Drop the Packets: Using Coarse-grained Data to detect Video Performance Issues

Tarun Mangla*
University of Chicago

Emir Halepovic
AT&T Labs – Research

Ellen Zegura
Georgia Institute of Technology

Mostafa Ammar
Georgia Institute of Technology

ABSTRACT

Understanding end-user video Quality of Experience (QoE) is important for Internet Service Providers (ISPs). Existing work presents mechanisms that use network measurement data to estimate video QoE. Most of these mechanisms assume access to packet-level traces, the most-detailed data available from the network. However, collecting packet-level traces can be challenging at a network-wide scale. Therefore, we ask:"Is it feasible to estimate video QoE with lightweight, readily-available, but coarse-grained network data?" We specifically consider data in the form of Transport Layer Security (TLS) transactions that can be collected using a standard proxy and present a machine learning-based methodology to estimate QoE. Our evaluation with three popular streaming services shows that the estimation accuracy using TLS transactions is high (up to 72%) with up to 85% recall in detecting *low* QoE (*low* video quality or *high* re-buffering) instances. Compared to packet traces, the estimation accuracy (recall) is 7% (9%) lower but has up to 60 times lower computation overhead.

CCS CONCEPTS

• Networks → Network measurement; **• Information systems → Multimedia streaming**;

KEYWORDS

QoE, Video streaming, Passive measurements

ACM Reference Format:
Tarun Mangla, Emir Halepovic, Ellen Zegura, and Mostafa Ammar. 2020. Drop the Packets: Using Coarse-grained Data to detect Video Performance Issues. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3386367.3431294

1 INTRODUCTION

Last-mile Internet Service Providers (ISPs), especially cellular ISPs, need to efficiently provision and manage their networks to meet the growing demand for Internet video [1, 19]. This network optimization requires ISPs to have an in-depth understanding of end-user video Quality of Experience (QoE). Understanding video QoE is, however, challenging for ISPs as they generally do not have access to applications at end-user devices. This is further exacerbated by an increasing use of end-to-end encryption, significantly limiting the information ISPs can obtain from the network traffic to estimate video QoE. ISPs are thus constrained to rely on their limited view of the network data to estimate video QoE metrics.

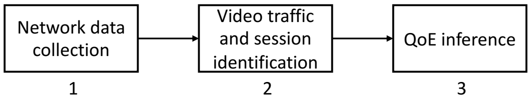


Figure 1: QoE inference steps

Video QoE estimation using network data primarily consists of three steps: i) collecting network data using a monitoring tool, ii) identifying video traffic and sessions from collected data, and iii) estimating session QoE metrics using methods designed for this purpose (see Figure 1). Prior work in this domain has mainly focused on designing QoE estimation mechanisms (step 3 in Figure 1) with a goal to improve inference accuracy [12, 14, 17, 22, 24, 25]. In doing so, most of these mechanisms assume access to packet traces, the most granular network data. However, collecting and processing packet-level data from the entire network is challenging because of the scale of ISP networks [10, 13]. At the same time, it is important for ISPs to understand network-wide video performance for efficient management and provisioning, especially in the case of capacity-constrained and highly heterogeneous cellular networks. This makes it challenging to use existing QoE estimation mechanisms in practice.

One possible approach is to develop flexible telemetry systems that provide the most useful metrics (e.g., HTTP transactions) required for inference by in-network processing of the packet data [8, 21, 28]. While this is a viable approach, it involves significant modifications to the existing measurement systems and has the following practical challenges, i) limited measurement resources and budget with the constraint that the same network data is often used for multiple purposes (e.g., security, performance), and ii) limited flexibility since such tools are provided by vendors [37].

Given these challenges, we ask: *"Is it feasible to detect video performance issues with lightweight, readily-available but coarse-grained network data?"* Our question is motivated by the fact that ISPs already collect coarse-grained data using standard telemetry systems for different network management functions [23, 33, 35, 37]. We consider whether such data can be used by ISPs to estimate coarse-grained QoE metrics (e.g., *low*, *high*) and thus to identify parts of the network that underperform in a lightweight manner.

[*] Work done while at Georgia Institute of Technology

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CoNEXT '20, December 1–4, 2020, Barcelona, Spain
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431294

Ultimately, this approach can enable adaptive video performance monitoring wherein an ISP collects fine-grained data only from the problematic locations for further diagnosis.

We specifically consider coarse-grained network data in the form of Transport Layer Security (TLS) transactions. The data is clearly lightweight as number of TLS transactions in a video session is significantly smaller (by a factor of 1400 in our dataset) as compared to packets. The data is also readily available as TLS transactions can be collected using a transparent proxy (e.g., Squid [4]). In fact, most cellular ISPs already use a transparent proxy for various network management functions (e.g., traffic accounting, differentiation) and such proxy has an off-the-shelf capability to report TLS transactions [3]. Finally, video traffic can be easily identified (step 2 in Figure 1) using the headers from TLS transaction data. Prior work has used similar data to infer QoE for web traffic [30] and unencrypted video[1] [23]. A major challenge, however, is that the TLS transaction data is coarse-grained. Thus, existing inference techniques will not work on this form of data. Another challenge in using this data is to delimit sessions[2] when a user watches back-to-back videos from the same service. Accurate session identification is important for accurate QoE estimation due to changes in streaming patterns and the corresponding traffic within a session as it progresses (see Section 2).

Therefore, we analyze the feasibility of using TLS transaction data to detect video performance issues. Specifically, we consider categorical estimation of key video QoE metrics [18, 20], namely, video quality, re-buffering ratio and a combined QoE metric that jointly considers the two individual metrics (Section 2). We first develop a machine learning (ML)-based approach that builds on previous work by adapting ML-based techniques to TLS transaction data. We evaluate our methodology using data collected under diverse emulated network conditions from three streaming services, namely, YouTube, Netflix, and Hulu (anonymized in the paper). We also compare the QoE estimation accuracy using TLS transaction data against packet traces. Finally, we present a simple heuristic to distinguish consecutive sessions from the same video service leveraging TLS transaction arrival and server access patterns.

Our key findings are summarized below:

- The TLS transaction data can be used to estimate combined QoE metric (Section 2) with an accuracy of up to 72% and detect *low* QoE (*low* video quality or *high* re-buffering) instances with a recall of 73%-85%.
- Compared to packet traces with an existing ML-based approach [12], estimation using TLS transaction data has up to 7% (9%) lower accuracy (recall), but it has 1400x lower memory overhead and 60x lower computation overhead.
- The session identification heuristic can accurately identify 89% of the consecutive sessions.

2 TARGET QOE AND NETWORK DATA

Most of the Internet video is streamed using a class of techniques, called HTTP-based Adaptive Streaming (HAS), that dynamically adapt the video quality based on the network conditions. In HAS, the video is divided into segments with each segment encoded into a pre-defined set of quality levels. The player at the client downloads video segments by sending HTTP requests. The quality of the segments is determined by the adaptation algorithm used in the player [15, 16, 36]. Here we describe the HAS QoE metrics we estimate and the network data used for their inference.

2.1 Target QoE metric

QoE in HAS is impacted by a variety of factors, namely, re-buffering, video quality, startup delay, and quality variations [6, 18, 20, 26]. Existing inference approaches estimate these objective QoE metrics for a video session in two different ways: *fine-granular* and *per-session*. The former estimates QoE metrics periodically within a session while the latter provides estimates only once for the entire session. The QoE estimation granularity of an approach is clearly impacted by the granularity of the input network data. Given that the data we use is coarse-granular, we consider categorical estimates (i.e, *low*, *medium*, and *high*) of *per-session* video QoE metrics. Such estimation enables ISPs to identify network locations with video performance issues in a lightweight manner. Specifically, we estimate the following key video QoE metrics [20]:

Re-buffering ratio (*rr*): It is defined as the stall time in proportion to the total playback time and measures the severity of stalls in a session. We classify *rr* into the following three categories: i) *zero*, if there are no stalls, ii) *mild*, if $0 < rr \leq 2\%$, and iii) *high*, otherwise.

Video quality: In HAS, videos are typically encoded into discrete quality levels with more bits typically required to encode higher video quality. The quality levels tend to be same for a video service (e.g., Netflix) and streaming protocol (e.g., HLS, DASH) combination[3]. We set thresholds and categorize the quality levels to *low*, *medium*, and *high* (see Section 4). The *video quality* of a session is defined as the majority category of the quality level played in a session [34]. In case of a tie, we select the lower category.

Combined QoE: We estimate the *combined QoE* of a session by jointly considering the individual QoE metrics. There are a number of ways to combine the individual metrics [6, 36]. In this paper, we use a simple approach of using the minimum category of the two QoE metrics. E.g., if a session had *zero* re-buffering but *low* video quality, its overall QoE is assigned to *low*. Our methodology can also work for other combinations.

Thus, for each session we estimate the categorical values of *video quality*, *re-buffering ratio*, and *combined QoE*.

2.2 Network data

ISPs typically collect different kinds of data from within their network which includes network device-level data (e.g., SNMP logs [29]) and aggregate statistics from passive traffic monitoring (e.g., NetFlow [9] and Proxy data [4]). Clearly, device-level data cannot be used to even identify video traffic, let alone assess its QoE. Therefore, we consider aggregate network traffic data that can be collected with standard monitoring tools for QoE inference.

Specifically, we consider encrypted network traffic data in the form of TLS transactions collected using a transparent proxy (e.g., Squid [4]) that inspects the unencrypted TLS headers. A major challenge, however, is that the TLS transaction data is coarse-granular.

[1] For unencrypted video, a proxy provides HTTP transactions

[2] Our definition of a session consists of streaming a single video

[3] Some videos may not be available at all quality levels. A service may use different quality levels based on the content type (e.g, live, on-demand).

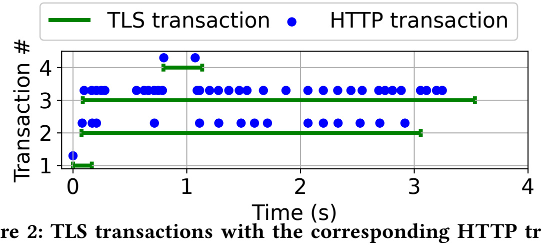


Figure 2: TLS transactions with the corresponding HTTP transactions within first 5 seconds of a Svc1 session. For clarity, only start of the HTTP transactions is shown.

Figure 2 shows the TLS transactions within the first 5 seconds of a sample session from Svc1 with the corresponding HTTP transactions[4], considered important for QoE inference in the related work [14]. Clearly, a single TLS transaction contains multiple and variable number of HTTP transactions. We observed an average of 12.1 HTTP transactions corresponding to every TLS transaction for the Svc1 sessions in our dataset (see Section 4).

It can also be challenging to correctly delimit session boundaries using TLS transaction data if multiple videos from the same service are watched back-to-back by a user. This is because the active TLS transactions do not always end immediately once the player is closed, but timeout after some duration, leading to overlapping transactions for consecutive sessions. Therefore, a timeout-based approach, wherein a session boundary is detected if there is no more video traffic for a certain time, would not work. Inaccurate session identification can lead to errors in QoE estimation due to differences in buffering state and steady state network characteristics in HAS [5]. Existing work suggests heuristic based on fine-granular traffic size information [8] which will not work with TLS transaction data due to its coarse-granularity.

Our goal is to analyze the feasibility of using such coarse-granular but readily-available and lightweight data to estimate video QoE. We consider two kinds of information available in a TLS transaction: i) start and end time, and uplink and downlink size, and ii) Server Name Indicator (SNI) field indicating the server hostname. We use the former for QoE estimation and the latter for video traffic and session identification.

We note that flow-level monitoring (e.g., NetFlow [9]) is another popular measurement technique. In fact, flow record data with size counters from Netflow is similar to TLS transaction data as there is typically a single TLS transaction in a TCP connection. Flow-level monitoring also provides an option of obtaining periodic summaries from long flows. A major challenge, however, with flow-level monitoring is identification of video traffic as it lacks application-layer data. Existing work has suggested solutions like augmenting flows with DNS information [7]. We consider using flow-level data as a part of future work and focus on understanding feasibility of using TLS transaction data for inference.

3 METHODOLOGY

We formulate the QoE estimation problem as a supervised machine learning problem. This is similar in spirit to existing work designed for packet data [11, 24]. We develop features specific to the coarse-granular TLS transaction data based on the semantics of HAS. We assume that the TLS transactions corresponding to video traffic have already been identified (e.g., using SNI field) and grouped into sessions. Later, we also present a heuristic to delimit TLS transactions corresponding to consecutive sessions from the same service. Here, we describe the three kinds of features constructed from the sequence of TLS transactions of a session below:

Session-level: These features consist of metrics calculated for the entire session. We calculate the session data rate, which is the total data divided by the session duration, in both downlink (SDR_DL) and uplink (SDR_UL) directions. In addition, we also log the session duration (SES_DUR) and the number of TLS transactions per second ($TRANS\_PER\_SEC$).

Transaction-level statistics: For a transaction, we already have its downlink size (DL_SIZE), uplink size (UL_SIZE), and duration (DUR). In addition, we calculate the following three metrics for every transaction: i) *Transaction Data Rate* (TDR), which is obtained by dividing the downlink data size by the transaction duration. Note that TDR is not the same as network throughput as there can be idle intervals in a TLS transaction with no network activity [5]. However, it is still an indicator of network quality as, intuitively, TDR is high if the available bandwidth was high. ii) *Downlink-To-Uplink* ($D2U$) *ratio*, which is the ratio of the downlink data to the uplink data. In HAS, the uplink data is typically an indicator of the number of video segments requested [31]. Hence, $D2U\ ratio$ represents the amount of data downloaded per segment. This can be a useful indicator of the video quality, and iii) *Inter-arrival time* (IAT) of the transactions to capture patterns in arrival of transactions.

Thus, we have 6 metrics for each transaction. From these metrics, we generate summary statistics, namely, minimum, median, and maximum value leading to 18 features in total [5].

Temporal Features: These features capture the temporal progress of data transfer during a session. We divide the session into predetermined intervals each starting from the beginning of the session and calculate the cumulative downlink (CUM_DL_XXs) and uplink data (CUM_UL_XXs) in each of these intervals. For transactions that only partially overlap with an interval, we get its share of downlink and uplink data based on the extent of the overlap with the interval[6]. This set of features can be useful in uncovering any temporal variations which may have been masked out in the aggregate transaction statistics.

We consider the following end-points for the intervals (in seconds): {30, 60, 120, 240, 480, 720, 960, 1200}. We use a maximum value of 1200 seconds as this is the maximum session duration in our dataset (see Section 4). The rationale behind using fine-granular intervals in the beginning is that a session is more likely to be impacted by poor network quality in the beginning because of empty video buffer. We explored other intervals (omitted due to lack of space) but found the above to yield the highest accuracy. Regardless, we consider these intervals as one of the hyperparameters of our model and an ISP can determine the intervals based on the data observed on their network for a service.

Thus, we use a total of 38 (4 + 18 + 16) features for each session (summarized in Table 3) to estimate its QoE metrics.

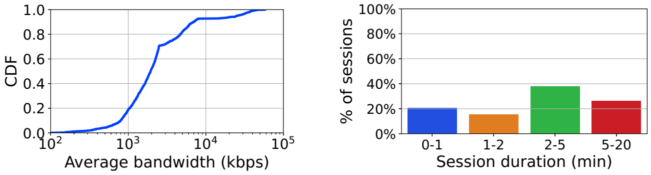
[4] The HTTP transactions are derived from packet traces [17].

[5] We considered other statistics such as standard deviation and mean, but found them to be highly correlated to one of the existing statistics.

[6] This is an approximation as it is not possible to determine the data transmission pattern within a transaction

Type	Statistic	Features
Session level	single value	SDR_DL, SDR_UL, SES_DUR, TRANS_PER_SEC
Transaction Statistics	MIN, MED (median), MAX	DL_SIZE, UL_SIZE, DUR, TDR, D2U, IAT
Temporal Statistics	interval based	CUM_DL_XXs, CUM_UL_XXs

Table 1: Summary of features



(a) Average bandwidth

(b) Duration

Figure 3: Bandwidth traces statistics

4 EVALUATION

We evaluate the QoE estimation accuracy using TLS transaction data and compare it with packet traces. We first describe the methodology used to collect the dataset for evaluation.

4.1 Data collection

We use a browser-based automation framework to collect data for training and testing. The framework streams video sessions under emulated network conditions and collects network data in the form of packet traces and TLS transactions. We emulate network conditions using publicly available bandwidth traces representing a diversity of network environments including fixed broadband, 3G and LTE [2, 27, 32]. Each session is streamed for a duration ranging from 10-1200 seconds. Figure 3a and 3b show the distribution of average bandwidth and duration of the traces, respectively.

Using the above methodology, we collect data for three popular streaming services which are denoted as Svc1, Svc2, and Svc3 (anonymized for confidentiality). We curate a list of 50-75 videos for each service including content from different genres such as animation, sports, and news, if available. The ground truth video QoE metrics are collected per second by injecting Javascript functions utilizing the HTML5 Video API to monitor re-buffering and service-specific functions (determined manually) to monitor video quality [8, 22]. The video quality levels are classified into one of the three categories. We use resolution-based thresholds in Svc1 and Svc2 as these services had a unique resolution per quality level. For Svc2, we classify video resolution of *360p* or lower as *low*, *480p* as *medium*, and *720p* or higher as *high*. The thresholds for Svc1 were *288p* for *low*, *480p* for *medium*, and remaining were tagged as *high*. For Svc3, we observed only three quality levels in our dataset and classify them into *low*, *medium*, and *high*. In practice, these thresholds can be set by the ISP based on its target quality. We use the per-second QoE information to obtain categorical values of per-session video quality, re-buffering ratio, and combined QoE.

Overall we had $2,111$ sessions for Svc1, $2,216$ for Svc2, and $1,440$ for Svc3. We observe difference in ground truth QoE metrics across services (see Figure 4) for sessions streamed under similar network conditions. This can be attributed to differences in the service design. We found that Svc1 uses a larger video buffer (240s) as compared to the other two services. Furthermore, Svc1 player attempts to avoid re-buffering by quickly filling the buffer at the expense of streaming at low video quality. However, the other two services, especially Svc2, switch video quality only when the video buffer runs low. Therefore, poor network conditions generally led to low video quality in Svc1, whereas in Svc2 and Svc3 (although to a lesser extent), it led to re-buffering.

4.2 Results

We use the Python Scikit library to train different machine learning models and use 5-fold cross validation for evaluating accuracy. We tested different ML-based models, namely SVM, k-NN, XGBoost, Random Forest, and Multilayer Perceptron. Here, we present results using Random Forest (others omitted due to lack of space) as it yielded the highest accuracy.

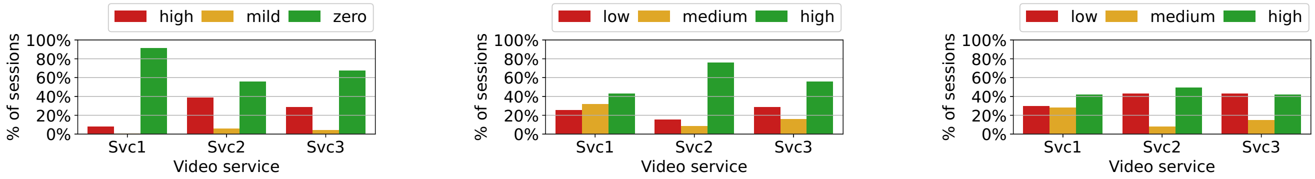
Accuracy for different QoE metrics: Figure 5 shows the classification accuracy of different QoE metrics in Svc1 and Svc2. While we report overall accuracy, and precision and recall values for *low* QoE metric class, we particularly focus on the recall value as one of our main goals is to correctly identify network locations with video performance issues. Thus, it is important to identify the true positives (*low* QoE sessions) with a high accuracy. ISPs can collect additional data, such as fine-grained network traces or readily available radio metrics (for cellular networks) in the location, for further fault diagnosis and management. For Svc1, the recall in identifying *low* video quality sessions is 68%, while the recall is only 21% in identifying *high* re-buffering (see Figure 5a). The trend is reversed for Svc2 with 71% recall for *high* re-buffering and only 40% recall for *low* video quality (see Figure 5b). The results are similar for Svc3 with 63% recall for *high* re-buffering and 58% for *low* video quality. In general, we observe that the accuracy metrics are high for the QoE metric that is more likely to degrade with poor network conditions in a video service. The accuracy metrics are high for the combined QoE metric across all three services with 73%-85% recall in identifying *low* combined QoE.

Table 2 shows the confusion matrix for the combined QoE metric in Svc1. Most of the mis-classifications happen between neighboring classes (e.g., *low* classified as *med*). This is most likely due to the model's inability in classifying instances that are closer to the class thresholds. Naturally, the error is higher for sessions with *medium* QoE, while the sessions with *low* or *high* combined QoE can be classified with a high accuracy across all three services.

Takeaway: The coarse-grained TLS transaction data can enable ISPs to detect video performance issues *aka low* combined QoE sessions with a high accuracy. In the remaining paper, we focus on results pertaining to combined QoE.

Feature importance: We next evaluate the impact of different kinds of feature on model accuracy. Table 3 shows the accuracy as features are incrementally added to the model. The recall (accuracy) is lowest when only session-level features are used and it improves by 6%-12% (6%-11%) as features capturing the transaction statistics and temporal distribution of data are added to the model. This shows that despite being coarse-granular, TLS transactions within a session can provide useful information about the QoE of a session.

Figure 6 shows the 10 most important features as reported by the Random Forest model across the three services. There are 4 features that appear in the top 10 list of all three services. These features



(a) Re-buffering ratio

(b) Video quality

(c) Combined QoE

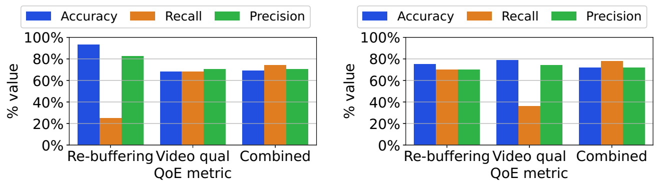
Figure 4: Distribution of QoE metrics across services

Actual	# sessions	Predicted low	Predicted med	Predicted high
low	632	72%	21%	8%
med	599	25%	43%	32%
high	880	5%	12%	84%

Table 2: Confusion matrix: Svc1, Combined QoE

Feature set	Svc1			Svc2			Svc3		
	A	R	P	A	R	P	A	R	P
Only Session-level (SL)	58%	61%	60%	66%	68%	63%	66%	77%	66%
SL + Transaction Stats (TS)	65%	72%	67%	69%	77%	68%	71%	84%	74%
SL + TS + Temporal Stats	69%	73%	71%	71%	78%	71%	73%	85%	75%

Table 3: Accuracy (A), Recall (R), and Precision (P) values for different feature sets



(a) Svc1

(b) Svc2

Figure 5: Accuracy for different QoE metrics

are downlink session data rate (*SDR_DL*), median transaction data rate (*TDR_MED*), median D2U ratio (*D2U_MED*), and the cumulative downlink data in the first minute (*CUM_DL_60s*). *TDR_MED* and *SDR_DL* represent the downlink data rate and hence capture information about the available bandwidth. *D2U_MED* represents the downlink to uplink data ratio and is likely to be higher when the video quality is high and vice-versa. Finally, *CUM_DL_60s* represents the data downloaded in the beginning of the session when the video buffer is usually low and when a session is more likely to suffer if the network conditions are poor. We also observe differences across services with 8 features that appear in only one out of the three services. This is likely due to the differences in service design and TLS transaction mechanisms across services.

We also empirically illustrate the usefulness of transaction-level statistics and temporal features by considering sessions that had similar session-level features. Figure 7a presents a box plot of *CUM_DL_60s* for Svc1 sessions with duration between 2 and 3 minutes and downlink session data rate between 1400 kbps and 1600 kbps. The number of instances are displayed at the top of each box. There is a clear difference in the distribution across *low* and *high* QoE sessions. The 25th (50th) percentile of *CUM_DL_60s* for *low* QoE sessions is 17 MB (21 MB), while it is 23 MB (24 MB) for *high* QoE sessions. We found similar differences for *D2U_med* for Svc2 sessions as shown in Figure 7b. We also find that the distribution of *medium* QoE sessions overlaps with the other QoE classes, thus, indicating it is challenging to classify these sessions.

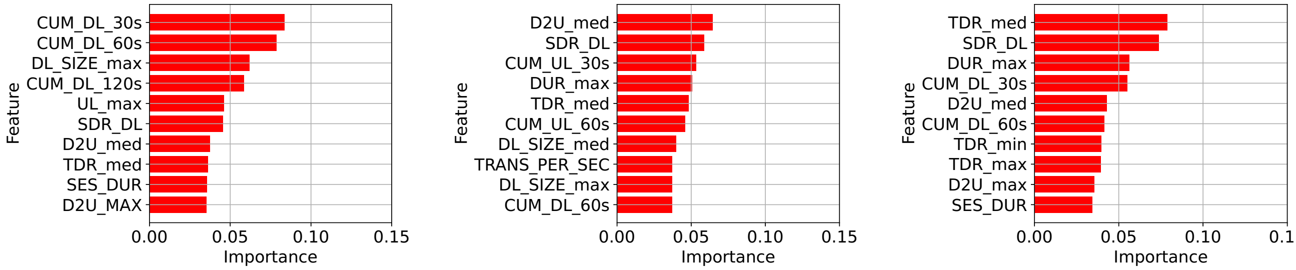
Takeaway: The analysis shows that in addition to session-level metrics such as duration and downlink data rate, there are also patterns within the TLS transactions of a session that differ based on the session QoE. An ML-based approach can learn these patterns to identify *low* QoE sessions.

Comparison with packet traces: We now compare the QoE estimation accuracy from TLS transaction data against packet traces. Multiple ML-based have been proposed in the related work to estimate QoE using packet traces [8, 14, 17]. Most of them are designed for real-time QoE inference and estimate metrics for every time window *T*. A comparison with these approaches would require estimation of per-session metrics from fine-granular estimation. For simplicity, we consider an algorithm that directly estimates per-session metrics. More specifically, we implement an algorithm proposed by Dimopoulos et al. called ML16 [12]. The algorithm uses features corresponding to video segments along with network metrics such as packet retransmissions, loss, and RTT. Furthermore, we use the feature set ML16 used for estimating video quality for combined QoE metric as it is a superset of the features used to estimate re-buffering.

Table 4 shows the accuracy metrics with respective gains in comparison to TLS transaction data. Using packet traces with ML16 results in an improvement of 5%-7% in overall accuracy and 4%-9% in recall for *low* combined QoE. This is intuitive as packet traces are highly fine-granular. Moreover, they can be used to derive information about video segments downloaded in a session which are fundamental to HAS and its QoE. We then compare the associated memory and computation overhead. In our dataset, the average number of packets per session in Svc1 are 27, 689 as compared to only 19.5 TLS transactions. The total computation time to extract relevant features from all Svc1 sessions using packet data is around 503 seconds as compared to only 8.3 seconds using TLS transaction data, a difference of factor of 60.

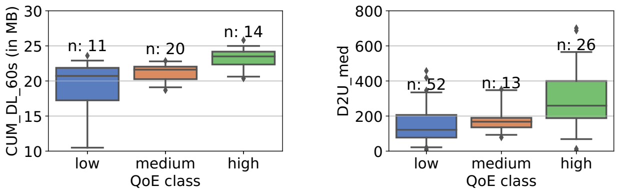
Takeaway: Packet traces provide higher accuracy than the TLS transaction data but with a significant computation and memory overhead. Therefore, ISPs can implement adaptive monitoring, wherein fine-granular network data is collected only for locations where performance issues are detected.

Session identification heuristic: Recall that session identification using TLS data can be a challenge for back-to-back sessions due to overlapping transactions. We develop a simple heuristic for session detection that is based on the following two insights: i)



(a) Svc1 **(b) Svc2** **(c) Svc3**

Figure 6: Top 10 important features across three services



(a) Svc1: ***CUM_DL_60s*** **for sessions with duration 2-3 minutes and** ***SDR_DL*** **between 1400-1600 kbps**

(b) Svc2: ***D2U_med*** **for sessions with duration between 2-3 minutes and** ***SDR_DL*** **between 1000-1200 kbps**

Figure 7: Distribution of sample transaction-level statistics and temporal features for a subset of sessions with similar session-level features (number of instances displayed at the top of each box)

Service	Accuracy	Recall	Precision
Svc1	74% (+5%)	82% (+9%)	73% (+2%)
Svc2	78% (+7%)	85% (+7%)	76% (+5%)
Svc3	78% (+5%)	89% (+4%)	78% (+3%)

Table 4: Accuracy using packet traces and ML16. Parenthesis values report the gain compared to TLS transaction data.

Actual	# Transactions	Predicted	
		Existing	New
Existing	13269	98%	2%
New	1545	11%	89%

Table 5: Transaction identification accuracy

The beginning of a session is characterized by more than one TLS transaction, and ii) The set of servers serving content are likely to change when a new session begins. Thus, for each transaction we consider the set of succeeding transactions starting within W seconds. Using these set of transactions, we calculate N, the number of transactions in the set, and δ, the percentage of transactions with a different server than the set of servers seen for the current session. A transaction is considered to start a new session, if N and δ are greater than N_{min} and δ_{min}, respectively. We use the following parameter values, $W = 3$ seconds, $N_{min} = 2$, and $\delta_{min} = 0.5$.

Table 5 shows the confusion matrix for Svc1 sessions with session beginnings correctly identified for 89% of the sessions. A timeout-based heuristic would have considered all of them as a single session as all these sessions were streamed back-to-back. We note that this is an extreme case compared to real-world scenario.

Takeaway: Session identification techniques need to be designed for the specific network data. The transaction arrival and server request pattern can enable accurate session identification for TLS transaction data.

4.3 Limitations

- **Streaming application design**: In addition to using volumetric features at a session-level, we rely on the patterns of data transfer across TLS transactions within a session for QoE inference. Clearly, the extent of such patterns and consequently the ability to infer QoE depends on the design of the streaming application. This is also observed in variance of important features across services in Figure 6. In an extreme case, an application may be designed to stream the entire session over a single TLS connection, thus, rendering the transaction-level statistics and temporal features used in our model ineffective.
- **Impact of user interactions**: Our experiments do not consider the impact of user interactions on QoE inference. Different kinds of user interactions, such as pausing and skipping, would manifest in different ways in the TLS transaction data. Understanding the impact of user interactions on inference accuracy is a part of the future work.
- **Real-time QoE inference**: TLS transaction information is available from the proxy only after the underlying TLS connection terminates. Therefore, our approach is not suitable for inferring and managing user dissatisfaction in real-time.

5 CONCLUSION

We find that coarse-grained but readily-available TLS transaction data can be used to estimate video QoE with reasonable accuracy and low overhead. The predictive capability can be attributed to two factors: i) downlink data-related features that capture network quality, ii) differences in TLS transaction statistics for *low* and *high* QoE sessions. Our future work will analyze the generalizability of the models across different device platforms and service types (e.g., live content). We also plan to more deeply explore the accuracy vs. scalability trade-off for other forms of network data such as more granular flow-level data collected using NetFlow.

ACKNOWLEDGEMENTS

We thank our shepherd and the anonymous reviewers for their comments and feedback. This work is funded in part by NSF grant NETS 1909040.

REFERENCES

[1] 2020. Cisco study. (2020). https://goo.gl/uz3SCN

[2] 2020. FCC dataset. (2020). https://www.fcc.gov/measuring-broadband-america

[3] 2020. Multi-Service Proxy. (2020). https://docuri.com/download/msp_59c1d020f581710b28642614_pdf

[4] 2020. Squid: Optimising Web Delivery. (2020). http://www.squid-cache.org/

[5] S. Akhshabi, A. C. Begen, and C. Dovrolis. 2011. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proc. ACM MMSys*.

[6] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. 2013. Developing a Predictive Model of Quality of Experience for Internet Video. In *Proc. of ACM SIGCOMM*.

[7] Ignacio N Bermudez, Marco Mellia, Maurizio M Munafo, Ram Keralapura, and Antonio Nucci. 2012. DNS to the rescue: discerning content and services in a tangled web. In *Proc. of ACM IMC*.

[8] Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Guilherme Martins, Renata Teixeira, and Nick Feamster. 2019. Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience. *Proc. of ACM SIGMETRICS* (2019).

[9] B. Claise. 2004. Cisco Systems NetFlow Services Export Version 9. IETF RFC 5246. (October 2004).

[10] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. 2003. Gigascope: a stream database for network applications. In *Proc. of ACM SIGMOD*.

[11] G. Dimopoulos, P. Barlet-Ros, and J. Sanjuàs-Cuxart. 2013. Analysis of YouTube user experience from passive measurements. In *Proc. of CNSM*.

[12] Giorgos Dimopoulos, Ilias Leontiadis, Pere Barlet-Ros, and Konstantina Papagiannaki. 2016. Measuring Video QoE from Encrypted Traffic. In *Proc. of ACM IMC*.

[13] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: Query-driven streaming network telemetry. In *Proc. of ACM SIGCOMM*.

[14] Craig Gutterman, Katherine Guo, Sarthak Arora, Xiaoyang Wang, Les Wu, Ethan Katz-Bassett, and Gil Zussman. 2019. Requet: Real-time QoE detection for encrypted youtube traffic. In *Proc. of ACM MMSys*.

[15] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proc. of ACM SIGCOMM*.

[16] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proc. of ACM CoNEXT*.

[17] Vengatanathan Krishnamoorthi, Niklas Carlsson, Emir Halepovic, and Eric Petajan. 2017. BUFFEST: Predicting Buffer Conditions and Real-time Requirements of HTTP(S) Adaptive Streaming Clients. In *Proc. of ACM MMSys*.

[18] S. Shunmuga Krishnan and Ramesh K. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-experimental Designs. In *Proc. of ACM IMC*.

[19] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. 2019. A large-scale analysis of deployed traffic differentiation practices. In *Proc. of ACM SIGCOMM*.

[20] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. 2015. Deriving and Validating User Experience Model for DASH Video Streaming. *IEEE Transactions on Broadcasting* (2015).

[21] Frank Loh, Florian Wamser, Christian Moldovan, Bernd Zeidler, Dimitrios Tsilimantos, Stefan Valentin, and Tobias Hoßfeld. 2020. Is the Uplink Enough? Estimating Video Stalls from Encrypted Network Traffic. In *Proc. of IEEE/IFIP NOMS*.

[22] Tarun Mangla, Emir Halepovic, Mostafa Ammar, and Ellen Zegura. 2018. eMIMIC: Estimating HTTP-based Video QoE Metrics from Encrypted Network Traffic. In *Proc. of IEEE/IFIP TMA*.

[23] Tarun Mangla, Emir Halepovic, Rittwik Jana, Kyung-Wook Hwang, Marco Platania, Mostafa Ammar, and Ellen Zegura. 2018. VideoNOC: Assessing Video QoE for Network Operators using Passive Measurements. In *Proc. of ACM MMSys*.

[24] M. H. Mazhar and Z. Shafiq. 2018. Real-time Video Quality of Experience Monitoring for HTTPS and QUIC. In *Proc. of IEEE INFOCOM*.

[25] Irena Orsolic, Dario Pevec, Mirko Suznjevic, and Lea Skorin-Kapov. 2017. A machine learning approach to classifying YouTube QoE based on encrypted network traffic. *Proc. of Springer, Multimedia tools and applications* (2017).

[26] Alexander Raake, Marie-Neige Garcia, Werner Robitza, Peter List, Steve Göring, and Bernhard Feiten. 2017. A bitstream-based, scalable video-quality model for HTTP adaptive streaming: ITU-T P. 1203.1. In *Proc. of IEEE QoMEX*.

[27] Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2011. Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning. *ACM TOMCCAP* (2011).

[28] Michael Seufert, Pedro Casas, Nikolas Wehner, Li Gang, and Kuang Li. 2019. Stream-based machine learning for real-time QoE analysis of encrypted video streaming traffic. In *Proc. of IEEE ICIN*.

[29] William Stallings. 1998. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.* Addison-Wesley Longman Publishing Co., Inc.

[30] Martino Trevisan, Idilio Drago, and Marco Mellia. 2017. PAIN: A Passive Web Speed Indicator for ISPs. In *Proc. of ACM, Internet QoE*.

[31] Dimitrios Tsilimantos, Theodoros Karagkioules, and Stefan Valentin. 2018. Classifying Flows and Buffer State for Youtube's HTTP Adaptive Streaming Service in Mobile Networks. In *Proc. of ACM MMSys*.

[32] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. 2016. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Comm. Letters* (2016).

[33] Zhaoguang Wang, Zhiyun Qian, Qiang Xu, Zhuoqing Mao, and Ming Zhang. 2011. An Untold Story of Middleboxes in Cellular Networks. In *Proc. of ACM SIGCOMM*.

[34] Shichang Xu, Subhabrata Sen, Z. Morley Mao, and Yunhan Jia. 2017. Dissecting VOD Services for Cellular: Performance, Root Causes and Best Practices. In *Proc. of ACM IMC*.

[35] X. Xu, J. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan. 2015. Investigating transparent web proxies in cellular networks. In *Proc. of PAM*.

[36] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. *ACM SIGCOMM CCR* (2015).

[37] Minlan Yu. 2019. Network telemetry: towards a top-down approach. *ACM SIGCOMM CCR* (2019).