$Session \leftarrow n\_minutes$ ▷ Streaming session
$Ctime \leftarrow 0$ ▷ Current time – Begin from 0
$Step \leftarrow window$ ▷ Step (1,2,3,4,5)-seconds
$Tpackets \leftarrow 0$ ▷ Total packets
$Tpackets\_GT100 \leftarrow 0$ ▷ Total packets size > 100
while $Ctime \leq Session$ **do**
  $Array\_Size\ [] \leftarrow Packet\_Size$
  $Array\_Time\ [] \leftarrow Packet\_Time$
  $BITS \leftarrow BITS + (len(p[IP]) \times 8)$
  if $Packet\_Size > 100$ **then**
    $Array\_GT100\_Size\ [] \leftarrow Packet\_Size$
    $Array\_GT100\_Time\ [] \leftarrow Packet\_Time$
  end if
  $Ctime \leftarrow Ctime + Step$
end while
while P = 10 to 90 **do**
  P percentile of $Array\_Size$
end while
while P = 10 to 90 **do**
  P percentile of throughput in a time window
end while
$Throughput \leftarrow BITS/window$
$Tpackets \leftarrow Count\ (Array\_Size)$
$Tpackets\_GT100 \leftarrow Count\ (Array\_GT100\_Size)$
$IAT\_w \leftarrow Last - First$ ▷ Last PT - First PT in a window
$IPG\_avg \leftarrow Average\ (IPG\ (Array\_Time))$
$IPG\_avg\_gt100 \leftarrow Average\ (IPG\ (Array\_GT100\_Time))$
$Std \leftarrow STD\ (Array\_Size)$
$Std\_gt100 \leftarrow STD\ (Array\_GT100\_Size)$
$Std\_IPG \leftarrow STD\ (IPG\ (Array\_Time))$
$Std\_IPG\_gt100 \leftarrow STD\ (IPG\ (Array\_GT100\_Time))$

Algorithm 2 EMA calculation, function takes IPG values of a window as an array

$FUNCTION \leftarrow START$
$\alpha \leftarrow 0.99$
$EMA\_array\ () \leftarrow IPG[0]$
$I \leftarrow 0$
while $I < count\ (IPG)$ **do**
  $EMA\_array\ () \leftarrow IPG[I] * (1-\alpha) + IPG[I-1] * \alpha$
  $I++$
end while
$return\ array\_sum(EMA\_array)$
$FUNCTION \leftarrow END$

Algorithm 3 DEMA calculation, function takes IPG values of a window as an array

$FUNCTION \leftarrow START$
$\alpha \leftarrow 0.99$
$DEMA\_array\ () \leftarrow IPG[0]$
$I \leftarrow 0$
while $I < count(IPG)$ **do**
  $DEMA\_array() \leftarrow (\alpha * IPG[I]) + ((1-\alpha) + DEMA\_array[I-1])$
  $I++$
end while
$return\ array\_sum(DEMA\_array)$
$FUNCTION \leftarrow END$

Algorithm 4 CUSUM calculation, function takes IPG values of a window as an array

$FUNCTION \leftarrow START$
$I \leftarrow 0$
$CUSUM \leftarrow 0$
while $I < count(IPG)$ **do**
  $CUSUM += I$
  $I++$
end while
$return\ CUSUM$
$FUNCTION \leftarrow END$

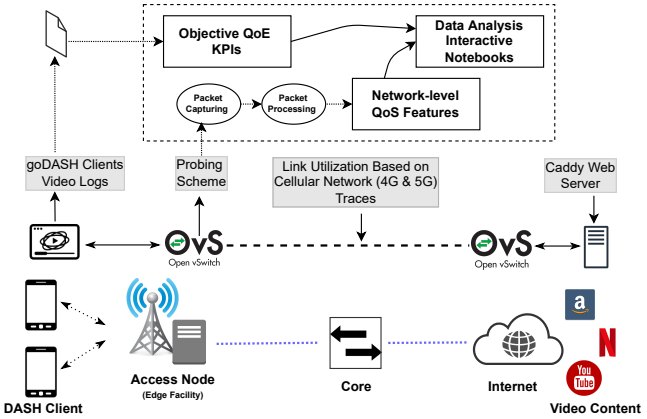


Fig. 2: DASH QoS and QoE Evaluation Block Diagram

IV. Experimental Use Case

Figure 2 shows a synopsis of the framework, which provides a lightweight headless DASH-compatible video streaming tool called `godash` [16], [17] at the client node(s) and `Caddy`, a `WSGI` web server to host a popular 2-seconds segment duration x264 videos named Sintel [1] and Tears of Steel [2] sourced from a publicly available 4K DASH video dataset [18]. We use 4G and 5G drive test traces [19] to generate a rich dataset with heterogeneous player configuration. Drive test traces were generated with 1-second granularity (window) in different scenarios such as i) Bus – Mobility, ii) Pedestrian – Low Mobility, iii) Static Download – Downloading a large file continuously for a fixed period. However, we consider static download scenarios to emulate them in the framework, which are synonyms for the network's maximum capacity, i.e., (4G and 5G). Inside the framework, link capacity as per 4G and 5G traces during each experiment are changed in near real-time (e.g., 1-second time granularity) using Linux Traffic Control (TC) and Hierarchical Token Bucket (HTB) [20]. The framework is also flexible to change the time of link capacity to visualize the behavior of two or more consecutive segments downloaded. The aforementioned 1-second time granularity is ideal for the detection of anomalies, troubleshooting approaches, as well as proactive traffic management [12].

A. Virtual Machine (VM) & Interactive Notebooks

For ease of use, we provide a Virtual Machine (VM) equipped with all these dependencies to run DASH video streaming experiments [21]. We also offer Interactive Notebooks to play with the produced dataset. There are two types of notebooks available, i) To see the impact of different 4G and 5G scenarios on different ABS algorithms performance, ii) Per-segment QoE logs and per-second QoS KPIs of a video (see Figure 7). The `Jupyter` notebook and CSV dataset are uploaded to `GitHub`.[3]

[1] https://bit.ly/3BPh3i0
[2] https://bit.ly/3A54g9W
[3] https://github.com/razaulmustafa852/EFFECTOR

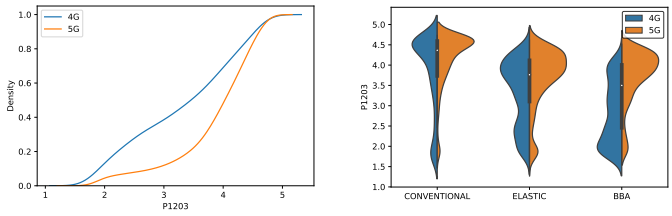
B. Framework Configuration

The Table III shows different types of use-cases for running a DASH streaming session. The framework primarily operates on six different combinations, as outlined in the table. First, configure the technology, i.e., 4G and 5G, to run the DASH video session that invokes a `Mininet`[4] based network topology consisting of two Open vSwitch (OvS) where first OvS is considered as edge point to capture raw network traffic using `Tcpdump`. The next parameter is the number of DASH clients followed by the ABS algorithm, i.e., (bba, conventional, elastic) that invokes the configuration of the `godash` and selects the ABS algorithm. Then trace selection parameter sets the link capacity (e.g., throughput) values from the CSV file employing by bash script to change the link capacity every 1-second between two OvS virtual interfaces. In the subsequent, choosing protocol implies specifying the available transport protocol, i.e., TCP and QUIC, for video streaming. Lastly, selecting the number of repetitions means running the same experiment multiple times, resulting in different generated folders with all the experiment's outputs.

To initiate experiment execution, the prerequisite is to open a terminal in the programs' directory and run the program which contains all the configuration (e.g., run.py). The DASH video streaming sessions will start and continue until the different combinations are set above. Each experiment will generate a folder in the current directory with the name "1_4g_Case_1_1_bba_tcp", which is an experiment – 1, Technology – 4G, Trace-Case – 1, DASH Clients – 1, ABS algorithm – bba and protocol – TCP, including video log and captured network traffic. Moreover, we provide two scripts to generate QoS logs by extracting QoS features from captured raw TCP and QUIC-based network traffic inside the VM.

TABLE III: EFFECTOR Configurations

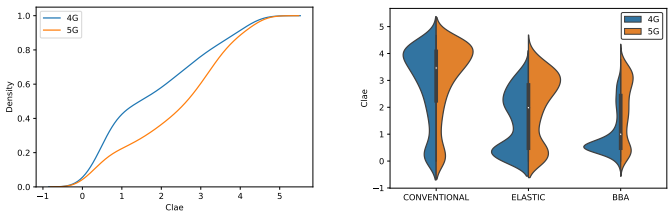
Parameter	Example
Technology	[4G, 5G]
Host	[1]
Algorithm	[BBA, Elastic, Conv]
Trace	[Use Cases]
Protocol	[TCP, QUIC]
Experiment Repetitions	[1,2,3,...,n]



(a) QoE P.1203 (b) QoE P.1203 by ABS

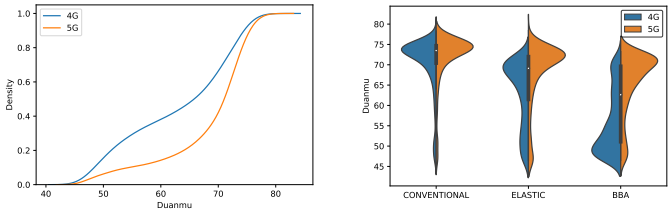
Fig. 3: QoE Model P.1203 in 4G vs. 5G

[4]http://mininet.org/



(a) QoE Clae (b) QoE Clae by ABS

Fig. 4: QoE Model Clae in 4G vs. 5G



(a) QoE Duanmu (b) QoE Duanmu by ABS

Fig. 5: QoE Model Duanmu in 4G vs. 5G

V. 4G vs. 5G

We start by comparing the P.1203 score collected over the 4G and 5G technologies. To offer a fair comparison of both the technologies, we select Video – Sintel and Protocol – TCP. We show three QoE models considering the space limitation i) P.1203, ii) Clae, and iii) Duanmu. We show the Cumulative Distribution Function (CDF) of the QoE models mentioned earlier for both technologies. We can see in Figure 3 (a), 5G outclass 4G in QoE. 80 % of the QoE P.1203 score for 4G remains 4.0; however, in the case of 5G, we observe a higher QoE of 4.5. We notice a similar 5G dominance in all cases of ABS (see Figure 3 (b)). Similarly, in Clae, we experience that 5G has a better overall QoE score than 4G (see Figure 4 (a)) and QoE model score by ABS (see Figure 4 (b)). Finally, We show the QoE model Duanmu in Figure 5 for both the technologies 4G and 5G in Figure 5 (a) and by ABS in Figure 5 (b).

A. Quality Shifts & Stalls

In Table IV, we show the percentage of a single quality – resolution for the 4G and 5G. We consider video – Sintel, Protocol – TCP using all ABS for a fair comparison. However, to see the ABS impact on the resolutions, we also provide Interactive Jupyter Notebook to visualize quality shifts during each video streaming session. In 5G, 87 % of the segments remain in 1920x1080 at higher resolutions, with very few segments in lower resolutions, whereas in 4G, 56 % of segments remain in 1920x1080, and there are frequent quality shifts during the session. 4G suffers from massive stalling events as compared to 5G (see Figure 6).

B. Interactive Notebooks

1) QoE Interactive Jupyter Notebook: In order to visualize and investigate the impact of QoE and QoS in 4G and

TABLE IV: Percentage of Segments Resolutions in 4G & 5G

Technology	320x180	384x216	512x288	640x360	736x414	1280x720	1920x1080
4G	5.93	4.12	8.14	7.75	5.93	11.54	56.55
5G	4.21	1.05	2.10	1.16	1.15	2.66	87.63

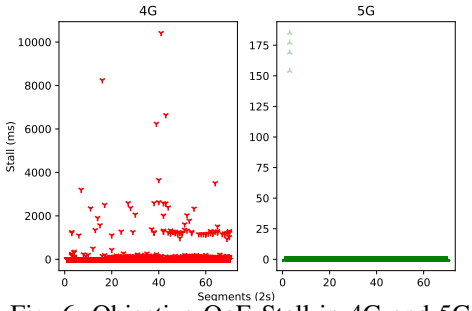


Fig. 6: Objective QoE Stall in 4G and 5G

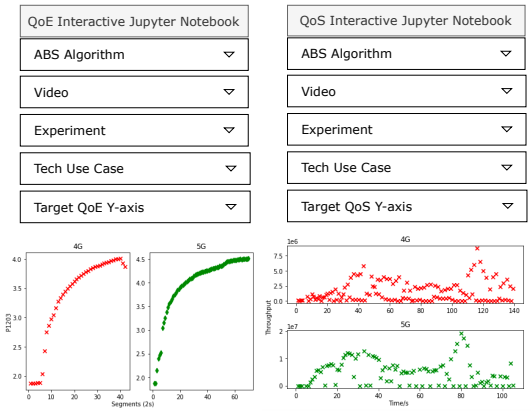


Fig. 7: Interactive Jupyter Notebooks for the QoE and QoS

5G technologies, we provide Interactive Jupyter Notebooks.[5] For the evaluation of QoE, we provide Notebook with nine objective QoE and five QoE Models (see Figure 7). In the first dropdown, select ABS algorithm – i) bba, ii) elastic, iii) conventional, followed by Video and Experiment. Note: We repeated each experiment 3-times with five different combinations of 4G and 5G use-cases. Next, select the 4G and 5G technology and the final selection is the target, which is nine objective QoE with five QoE Models (P.1203, Yin, Yu, Duanmu, Clae). The interactive notebook provides a subplot with 4G on the left and 5G on the right to compare both technologies. On the x-axis, we show segments which are a total of 70 and 70x2=140-seconds a duration of video streaming sessions. Whereas on the y-axis, we show the objective QoE and QoE Models.

2) QoS Interactive Jupyter Notebook: In this section, we provide Notebook to visualize the QoS metrics of the dataset generated with different combinations of 4G and 5G technologies. We provide 31 QoS features derived with a window of 1-second granularity. The QoS features are mentioned in Table I, and the Notebook layout is shown in Figure 7. The QoS Notebook drop down selection follows the same pattern as QoE Notebook. On the x-axis, we provide 1-second time granularity with a sequential increase up to the video streaming session time, which is 140-seconds. On the y-axis, we show the target QoS features. In Table V and Table VI we show the Mean, Standard Deviation, Min, Max, 25 %, 50 %, 75 % and Max value of throughput in Mbps for both the technologies 4G and 5G for the protocols TCP and QUIC respectively. We can see that 5G provides much better QoS than 4G for all the combinations of use-cases.

VI. QoS Features Correlation with QoE

This section explains the correlation of QoS features derived from EFFECTOR with the QoE Model P.1203 scores. To do that, we use a use case of commercial 4G and 5G datasets collected in the wild with Channel Level Metrics (CLM) and Emulate them in EFFECTOR (see Figure 11). The dataset is collected in France over a period of six months with YouTube as a baseline [22]. We collected 100+ CLM and objective QoE of YouTube with the smallest granularity of 1-second. We choose different use cases i) Pedestrian, ii) Mobility, iii) Indoor, and iv) Outdoor. We use G-NetTrack Pro[6] and YouTube IFRAME API.[7] We saved both QoE logs from the player such as i) Stall, ii) Video Bytes Downloaded, iii) Current Quality – Resolution, and iv) Time, along with Player Events such as 3 – Stall, 1 – Playing, 0 – Ended, 2 – Paused, 5 – Cued, -1 – Unstarted [23]. The complete explanation of each feature and broad objectives are out of the scope of this paper. However, the intuition behind here is to select the use case to show the QoS correlation derived from EFFECTOR with QoE KPIs. We select 4+ use cases from our own commercial dataset and emulate them in the framework with 1-second granularity.

The experimental setup is already explained in Section IV. We select two use cases from 4G and 5G, where we experience real stalling events and quality shifts (Change in Resolutions). Use cases are embedded in the framework and are also available on GitHub with QoE of the real YouTube traffic, 100+ CLM, and QoS and QoE of the emulation in the framework.

In Figures 8, 9, 10, we show two QoS metrics Throughput and EMA with two QoE metrics P.1203 score and Stalls using protocol – TCP, video – Tears of Steel and Segment – 2s. On the x-axis of two QoS features, there is time in seconds. We show a 1-second window QoS patterns. Whereas on the x-axis of two QoE metrics, there are video segments. Y-axis on each case shows the QoS and QoE values. We can see in

[5]https://bit.ly/3F6Oxdo

[6]https://bit.ly/3MU0Rj0

[7]https://bit.ly/3DiAuQD

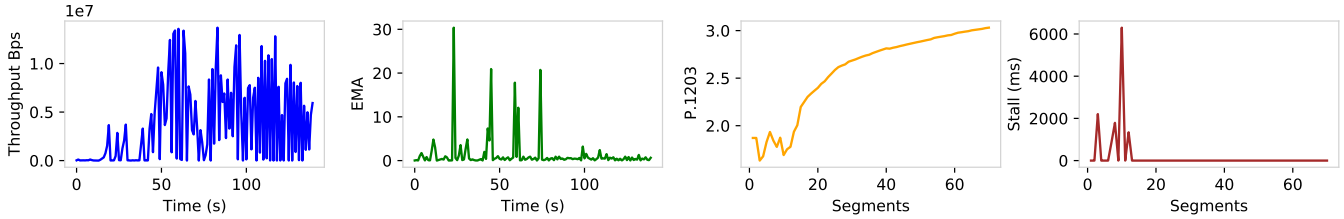


Fig. 8: QoS Correlation with Objective QoE – Stall and P.1203 Using BBA – ABS and a Window Size of 1-Second

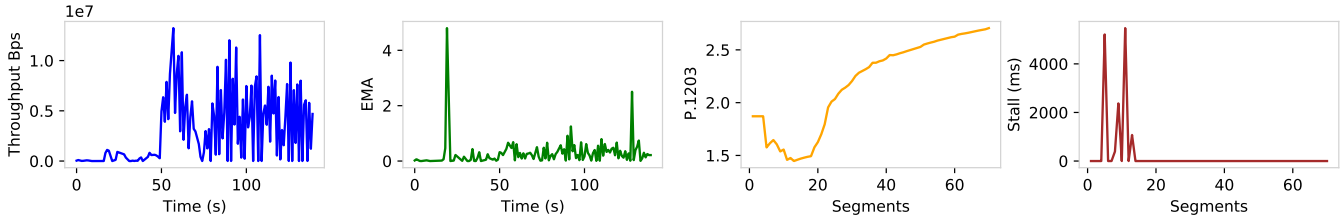


Fig. 9: QoS Correlation with Objective QoE – Stall and P.1203 Using Elastic – ABS and a Window Size of 1-Second

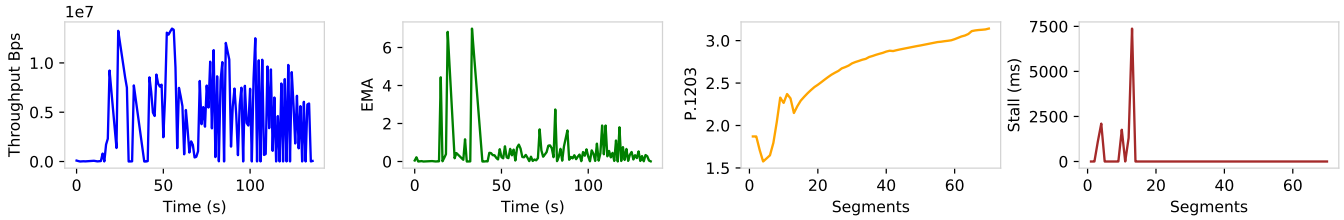


Fig. 10: QoS Correlation with Objective QoE – Stall and P.1203 Using Conventional – ABS and a Window Size of 1-Second

TABLE V: Throughput (Mbps) Across 4G and 5G on Different Use Cases (Video – Sintel) TCP

Technology	Mean	STD	Min	25 %	50 %	75 %	Max
4G	3.26	3.90	0.000416	0.17	1.41	5.60	19.39
5G	5.10	4.40	0.000416	0.625736	4.90	7.88	19.76

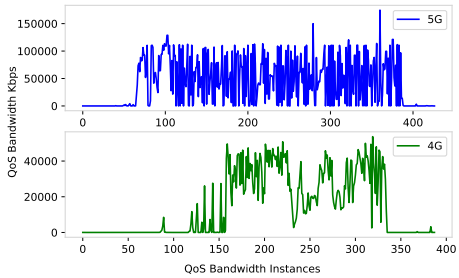


Fig. 11: 4G and 5G Use-cases Used from Commercial 4G and 5G Datasets to Emulate in EFFECTOR

each case of ABS, throughput and EMA values are highly correlated with P.1203 score and objective QoE KPI stall. The first 50-seconds of the streaming session suffer from low QoS resources given during emulation, thus, causing low throughput and QoE score. Low throughput is highly correlated with P.1203 score and thus causing stalling events during the first 20-segments of the streaming session. We consider a 2-second segment, thus 20x2=40-second of the streaming session. We can also see a few peaks of EMA values in the first 45-50-second of the streaming session, which shows the correlation of the QoS feature derived from IPG correlation with P.1203 and stalls.

VII. Conclusions

In order to provide a better adaptive streaming service experience (e.g., QoE), network and service operators are required to assess the DASH user's perceived performance. However, the assessment approach on end-to-end encrypted network traffic yields challenges for network operators.

In this paper, we presented EFFECTOR, a state-of-the-art adaptive streaming compatible framework with a QoS and QoE evaluation method to assess and correlate the user's perception of the DASH content when streamed through the encrypted network. Our proposed in-band QoS feature engineering method is based on monitoring network traffic at edge nodes in near real-time, which does not require chunk-level inspection but rather observing the pattern of network packet arriving time and volume. The framework has been evaluated in an emulated environment considering real-world circumstances in which DASH videos are played while subjected to 4G and 5G network performance. As a

TABLE VI: Throughput (Mbps) Across 4G and 5G on Different Use Cases (Video – Tears) QUIC

Technology	Mean	STD	Min	25 %	50 %	75 %	Max
4G	1.75	2.17	0.000416	0.39	1.05	2.20	15.08
5G	6.32	6.03	0.000416	0.08	5.02	11.99	25.40

result, this work produces a rich dataset of network-level extracted QoS, and application-level QoE features with a heterogeneous combination that is presented in the form of Interactive Jupyter Notebooks to visualize the trend of QoS and QoE. We believe our generated dataset is able to make a correlation between QoS and QoE in near real-time and session-wise. Such co-relation is highly relevant to network operators to review the service-level agreement for proactive network capacity planning and reactive QoE-aware network traffic management.

In future work, we aim to include machine learning sandbox to make the QoS-QoE correlation model and find out potential QoS feature influences on specific QoE KPIs. Also, we plan to add SQAPE[8] reference topology alike complex network scenario, `Mininet-WiFi` access node to replicate 4G and 5G trace's channel condition (e.g., SINR, RSRP/Q, CQI) and deliver the framework in a resource-efficient ready to use container format.

ACKNOWLEDGMENT

This research was supported by the Innovation Center, Ericsson S.A., Brazil. This document, however, does not necessarily represent Ericsson's official viewpoint. This study was partially funded by CAPES, Brazil - Finance Code 001. This work was also supported by the Sao Paulo Research Foundation (FAPESP), Brazil grants 2020/05182-3 and 2021/00199-8.

REFERENCES

[1] N. Panwar, S. Sharma, and A. K. Singh, "A survey on 5G: The next generation of mobile communication," *Physical Communication*, vol. 18, pp. 64–84, 2016.

[2] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring video qoe from encrypted traffic," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 513–526, 2016.

[3] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, pp. 187–198, 2014.

[4] L. Dinh-Xuan, M. Seufert, F. Wamser, and P. Tran-Gia, "Study on the accuracy of qoe monitoring for http adaptive video streaming using vnf," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 999–1004, IEEE, 2017.

[5] R. Ul Mustafa, S. Ferlin, C. Esteve Rothenberg, D. Raca, and J. J. Quinlan, "A supervised machine learning approach for dash video QoE prediction in 5G networks," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 57–64, 2020.

[6] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pp. 97–108, 2012.

[7] A. H. Zahran *et al.*, "ARBITER+: Adaptive Rate-Based InTElligent HTTP StReaming Algorithm for Mobile Networks," *IEEE Transactions on Mobile Computing*.

[8] M. J. Khokhar, T. Ehlinger, and C. Barakat, "From network traffic measurements to QoE for internet video," in *2019 IFIP Networking Conference (IFIP Networking)*, pp. 1–9, IEEE, 2019.

[9] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 1331–1339, IEEE, 2018.

[10] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring streaming video quality from encrypted traffic: Practical models and deployment experience," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–25, 2019.

[11] I. Orsolic and L. Skorin-Kapov, "A framework for in-network QoE monitoring of encrypted video streaming," *IEEE Access*, vol. 8, pp. 74691–74706, 2020.

[12] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Vicrypt to the rescue: Real-time, machine-learning-driven video-QoE monitoring for encrypted streaming traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2007–2023, 2020.

[13] N. F. S. de Sousa, M. T. Islam, R. U. Mustafa, D. A. L. Perez, C. E. Rothenberg, and P. H. Gomes, "Machine learning-assisted closed-control loops for beyond 5G multi-domain zero-touch networks," *Journal of Network and Systems Management*, vol. 30, no. 3, pp. 1–29, 2022.

[14] R. U. Mustafa, M. T. Islam, C. Rothenberg, S. Ferlin, D. Raca, and J. J. Quinlan, "Dash qoe performance evaluation framework with 5G datasets," in *2020 16th International Conference on Network and Service Management (CNSM)*, pp. 1–6, IEEE, 2020.

[15] R. Ul Mustafa and C. Esteve Rothenberg, "A framework for QoS and QoE assessment of encrypted video traffic with 4G and 5G open datasets," in *IEEE Globecom Demo Session*, 2022.

[16] D. Raca, M. Manifacier, and J. J. Quinlan, "Godash—go accelerated has framework for rapid prototyping," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pp. 1–4, IEEE, 2020.

[17] J. O'Sullivan, D. Raca, and J. J. Quinlan, "godash 2.0-the next evolution of has evaluation," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pp. 185–187, IEEE, 2020.

[18] J. J. Quinlan *et al.*, "Multi-profile Ultra High Definition (UHD) AVC and HEVC 4K DASH Datasets," in *9th ACM MMSys Conference*.

[19] D. Raca, D. Leahy, C. J. Sreenan, and J. J. Quinlan, "Beyond throughput, the next generation: a 5G dataset with channel and context metrics," in *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 303–308, 2020.

[20] D. G. Balan and D. A. Potorac, "Linux htb queuing discipline implementations," in *2009 First International Conference on Networked Digital Technologies*, pp. 122–126, IEEE, 2009.

[21] Dashframework, "Github repository." https://github.com/razaulmustafa852/EFFECTOR, 2022.

[22] R. Ul Mustafa, C. Esteve Rothenberg, and C. Barakat, "YouTube goes 5G: Benchmarking YouTube in 4G vs 5G." https://dx.doi.org/10.21227/h00h-ew92, IEEE DataPort, 2022.

[23] R. Ul Mustafa, C. Esteve Rothenberg, and B. Chadi, "Youtube goes 5G: Benchmarking youtube in 4G vs 5G through open datasets," in *IEEE Globecom Demo Session*, 2022.

[8] https://bit.ly/3Qo6y97