A Unified Framework for Automatic Quality-of-Experience Optimization in Mobile Video Streaming

Yan Liu
Department of Information Engineering
The Chinese University of Hong Kong
Hong Kong
Email: liuyan@ie.cuhk.edu.hk

Jack Y. B. Lee
Department of Information Engineering
The Chinese University of Hong Kong
Hong Kong
Email: yblee@ie.cuhk.edu.hk

Abstract—Mobile video streaming is one of the fastest growing applications in the mobile Internet. Nevertheless, delivering high-quality streaming video over mobile networks remains a challenge. Researchers have since developed various novel streaming algorithms such as rate-adaptive streaming to improve the performance of mobile streaming services. However, selection or optimization of streaming algorithms is far from trivial and there is no systematic way to incorporate the tradeoffs between various performance metrics. This work aims at attacking the heart of the problem by developing a novel framework called Post Streaming Quality Analysis (PSQA) to automatically tune any streaming algorithms to maximize a given quality-of-experience (QoE) objective. We show that PSQA not only can be applied to optimize the performance of existing streaming algorithms, but also opens a new way for the exploration of new adaptive video streaming protocols and QoE metrics. Simulation results based on real network throughput traces show that PSQA can optimize existing and new streaming algorithms to achieve QoE that is remarkably close to the optimal achieved using brute-force method \textit{ex post facto}.

Keywords— Mobile Network, Video Streaming, Quality-of-Experience

I. INTRODUCTION

Without a doubt, mobile video streaming is one of the fastest growing applications in the mobile Internet. A report by Cisco [1] estimated that global mobile video streaming traffic will increase 13-fold from 2014 to 2019, accounting for 72% of all mobile data traffic by 2018. Nevertheless, today’s mobile video services are still far from perfect.

First, while modern mobile networks offer very high peak bandwidth (e.g., over 10Mbps in HSPA and even more in LTE), their wireless transmission nature unavoidably exhibits rapid and significant bandwidth fluctuations [2]. Second, with the recent widespread adoption of streaming video over HTTP/TCP (e.g., YouTube, Netlix, Hulu, etc.), the throughput of a video session is further modulated by TCP’s built-in flow and congestion control algorithms which also interact with mobile networks’ bandwidth fluctuations.

Given that the Internet, mobile networks included, does not yet offer any bandwidth guarantees, researchers have turned their focus to the development of adaptive video streaming systems where the video quality (i.e., bitrate) is adjusted according to the network condition to improve streaming quality [3]. In addition to research studies, commercial adaptive streaming solutions such as those from Apple [4], Microsoft [5], and Netflix [6] have also been widely deployed in the Internet.

Several recent studies [7] [8] revealed that these adaptive streaming solutions apparently have very different design tradeoffs. For example, Microsoft’s Smooth Streaming platform exhibited very conservative behavior in its video bitrate selection while the Netflix player implemented more aggressive rate-adaptation behavior which favored higher video quality even at the expense of more frequent bitrate switches. More generally, existing adaptive video streaming protocols are all designed to provide certain tradeoffs between \textit{picture quality} (e.g., video bitrate, frequency of bitrate switches, etc.) and \textit{streaming performance} (e.g., startup delay, playback buffering, etc.). However the tradeoffs are often only \textit{implicit} and generally \textit{non-quantifiable}. As the overall service quality experienced by the user – known as quality-of-experience (QoE), is affected by both quality metrics, it is difficult, if not impossible, to tune these protocols to maximize QoE.

As opposed to existing approaches which are mostly based on heuristics, this work aims to address this challenge by developing a unified framework called Post Streaming Quality Analysis (PSQA) which can incorporate the two sets of conflicting objectives to offer a \textit{practical}, \textit{self-adaptive} mechanism for maximizing the QoE of mobile video services. The proposed framework is rooted in three insights.

First, user-level QoE is determined by both picture quality and streaming performance. Moreover, we argue that their exact compositions are not unique but can vary across different users, types of service (free versus paid), or even contents (user-generated video versus movies). Thus the one-size-fits-all tradeoffs implemented in existing adaptive streaming protocols are far from optimal. Second, our investigations revealed that despite their bandwidth fluctuations, mobile networks do exhibit certain consistent long-term statistical properties which can be exploited for automatic tuning of streaming protocols. Third, PSQA exploits these statistical correlations by analyzing the optimal streaming parameters of \textit{past} streaming sessions with respect to a given QoE objective to enable it to automatically adapt the streaming protocol for \textit{future} streaming sessions. Simulation results based on production mobile network throughput traces suggested that PSQA can achieve mobile video services QoE that is remarkably close to the optimal achieved using brute-force method \textit{ex post facto}.
The remainder of the paper is organized as follows. Section II reviews the background and related works; Section III presents the PSQA framework; Section IV applies the framework to optimize five existing adaptive streaming algorithms; Section V applies the PSQA framework to explore the design of a generalized adaptive streaming algorithm; Section VI summarizes the study and outlines some future work.

II. BACKGROUND AND RELATED WORKS

In this section, we first review related works in mobile video streaming algorithms and then review studies in quantifying the quality-of-experience in video streaming.

A. Mobile Video Streaming Algorithms

The technologies behind mobile video streaming have undergone major transformations in the last decade. First, almost all mobile video services have migrated from the RTSP/RTP streaming protocols to HTTP-over-TCP streaming (also known as progressive download), due to the need to traverse firewalls/NATs often deployed in mobile networks. This creates the need for new video streaming protocols to cater for the different characteristics of the TCP transport (e.g., zero data loss but potentially long delay). The user experience has also changed substantially as corrupted video frames are now a thing of the past, and are replaced by playback rebuffering – temporary suspension of video playback to wait for video data to be delivered.

Second, it is now an accepted fact that mobile bandwidth is and will continue to be highly variable. Thus to provide good streaming quality it is common to encode a video into multiple bitrate versions so that the right version can be chosen to match the network condition at the time of streaming. This leads to intense research in the design of intelligent bitrate selection algorithms and adaptive streaming protocols [3]-[35].

A detailed review of the existing mobile streaming algorithms is beyond the scope of this work, interested reader may refer to a recent survey by Seufert et al. [9]. The design space for mobile streaming algorithms can be quite large. For example, there are various performance metrics in evaluating streaming algorithms, including video bitrate (e.g., [10] [11]), subjective video quality (e.g., [12] [13]), fairness (e.g., [10] [14]), stability (e.g., [10] [15] [16]), and efficiency (e.g., [10] [17]). System variables used for selecting or adapting the video bitrate include measured/estimated throughput (e.g., [18] [19] [20]), client buffer occupancy (e.g., [11] [20] [21]), recent or past traffic statistics (e.g., [22] [23]), current and predicted geographical location (e.g., [17] [24]), etc.

Not surprisingly, with many possible metrics and variables, researchers as well as the industry developed a wide range of streaming algorithms designed to achieve different sets of tradeoffs. Ultimately, the fundamental tradeoff remains the same – between video quality and streaming performance, both are essential factors to the user’s quality of experience.

B. Quality of Experience

It is easy to see that either high video quality (e.g., high bitrate) or high streaming performance (e.g., continuous playback without rebuffering) alone cannot provide a good quality of experience in streaming video. Moreover, video quality itself is not a single metric, but is composed of multiple factors such as:

- mean video bitrate – average bitrate of the video session (e.g., [20] [25] [26]);
- frequency of video quality switches (e.g., [13] [27] [28]); and
- magnitude of video quality switches (e.g., [13] [27] [28]).

Similarly, streaming performance is also influenced by multiple factors such as:

- startup delay (e.g., [25] [26] [27]) – waiting time for playback to commence;
- frequency of video rebuffering (e.g., [26] [27]); and
- duration of video rebuffering (e.g., [25] [27]).

Existing studies often employed different combinations of the above factors into quantifying the overall QoE for streaming video. In addition, even for the same factor different studies may adopt different approaches to incorporate its impact to the final QoE. For example, weighted sum [29], exponential/logarithmic [27] [30], threshold-based table look-up [26], decision tree [31], etc.

Finally, existing studies generally tune and verify their QoE metric using one of two approaches. In the first approach, e.g., Mok et al. [12] [26] and Liu et al. [27], researchers employed subjective experiments carried out by human subjects to determine the parameters/coefficients for their QoE functions. In the second approach, e.g., study by Dobrian et al. [25], researchers employed crowd-sourcing method to evaluate how QoE factors impact user engagement. Interested readers are referred to the tutorial by Chen et al. [32] for more in-depth discussions of video quality assessment methods.

It is clear that the relationship between different factors and user’s perceived QoE is highly complex. Conceivably, the desired QoE characterization also depends on video content, user preferences, subscription model, service provider preferences, and so on. Hence, it is likely that different services will require different QoE functions.

Existing streaming algorithms, however, are often not designed for different QoE functions and hence their actual QoE performance will vary across different types of streaming services. This work tackles this challenge by developing a new framework that enables one to relate an arbitrary streaming algorithm to an arbitrary QoE function, and then automatically optimize the streaming algorithm’s performance with respect to the given QoE function. This framework enables the optimization of existing streaming algorithms for different environments, and opens up a systematic way to explore the design of new adaptive streaming algorithms.

III. POST-STREAMING QUALITY ANALYSIS

The principle of PSQA is to exploit consistent statistical properties exhibited by mobile networks over a long timescale (e.g., days) to the automatic optimization of streaming algorithms. The PSQA framework does not mandate the form nor the parameters of the QoE function, nor the form or logic of the streaming algorithm. PSQA begins with an offline analysis
phase where captured throughput trace data are analyzed to determine the optimal parameters for a given streaming algorithm according to a given QoE function. This is done periodically (e.g., daily) to re-optimize the system parameters for use in the second phase – online prediction phase, where the actual streaming occurs based on the optimized parameters. The goal is to automatically tune the streaming algorithm based on past traffic data to improve performance according to the given QoE function.

A. Analysis Phase

The analysis phase makes use of past traffic trace data to tune the parameters of the target streaming algorithm according to a given QoE function. Specifically, the trace data can be TCP throughput trace data obtained as a by-product of past streaming sessions (e.g., via standard network trace capturing tools such as tcpdump [33]). These trace data will be used as inputs to a trace driven simulator/calculator where the actual streaming algorithm will be simulated to obtain its QoE performance.

As discussed earlier, the PSQA framework does not assume a particular form of QoE functions. Therefore we define a general function $U(A, P)$ to be the QoE function where $A$ and $P$ represent a vector of active parameters and passive variables respectively. Active parameters are parameters which can be controlled by the streaming algorithm, e.g., startup delay, video bitrate. By contrast, passive variables can be measured but not directly controlled, e.g., playback rebuffering frequency. Obviously the set of active parameters and passive variables depends on the chosen streaming algorithm and QoE function. We investigate existing ones in Section IV and explore a new one in Section V.

Once the QoE function is chosen, the service provider can then determine the objective function to optimize for the streaming service. For example, to maximize the average QoE experienced by all users, the service provider can employ the following objective function:

$$\max_{N} \frac{1}{N} \sum_{i=0}^{N-1} U(A_i, P_i)$$

where $N$ is the total number of streaming sessions in the past $M$ days. Other objective functions (e.g., max-min) are also possible and again the PSQA framework does not restrict its form.

In principle, given the QoE function, the objective function, and the streaming algorithm, one can then determine the optimal active parameters, i.e., $A$, such that the objective function is maximized for the past $N$ streaming sessions. However, we would still know nothing about how to choose the active parameters for future streaming sessions. Thus the real challenge is not in finding the exact values for active parameters – which can readily be done as the past throughput trace data are available, but in finding measurable system variables which have consistent statistical correlations with active parameters such that the active parameters for future streaming sessions can be predicted from them.

To formalize this relation we define an estimator function, denoted by $G(\cdot)$, which takes measurable system variables (e.g., throughput metrics) as inputs, and outputs a set of predicted active parameters for video session $i$:

$$A_i = G(\theta_{i,0}, \theta_{i,1}, ... \theta_{i,j-1}, k_{i,0}, k_{i,1}, ..., k_{i,M-1})$$

where $\theta_{i,j}$’s are $J$ input system variables that can be measured online and $k_{i,j}$’s are $M$ internal parameters of the specific adaptation algorithms to be optimized.

The estimator function offers the missing link to relate the set of system variables to the active parameters and the key is to find the set of optimized internal parameters $\{k_{i,0}^*, ..., k_{i,M-1}^*\}$ such that the objective function in (1) is maximized, i.e.,

$$\max_{k_{i,j}} \frac{1}{N} \sum_{i=0}^{N-1} U(G(\theta_{i,0}, \theta_{i,1}, ... \theta_{i,j-1}, k_{i,0}^*, k_{i,1}, ..., k_{i,M-1}^*), P)$$

This completes the offline analysis phase of PSQA. Note that the exact form of the estimator function is defined by the streaming algorithm employed. For example, a non-adaptive streaming algorithm which selects the video bitrate to be equal to a given ratio of the mean throughput of the previous streaming session (e.g., [22]) will have an estimator function with one input system variable – mean throughput of the previous session; one internal parameter – the bitrate-throughput ratio; and one active parameter – the video bitrate to be used for the new session. PSQA’s analysis phase will optimize the ratio to maximize the given objective function for the past $N$ sessions.

B. Prediction Phase

The second phase – prediction phase, is to make use of the optimized internal parameters $\{k_{i,0}^*, ..., k_{i,M-1}^*\}$ and apply the estimator function in (2) to automatically configure the active parameters for a new streaming session $\hat{x}$:

$$A_i = G(\theta_{i,0}, \theta_{i,1}, ... \theta_{i,j-1}, k_{i,0}^*, k_{i,1}, ..., k_{i,M-1}^*)$$

The intuition is that if the estimator function can capture the statistical behavior of the underlying mobile network, then the optimized internal parameter will likely result in good QoE for the new streaming session. Hence design of the estimator function is an interesting problem in its own right and the existing streaming algorithms implicitly implement their own versions. We employ the PSQA framework to evaluate and compare five existing streaming algorithms in the next section.

IV. APPLICATION TO EXISTING STREAMING ALGORITHMS

In this section, we apply the PSQA framework to five existing video streaming algorithms by Jiang et al. [10] (FESTIVE), Huang et al. [11] (BBA2), Liu et al. [18] (LBG), the Open Source Media Framework (OSMF) [34], and the Android Stagefright library [35] (Stagefright), using four existing QoE functions [20] [26] [29] [30]. We show how the PSQA framework can improve the performance of existing algorithms by re-optimizing their internal operating parameters.

A. Methodology

To evaluate streaming algorithms in realistic network settings we employ trace-driven simulations where the simulator implements the streaming algorithms and execute them with throughput trace data obtained from real-world production 3G/HSPA networks. Specifically, we collaborated with a mobile operator to setup a testbed with a server running Linux Apache httpd [36] in a wired network to deliver video data over HTTP/TCP. The client is a stationary notebook computer.
running Microsoft Windows 7, equipped with a 3G/HSPA USB modem for connecting to the Internet. We developed custom software to automatically run HTTP/TCP sessions mimicking HTTP-based video streaming sessions and captured the actual throughput trace data in parallel. It is worth noting that as the measurements were done in a production network, the trace data are likely to have incorporated the impacts of signal quality variation, interference, multiple competing users, and so on.

At the time of writing we collected three months’ trace data - about 20,000 streaming sessions of duration 300 seconds, each from three different locations. These trace data were then fed into the simulator which implemented the streaming algorithms to obtain various performance metrics. Each video was encoded into 8 bitrate versions: 200kbps, 400kbps, 600kbps, 1200kbps, 3500kbps, 5000kbps, 6500kbps, and 8500kbps [37].

Note that all five existing streaming algorithms have at least one configurable parameter which can impact their performance. We conducted three sets of experiments. In the first experiment we used parameters from the original studies unchanged. In the second experiment we applied PSQA to each algorithm to optimize one internal parameter (see Table I) which from our understanding has the biggest impact to performance.

The estimator functions for all five algorithms have the same output active parameter \( A_t \), which is the video bitrate to be selected for video segment \( x \). Each algorithm employed different input system variables, i.e., the \( \theta_j \)'s, and PSQA was used to optimize the multiplier to the internal parameter, i.e., \( \kappa_0 \), within a range as given in Table I. We configured PSQA to execute the analysis phase once a day, using previous 28 days’ trace data in the optimization process. The optimized \( \kappa_0 \) was then applied to the algorithm in the prediction phase in place of the value from the respective original studies.

The third experiment is designed to provide a benchmark for comparison. Specifically, we ran the optimization in (3) for the entire period, assuming all throughput trace data are known \( a \) priori, to obtain the optimal \( \kappa_0 \) for each algorithm and compute the resultant QoE performance. Clearly this after-the-fact (ATF) optimization cannot be realized in practice but it nonetheless offers a useful benchmark to evaluate the algorithms’ absolute performance.

### B. QoE Function

Clearly the choice of the QoE function will have a significant impact on the evaluation of a streaming algorithm’s performance. Therefore instead of adopting one particular QoE function, we employed four existing QoE functions from the literature [26,30,25,20], summarized below:

**QoE Function 1 (\( U_3 \))**: Developed by Mok et al. [26], this QoE function incorporates only factors from streaming performance and is defined as

\[
U = 4.23 - 0.0672L_{st} - 0.742L_p - 0.106L_{rb}
\]  

(5)

where \( L_{st}, L_p \) and \( L_{rb} \) are metrics for startup delay, frequency of rebuffering, and mean duration of rebuffering respectively, which are mapped according to Table III in [26].

**QoE Function 2 (\( U_2 \))**: Developed by Joskowicz and Ardao [30], their proposed QoE function is based only on video bitrate:

\[
U = 4.75 - 4.5e^{0.77R}
\]  

(6)

### Table I. Internal Parameter to be Optimized by PSQA.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>System Metric (( \theta_i ))</th>
<th>Internal Parameter (( \kappa_0 ))</th>
<th>Range of multiplier to ( \kappa_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBG</td>
<td>Video segment duration over segment fetch time</td>
<td>Multiplier to switch-up factor (( e^{-0.1} ))</td>
<td>(-1.0, 4.0)</td>
</tr>
<tr>
<td>OSMF</td>
<td>Video segment duration over segment fetch time</td>
<td>Multiplier to unnamed parameter (value = 1.0)</td>
<td>(0.2, 4.0)</td>
</tr>
<tr>
<td>Stagefright</td>
<td>Estimated bandwidth</td>
<td>Multiplier to unnamed parameters (values of 0.8 and 0.7)</td>
<td>(0.2, 4.0)</td>
</tr>
<tr>
<td>FESTIVE</td>
<td>Estimated bandwidth</td>
<td>Multiplier to parameter ( \rho ) in [10]</td>
<td>(0.2, 4.0)</td>
</tr>
<tr>
<td>BBA2</td>
<td>Current video buffer occupancy</td>
<td>Multiplier to maximum buffer size ( (B_{max}=240s) ) and reservoir ( r=96s )</td>
<td>(0.1, 4.0)</td>
</tr>
</tbody>
</table>

### Table II. Performance Under QoE Function \( U_3 \).

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Original PSQA Improvement by PSQA</th>
<th>ATF</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBG</td>
<td>3.12 3.25 4.2%</td>
<td>3.24</td>
<td>4.0</td>
</tr>
<tr>
<td>OSMF</td>
<td>3.12 3.26 4.5%</td>
<td>3.25</td>
<td>0.2</td>
</tr>
<tr>
<td>Stagefright</td>
<td>3.25 3.26 0.3%</td>
<td>3.25</td>
<td>0.2</td>
</tr>
<tr>
<td>FESTIVE</td>
<td>3.20 3.25 1.6%</td>
<td>3.25</td>
<td>0.2</td>
</tr>
<tr>
<td>BBA2</td>
<td>3.26 3.26 0.0%</td>
<td>3.26</td>
<td>0.8</td>
</tr>
</tbody>
</table>

### Table III. Performance Under QoE Function \( U_2 \).

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Original PSQA Improvement by PSQA</th>
<th>ATF</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBG</td>
<td>3.43 4.29 25.1%</td>
<td>4.29</td>
<td>-1.0</td>
</tr>
<tr>
<td>OSMF</td>
<td>3.30 4.38 32.7%</td>
<td>4.38</td>
<td>4.0</td>
</tr>
<tr>
<td>Stagefright</td>
<td>2.51 3.52 40.2%</td>
<td>3.52</td>
<td>4.0</td>
</tr>
<tr>
<td>FESTIVE</td>
<td>2.80 3.36 20.0%</td>
<td>3.36</td>
<td>4.0</td>
</tr>
<tr>
<td>BBA2</td>
<td>2.80 3.60 28.6%</td>
<td>3.60</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Table IV. Performance Under QoE Function \( U_1 \times 10^1 \).

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Original PSQA Improvement by PSQA</th>
<th>ATF</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBG</td>
<td>1.09 1.33 22.2%</td>
<td>1.35</td>
<td>-1.0</td>
</tr>
<tr>
<td>OSMF</td>
<td>1.00 1.19 19.0%</td>
<td>1.22</td>
<td>3.6</td>
</tr>
<tr>
<td>Stagefright</td>
<td>0.55 1.36 147.2%</td>
<td>1.36</td>
<td>2.2</td>
</tr>
<tr>
<td>FESTIVE</td>
<td>0.86 1.12 30.2%</td>
<td>1.11</td>
<td>1.6</td>
</tr>
<tr>
<td>BBA2</td>
<td>0.73 1.37 87.6%</td>
<td>1.36</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Table V. Performance Under QoE Function \( U_4 \times 10^1 \).

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Original PSQA Improvement by PSQA</th>
<th>ATF</th>
<th>( \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBG</td>
<td>3.12 3.26 4.5%</td>
<td>3.29</td>
<td>0.2</td>
</tr>
<tr>
<td>OSMF</td>
<td>2.18 2.35 7.8%</td>
<td>2.35</td>
<td>1.2</td>
</tr>
<tr>
<td>Stagefright</td>
<td>1.52 3.84 152.6%</td>
<td>3.83</td>
<td>2.0</td>
</tr>
<tr>
<td>FESTIVE</td>
<td>1.99 3.19 60.3%</td>
<td>3.19</td>
<td>1.6</td>
</tr>
<tr>
<td>BBA2</td>
<td>1.97 3.33 69.1%</td>
<td>3.33</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The original study did not consider adaptive video streaming where the video bitrate may vary for different video segments. Therefore we extend the QoE function in (6) to adaptive video streaming by averaging the QoE over all segments:

\[
U = \frac{d}{T} \sum_j U(T_j)
\]  

(7)
where $\bar{p}_j$ is the video bitrate selected for the $j^{th}$ segment, $T$ is video duration (e.g., $T=300s$), $\delta$ is the video segment duration (e.g., $\delta=2s$).

**QoE Function 3 ($U_3$):** Developed by Liu et al. [29], their proposed QoE function incorporates both video quality in terms of video bitrate and streaming performance in terms of buffer ratio:

$$U_3 = -3.7 \times \text{bitrate} + \frac{\text{buffer ratio}}{20}$$

where bitrate is the average video bitrate in kbps and buffer ratio is the ratio of total video rebuffering duration to total video session duration (i.e., video playback duration plus video rebuffering duration), in percentage.

**QoE Function 4 ($U_4$):** A recent study by Yin et al. [20] considered quality variations in adaptive video streaming with an even more comprehensive QoE function:

$$U_4 = \sum_{j=1}^{n} |p_j - \bar{p}_j| - 3000 \times T_p - 3000 \times T_s$$

where $T_p$ is the total video rebuffering duration and $T_s$ is the startup delay. The impact of video quality variations due to bitrate adaptations is captured by the second term in (9) which is the absolute bitrate differences between consecutive video segments.

C. Results

Table II-V summarize the mean QoE over all streaming sessions. Note that although we scaled the results of QoE functions $U_1$ (by $\times 10^3$) and $U_2$ (by $\times 10^5$) to ease reading/plotting it is not normalization as the numerical values from different QoE functions simply cannot be directly compared. The columns labeled “Original” are results from experiment one, i.e., streaming algorithms with internal parameters from their original studies. The columns labeled “ATF” are results obtained from experiment three, i.e., streaming algorithms optimized with complete a priori knowledge of trace data. The columns labeled “$\phi$” are results from experiment four, i.e., streaming algorithms optimized with complete a priori knowledge of trace data. A ratio of 0.2 means the ATF internal parameter is 0.2 times the original value of the internal parameter from its original study.

We first consider the streaming algorithms’ performance across the four QoE functions using their original parameters. QoE functions $U_1$ and $U_2$ are unique in that $U_1$ ($U_2$) considers only streaming performance (video quality) with no regard to video quality (streaming performance). As the two qualities are inherently a tradeoff, a streaming algorithm, e.g., StageFright, which achieved good performance in one quality, e.g., $U_1=3.25$, will exhibit lower performance in the other quality, e.g., $U_2=2.51$.

In contrast, QoE functions $U_3$ and $U_4$ incorporated the impacts of both video quality and streaming performance. The performance differences between algorithms widened considerably under these QoE. For instance, the ratio between the highest and the lowest QoE under $U_3$ and $U_4$ are 1.98 and 2.05 respectively while the same for $U_1$ and $U_2$ are only 1.04 and 1.37 respectively. These results demonstrate the importance of the QoE function in the design of a streaming algorithm and in almost all existing studies the former is not explicitly formulated.

Next we consider the PSQA-optimized results. There are two observations. First, PSQA was able to improve the performance of all streaming algorithms under all four QoE functions (except BBA under $U_1$ which is a tie). The improvements are often substantial compared to the original ones. Second, compare to the After-the-fact results (i.e., under ATF column), the PSQA-optimized results are remarkably close, i.e., comparable to having a priori knowledge of the trace data. Moreover this is consistent across all streaming algorithms and QoE functions thus suggesting the generality of the PSQA framework. In some cases the PSQA-optimized results even outperformed the After-the-fact results because PSQA optimizes with a sliding window of 1 day (and hence can adapt to variations over timescale of 1 day) while After-the-fact optimizes one internal parameter for the entire period of 49 days.

Finally, we observe that PSQA also narrowed the gap between different streaming algorithms. It significantly improved algorithms with lower performance by optimizing its internal parameter (c.f. Table I) according to the QoE function.

D. Practicality

PSQA does require additional processing and computation in the analysis phase and thus it is imperative that the requirements are within practical limits. In the above experiments the PSQA analysis phase was implemented in a python script and executed in a Linux machine with an Intel Core i7-3770 CPU running at 3.4GHz. The code utilized only a single CPU core and the analysis phase was completed in 891 seconds for LBG, 523 seconds for OSMF, 813 seconds for StageFright, 758 seconds for FESTIVE, and 771 seconds for BBA2. The differences are mainly due to different complexities in the logic of the streaming algorithms and the size of the parameter search space. As the analysis phase only needs to be executed periodically and in the previous experiments – only once a day, the computation time required to implement PSQA is well within practical limits.

V. A GENERALIZED ADAPTIVE STREAMING ALGORITHM

The PSQA framework opens up a new way to explore the design of streaming algorithms. Specifically, the performance of a particular streaming algorithm rests on three keys: the choice of adaptation logic, passive and internal parameters, and the chosen QoE metric. The challenge is that these are obviously dependent of one another, and so it could be difficult to isolate their exact performance impacts and draw more general observations.

PSQA offers a partial solution to this challenge by automatically optimizing the internal parameters for a given streaming algorithm under a given QoE metric. This can bring new insights into the relation between specific streaming logics and their impact to the final QoE as any anomalies due to improper internal parameter choices are minimized.

In this section we apply PSQA to explore the design of a generalized adaptive streaming (GS) algorithm that incorporates key QoE elements into its logic, with surprising results that raise some fundamental questions.
switch only to adjacent bitrate levels (one level higher or one level lower) while \( L = 2 \) means GS can switch to bitrates up to two levels away (two levels higher/lower).

Finally, a streaming algorithm ultimately relies on various system variables to inform its bitrate selection. GS employs two main variables for this purpose: throughput [10] [18] and client buffer occupancy [11] [21], both of which are widely used in existing adaptive streaming algorithms. The former can be measured online, e.g., the average TCP throughput in downloading the previous video segment, while the latter is known to the video player.

To relate the two parameters to the choice of video bitrate we introduce the third internal parameter called bitrate ratio, denoted by \( \gamma \), to control GS’s aggressiveness in choosing higher video bitrate. Let \( c_i \) be the average throughput in downloading the latest video segment, and \( d_i \) be the current client buffer occupancy measured in video playback duration at bitrate switching interval \( i \). Then the video bitrate to be selected for video segments in the next interval, denoted by \( r_i \), is computed from

\[
 r_i = \frac{\gamma c_i \cdot d_i + \tau}{\tau} \tag{10}
\]

The intuition behind (10) is that throughput and video data already received in the client buffer are both resources, albeit in different units (rate versus amount). Eq. (10) converts the latter into rate and then apply a discount factor \( \gamma \) to determine the video bitrate to be used for the next interval.

In practice there are a finite number of discrete video bitrate choices available so the computed video bitrate will be mapped to the closest bitrate level available:

\[
k_i = \arg \min_{k} | V_k - r_i | \tag{11}
\]

where \( V_k, k=0,1,\ldots,K-1, \) are the bitrate of video bitrate level \( k \).

To implement the maximum bitrate switching magnitude limit \( L \) the computed bitrate level is further limited by

\[
k_i = \begin{cases} 
  k_{i-1} - L & \text{if } k_i < k_{i-1} - L \\
  k_{i-1} + L & \text{if } k_i > k_{i-1} + L \\
  k_i & \text{otherwise} 
\end{cases} \tag{12}
\]

Therefore the estimator function for GS is in the form of

\[
k = G(c,d,\gamma,\tau,L) \tag{13}
\]

where the vectors \( k, c, \) and \( d \) are the selected bitrate levels, measured throughputs, and buffer occupancies respectively. We then apply PSQA to optimize the average QoE for all past \( N \) sessions:

\[
\max_{\gamma,\tau,L} \sum_{i=1}^{N} U(G(c,d,\gamma,\tau,L), P) \tag{14}
\]

where the passive variables \( P \) are selected according to the QoE function employed. The optimized internal parameters \( \{\gamma, \tau, L\} \) are then applied in the prediction phase to perform the actual bitrate adaptation online for new streaming sessions.
B. Performance Comparisons

Using the same throughput trace data as described in Section IV-A, we conducted trace-driven simulations for GS. We first compare the performance of GS to the five existing streaming algorithms in Table VI across the four QoE functions. The proposed GS algorithm performed remarkably well against the existing algorithms. In fact, except for $U_1$ where it is a tie, GS outperformed all five existing algorithms under $U_2$ to $U_4$.

Next we compare GS to the same five streaming algorithms but this time they were optimized by PSQA as described in Section IV. The results in Table VII again show that GS performed well compared to the PSQA-optimized existing algorithms, achieving comparable performance under $U_1$ and $U_3$, and best performance under $U_2$ and $U_4$. As all algorithms are PSQA-optimized, GS’s performance advantage is not due to PSQA, but an inherent property of its adaptation logic.

The above results are somewhat surprising given GS’s adaptation logic is far from complex. In fact it is as simple as it can be given the parameters and constraints. Moreover, the fact that GS performed consistently across all four QoE functions suggests that it is very moldable and can be tuned to a wide range of QoE functions via PSQA optimization.

In addition to optimizing streaming algorithms, the PSQA framework also offers a new way to gain deeper insights into the performance impact of individual system parameters. We demonstrate this by investigating the impact of individual internal parameters in the next two sections.

C. Trade-off between Video Quality and Streaming Performance

The first internal parameter – $\gamma$ controls GS’s aggressiveness in bitrate selection. While PSQA can optimize it automatically, we can also use it to investigate the parameter’s impact to the resultant QoE. To do this we ran a separate set of PSQA optimization for GS with a modified estimator function where only two internal parameters, namely $\tau$ and $L$, are optimized. The remaining one, i.e., $\gamma$ is fixed in each optimization run and we vary it from 0.0 to 4.0 in steps of 0.04 to obtain the resultant QoE curves in Fig. 1. For comparison, the markers represent the QoE performance when all three internal parameters, i.e., including $\gamma$ are PSQA optimized.

The idea is that if one varies one control parameter (e.g., $\gamma$) while holding the other parameters (e.g., $\tau$ and $L$) constant as is normally done, the results may not reflect the full picture of the control parameter’s impact if the three parameters are not independent. By contrast, the above procedure optimizes $\tau$ and $L$ for each value of $\gamma$ so that the latter’s impact can be more accurately characterized.

Returning to Fig. 1 we observe that, not surprisingly, the QoE performance differs significantly for different QoE functions $U_1$ to $U_4$. However, for $U_1$ and $U_2$ the QoE performance kept on increasing towards one of the extremes. For example, $U_1$ increases for smaller $\gamma$ all the way down to $\gamma=0$. Referring to the definition of $U_1$ in (5) we can see that it only considers streaming performance in calculating its QoE, with no regard to video bitrate at all. By reducing $\gamma$ to very small value the video bitrate will become very low (the average selected bitrate is 200 kbps, the lowest video bitrate level available) while streaming performance will become near perfect. Similarly, $U_2$ considers only video bitrate and not streaming performance, thus the performance kept on increasing for larger $\gamma$. However, at $\gamma=4.0$ the bitrate is already so high such that all streaming sessions suffered from playback rebuffering.

Obviously neither of the above is acceptable performance in practice. What is interesting is that the PSQA framework offers a way to detect such abnormality. A more complex example is QoE function $U_3$ which was designed to address the above limitation by incorporating both video bitrate and rebuffering ratio into the QoE calculation. Thus one would expect it to exhibit peak QoE performance somewhere between $\gamma$’s upper and lower limits. Interestingly it is not that simple and $U_3$ actually exhibits increasing QoE towards both extremes of $\gamma$. As $\gamma$ controls the tradeoff between video quality and streaming performance, this suggests that $U_3$ has two optimal operating points, i.e., very high video quality (e.g., mean bitrate of 7912 kbps) with low streaming performance (e.g., rebuffering ratio of 63.8%), or very good streaming performance (e.g., rebuffering ratio of 0.3%) with lower video quality (e.g., mean bitrate of 2880 kbps).
In contrast, QoE function $U_4$ exhibited a more consistent characteristics with peak QoE performance around $\gamma=0.5$, i.e., striking a balance between video quality and streaming performance. This example demonstrates the application of PSQA to analyzing the property of QoE metrics and their relation to a streaming algorithm’s parameters.

D. Impact of Bitrate Switching Frequency

A key tool in adaptive video streaming is the switching of video bitrate at video segment boundaries. Current streaming algorithms typically perform the decision for every video segment, with duration ranging from 2~10 seconds. Intuitively one would expect that if the streaming algorithm can switch more frequently then it will be better equipped to adapt to bandwidth fluctuations, potentially improving performance. We put this intuition to the test in this section.

Using similar procedure as in the previous section, we ran a separate set of PSQA optimization on GS but optimizing only $\gamma$ and $I$, for a given $\tau$, the latter of which controls the minimum time between bitrate switches. Fig. 2 plots the QoE performance for minimum bitrate switching period ranging from 2 to 300 seconds.

We observe that the variations across $\tau$ are far smaller than expected. Moreover, at $\tau=300s$, which is the same as the video duration, the GS algorithm may then appear to reduce to non-adaptive streaming as once the bitrate is chosen at the beginning of the video session it stays constant. This suggests that the gain due to more frequent bitrate adaptations may be far less than expected which is rather counter-intuitive given the success of adaptive streaming in practice.

The insight to this paradox is that the non-adaptive version of GS is not the same as the non-adaptive streaming tools currently in use in the Internet. Current non-adaptive streaming systems often do not select the video bitrate automatically for the user nor exploit knowledge of past throughput statistics to assist their bitrate selection. Hence the bitrate manually chosen by the user is understandably not very reliable, and could result in poor streaming performance. Adaptive streaming tools addressed this problem, often by beginning with a low-quality version and then adjusting the bitrates for subsequent video segments based on various measured system metrics. By contrast, even with $\tau=300s$ GS still exploits the knowledge of past network statistics through the use of PSQA in its bitrate selection.

Another interesting observation is that under $U_4$ the optimal minimum switching duration is not the shortest one but at $\tau=10s$. We conjecture that this is due to $U_4$’s use of bitrate differences in each switch to capture the impairment of bitrate variations, and too-short a switching duration may lead to wider bitrate changes due to the smaller averaging window.

Digging further into the results we group the results according to the video sessions’ mean throughput and plot their QoE performance versus throughput levels in Fig. 3. This set of results finally reveals the value of shorter bitrate switching intervals – at the lowest throughput level. Video sessions of the lowest throughput level are likely suffering from poor network conditions and hence one would expect the throughput to fluctuate much more significantly. The shorter bitrate switching interval thus allows the streaming algorithm to react more quickly to the rapidly changing throughput to reduce playback rebuffering. This result strongly suggests that there is room for further performance gains by optimizing a streaming algorithm according to different throughput levels.

VI. SUMMARY AND FUTURE WORK

The PSQA framework developed in this work introduces a new paradigm to the design of adaptive video streaming algorithms. For existing streaming algorithms we can apply PSQA to automatically optimize their internal parameters with respect to a given target QoE objective. In addition, we applied PSQA to explore the design of a simple yet intuitive adaptive streaming algorithm that performed well across a wide range of QoE functions.

There are three directions for future work. First, PSQA offers a new tool to systematically investigate and to uncover relations between the design of adaptation logic and its impact to various QoE metrics. This can potentially shed lights on the relative importance of various system parameters (e.g., throughput, buffer) and adaptation logic designs. Second, PSQA also opens up the way to automatic design of adaptation logic for streaming video. For example, by incorporating techniques from machine learning to generate adaptation logics from past throughput trace data. Such adaptive streaming algorithms will then not only adapt the video bitrate, but also adapt its own logic according to the evolution of the mobile network. Third, while this work focuses on mobile networks, the PSQA framework conceivably could also be applied to fixed networks thereby broadening its application to the global Internet.

ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their insightful comments and our anonymous mobile operator collaborator for contributing the trace data used in the simulations. This work is funded in part by a research grant (GRF14203714) from the HKSAR Research Grant Council and the Research Committee of CUHK.