Abstract—Current mobile video streaming is primarily delivered over HTTP. In addition to conventional web access logs, many service providers also log the data usage for billing and analytics purposes. This work exploits these vast amount of bandwidth trace data to enable the provisioning of streaming performance and video quality guarantees in adaptive mobile video streaming. We develop a post-streaming analysis framework to extract from past trace data statistical correlations between key system parameters that enables the automatic configuration of future streaming parameters including startup delay and rate adaptation such that a given minimum video quality and a target playback rebuffering probability can be maintained. The proposed framework is evaluated and compared to two existing adaptive streaming algorithms via a trace-driven simulator using extensive throughput trace data captured from a production 3G/HSPA network.

Keywords—Mobile, video, streaming, performance

I. INTRODUCTION

Mobile video streaming is quickly becoming the main application in the current mobile Internet. Beginning with user-generated contents and free contents, mobile video is following the footsteps of Internet video to offer more and more paid premium contents. This shift, however, creates new challenges for content providers and mobile operators.

In particular, today’s mobile users are reasonably tolerant to playback quality glitches in free contents and it is not uncommon to encounter low video quality and/or playback hiccups from time to time in mobile streaming. However, once users are paying for premium contents such as sports, TV shows, or movies, the expectation will likely to be very different and far more demanding.

First, it will be reasonable to assume that users will expect a minimum video quality level (e.g., similar to YouTube’s 720p level) throughout the video duration. In contrast, current adaptive streaming protocols such as Apple’s HLS [1] or the emerging MPEG-DASH [2] rely on the availability of a wide range of video bitrate versions available at the server so that the player can dynamically switch to a higher/lower bitrate version to adapt to bandwidth variations. As a result, during periods of low bandwidth it is likely that the player will drop the video bitrate to a level lower than what is expected for a paid premium service. While this is a valid design tradeoff for free contents the same cannot be assumed for paid premium services.

Second, current mobile streaming platforms were not designed to provide any streaming performance guarantees, and as such, the real-world streaming performance (e.g., in terms of playback hiccups or known as rebuffering) often vary substantially depending on network conditions. Again, this inconsistent streaming performance may be tolerable in free contents but will likely be unacceptable in paid premium services.

Although adaptive streaming has been studied extensively in recent years [3,4,5,6,7] and all of them achieved various degrees of improvements over non-adaptive streaming systems, few of the previous studies addressed the previously-mentioned challenges individually, let alone jointly. In this work we show that one can exploit the vast amount of throughput trace data that are often generated for billing/analytic purposes, for addressing the above-mentioned challenges. Specifically, we extend the post-streaming analysis framework developed by Liu and Lee [8] and Wu et al. [9] to adaptive mobile video streaming with a goal to offer configurable video quality guarantee and at the same time provide predictable streaming performance. We analyzed the proposed method’s performance in real-world mobile network scenarios via extensive trace-driven simulations, gaining insights into the tradeoffs between target video quality guarantee, streaming performance, bandwidth efficiency, and startup delay. The results show that despite the seemingly unpredictable bandwidth fluctuations in mobile networks, it is in fact feasible to provision paid premium adaptive streaming services with both video quality and streaming performance guarantees, paving the way for the deployment of high-quality premium mobile streaming services.

The rest of the paper is organized as follows: Section II reviews some previous related work and demonstrates their limitations; Section III presents the proposed Adaptive Post-Streaming Prefetch-Buffer Analysis (A-PSPBA) framework; Section IV evaluates its performance through extensive trace-driven simulations; and Section V concludes the study and outlines some future work.
TABLE I. VIDEO RESOLUTIONS AND BITRATES.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Bitrate (kbps)</th>
<th>144p</th>
<th>240p</th>
<th>360p</th>
<th>480p</th>
<th>720p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitrate</td>
<td></td>
<td>140</td>
<td>300</td>
<td>610</td>
<td>1100</td>
<td>2200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Bitrate (kbps)</th>
<th>1080p</th>
<th>1080p</th>
<th>1080p</th>
<th>1080p</th>
<th>1080p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitrate</td>
<td></td>
<td>4200</td>
<td>6000</td>
<td>8000</td>
<td>10000</td>
<td></td>
</tr>
</tbody>
</table>

TABLE II. PERFORMANCE COMPARISON OF ORIGINAL AND MODIFIED ANDROID HLS (A-HLS) STREAMING 2-HOUR VIDEOS.

<table>
<thead>
<tr>
<th></th>
<th>A-HLS (All bitrate versions)</th>
<th>Modified A-HLS (≥720p only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebuffering probability</td>
<td>0.3%</td>
<td>18.21%</td>
</tr>
<tr>
<td>Avg. bitrate (Mbps)</td>
<td>3.138</td>
<td>3.537</td>
</tr>
<tr>
<td>% of sessions with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>720p / 480p / 360p</td>
<td>89.7% / 68.1% / 44.9%</td>
<td>0% / 0% / 0%</td>
</tr>
<tr>
<td>% of duration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>720p / 480p / 360p</td>
<td>12.8% / 4.66% / 2.12%</td>
<td>0% / 0% / 0%</td>
</tr>
<tr>
<td>Bandwidth utilization</td>
<td>56.4%</td>
<td>65.0%</td>
</tr>
</tbody>
</table>

II. BACKGROUND AND RELATED WORK

Current mobile video streaming platforms primarily deliver video data over HTTP/TCP to take advantage of the latter’s infrastructure and compatibility. Much research has been done in HTTP-based streaming in recent years and the readers are referred to studies such as Akhshabi et al. [10] and Seuffert et al. [11] for comprehensive reviews and comparisons of existing streaming protocols.

A. Current Adaptive Video Streaming Protocols

This work focuses on adaptive video streaming over HTTP, of which Apple’s HLS is the most widely deployed in the mobile Internet. Even smartphones running the Android operating system are beginning to support HLS adaptive streaming due to its popularity. Nevertheless, as a number of studies [12,13] have shown, HLS is neither designed to provide video quality guarantee nor predictable streaming performance. To demonstrate this we simulated the adaptation algorithm as implemented in the Android OS (as of version 5.0) [14] – henceforth called A-HLS, using a trace-driven simulator. The simulator replicates HTTP/TCP throughput variations captured in a custom measurement platform operating in a production 3G/HSPA network (c.f. Section IV) and thus offers a realistic operating environment for evaluating the adaptive streaming protocol.

In the simulation the server is equipped with video qualities listed in Table I and the video is divided into fixed-duration segments of Δ=10 seconds each. We conducted simulations using 77-days’ of throughput trace data which is equivalent to 924 2-hour streaming sessions. For each streaming session we recorded the video bitrate selected and the number of playback rebuffering – suspension of playback due to player buffer underflow. The first two columns in Table II summarize A-HLS’s performance. We observe that A-HLS achieved relatively low rebuffering probability, e.g., 0.3%, which is consistent with its conservative bitrate selection algorithm. The tradeoff is in video quality. Its bandwidth utilization at 56.4% means that only slightly more than half the available bandwidth is used for delivering video data, with the rest unused.

More importantly, if this is a premium streaming service then the quality may be far from acceptable. For example, 89.7% of the streaming sessions encountered contains one or more video segments with quality below 720p, and 44.9% of them dropping to quality below even 360p which is clearly not acceptable. It is worth noting that the available bandwidth, i.e., 3.138Mbps/56.4%=5.56Mbps, is higher than the video bitrate at 720p at 2.2Mbps. In fact even A-HLS’s own average video bitrate at 3.138Mbps is higher than that of 720p and yet the majority of the streaming sessions suffered from low-quality video segments.

The above results clearly demonstrate a property of A-HLS which also applies to many other existing streaming protocols: it was not designed to offer video quality guarantee and thus will utilize whatever (low) video bitrates provided by the server to compensate for short-term network throughput variations, even if the network has the capacity for higher quality video in longer timescales.

It may appear that one can force the video quality to not lower than a target minimum simply by removing all those lower quality levels from the server, especially given that the average video bitrate achieved (3.138Mbps) by A-HLS is in fact higher than the bitrate of 720p (2.2Mbps). To test this idea we set the target minimum video quality to 720p by removing all lower bitrate versions (i.e., 144p to 480p) from the server and then rerun the simulation with the exact same set of trace data. The results are summarized in the third column of Table II.

For this simulation none of the streaming sessions’ video quality degrades to lower than 720p as the lower bitrate video versions are simply not available. However, this also means that when the network throughput drops, which is outside the control of the streaming protocol, the player may not be able to cope with it by switching to a lower bitrate version. This inevitably results in higher rebuffering probability which increased from 0.3% to over 18%. Given that video quality and streaming performance are both essential to paid premium streaming services this is clearly not acceptable either.

B. Non-Adaptive Streaming with Video Quality Guarantee

The above challenge is first investigated by Wu et al. [9] where the authors extended the statistical framework developed by Liu and Lee [8] for predictable streaming performance to incorporate video quality guarantee in constant-bitrate, i.e., non-adaptive streaming. The key insight is that in order to shield the streaming system from short-term throughput degradation which is the cause of video quality drops, without also running into playback rebuffering, one have to adjust the startup delay such that sufficient video data are prefetched before playback commences. These extra prefetched data then enables playback to continue at the required video quality during short-term network throughput degradations.

Nevertheless, non-adaptive streaming does suffer from one limitation – the lack of adaptation capability will negatively impact the streaming protocol’s ability to exploit available bandwidth. Intuitively, if the video bitrate cannot be changed...
then the rate decision will necessary be more conservative to cope with the potential bandwidth degradations, leading to lower bandwidth utilization. A streaming protocol may underutilize bandwidth if its bitrate choice is too low such that all video data are delivered completely before playback ends.

For example, using the same set of trace data we simulated the non-adaptive PSPBA [9] with a target rebuffering probability of 5% which achieved a bandwidth efficiency of only 58%. Clearly there is much room for improvement as only 58% of the available throughput was used for delivering video data, leaving the rest unused. This is where adaptive video streaming comes into the picture. By allowing the system to switch to a higher video bitrate during periods of upward throughput changes, adaptive streaming can potentially achieve substantially higher average video bitrate than non-adaptive streaming. The challenge is to do so while still guaranteeing both video quality and streaming performance. We present a new Adaptive-PSPBA framework in next section to address this challenge.

III. ADAPTIVE POST-STREAMING PREFETCH-BUFFER ANALYSIS

The principle of post-streaming analysis is to capture the vast amount of throughput trace data generated as a by-product of streaming and then analyze them to construct a statistical model for configuring the streaming parameters in future streaming sessions. In the following we extend the PSRA framework proposed by Liu and Lee [8] to provide both video quality and streaming performance guarantees in adaptive video streaming.

A. System Model

We first develop a system model to relate the following key system parameters: startup delay, video bitrate choice, network throughput, and a rate-adaptation parameter. Let \( d_i \) measured in seconds, be the startup delay selected for streaming session \( i \). This means the player will buffer video data for \( d_i \) seconds before commencing playback unless the whole video is downloaded completely before \( d_i \), in which case playback starts immediately after that. Note that the exact amount of video data that are buffered is not constant and depends on the actual network throughput during the startup period.

In contrast, conventional streaming protocols often buffer for a given amount (in bytes) or duration (in playback time) of video data before commencing playback, resulting in variable and unpredictable startup delay. We chose fixed startup delay as it can be displayed to the user at the beginning so that the user can know exactly when playback can begin. With a known waiting time we expect longer startup delay to be significantly more tolerable as the user can switch to other activities knowing when to come back to begin video playback.

Let \( T \) be the video duration and the video is divided into fixed-duration segments of \( \Delta \) seconds. Let \( C_i(t) \) be the throughput in session \( i \) at time \( t \) seconds after prefetching begins. In adaptive streaming the bitrate for future video segments will be adjusted dynamically based on network and player conditions. Existing players typically select the bitrate whenever a new segment is to be requested, i.e., periodically with intervals of \( \Delta \) seconds. In the proposed A-PSPBA framework we decouple the video quality switching period, denoted by \( \tau \) in seconds, from the video segment duration \( \Delta \). Practically video quality switching at video segment boundary is far easier to implement so we assume the quality switching period is an integer multiple of the segment duration, i.e., \( \tau = N \Delta \), where \( N \) is an integer.

The bitrate switching interval enables the service provider to further fine-tune the tradeoff between bitrate variations and bandwidth utilization. In particular, shorter bitrate switching interval allows closer tracking of the fluctuating throughput and hence could more efficiently utilize the available bandwidth. On the other hand, too frequent bitrate switches have also been shown to impair perceived video quality [15] and hence the parameter \( \tau \) allows a controlled tradeoff between these two conflicting factors.

In the previous work by Wu et al. [9] the initial video bitrate is determined based on the throughput of the previous streaming session. This can work well in a busy mobile cell with many streaming users but may suffer from lower throughput correlation in lightly utilized cells as the time from the previous streaming session may be significant, thereby reducing the throughput correlations between the two sessions.

To tackle this limitation we propose to eliminate the dependency on the previous streaming session and employ measurements only from the same streaming session. Specifically, the initial video quality is fixed to the target minimum video quality specified by the service provider, denoted by \( R_{\text{min}} \). This is then applied to the first \( H \) video segments such that video quality adaptation will begin from segment \( H+1 \). The parameter \( H \) controls the number of video segments used for in-band measurement of the average throughput which is a key input parameter to the adaptation algorithm. Increasing \( H \) allows more accurate throughput measurement at the expense of longer delay before adaptation begins (which could raise the video quality above the target minimum if throughput allows).

A central component to any adaptive streaming system is the rate adaptation algorithm, which often makes use of various throughput and client buffer statistics to adapt the video bitrate to prevent playback rebuffering. The proposed A-PSPBA framework does not completely dictate the design of the rate adaptation algorithm so it can be explored in conjunction with the framework. In the following we present an intuitive rate adaptation algorithm which incorporates intra-session throughput measurements and client buffer occupancy in its rate selection logic.

Specifically, given the bitrate switching interval \( \tau \), the bitrate to be used for video segment \( j \) of session \( i \) is determined from

\[
R_{i,j} = \begin{cases} 
R_{\text{min}}, & \text{for } j = 0,1,\ldots,(H-1) \\
\max \left\{ y \cdot \frac{C_i(u \cdot \Delta) - b_{i,u}}{\tau}, R_{\text{min}} \right\}, & \text{for } j \geq H 
\end{cases}
\]  

(1)

The first case in (1) represents the initial startup phase where the first \( H \) segments will be assigned the target
minimum bitrate $R_{\text{min}}$. In the second case in (1) the bitrate is computed from a product of $\gamma$ which is a parameter to control the tradeoff between video quality and streaming performance; $\bar{c}_{u,i}$ - which when $u=0$ is the average throughput in transferring the first $H$ segments and when $u>0$ is the average throughput of the previous switching interval $u-1$; and $(b_{i,u}+\tau)/\tau$ - which incorporates the impact of the client buffer occupancy at the beginning of switching interval $u$, denoted by $b_{i,u}$ measured in playback time (seconds). Larger client buffer occupancy will increase the selected video bitrate but the effect is amortized over the switching interval $\tau$. Finally if the computed bitrate is lower than the target minimum $R_{\text{min}}$ then the latter will be selected to guarantee video quality. Note that all video segments in the same bitrate switching interval are assigned the same video bitrate. Thus (1) only needs to be computed at the beginning of a new interval, i.e., once every $\tau$ seconds.

In practice, a service provider will likely encode only a fixed number of video bitrate versions for a given content and so the bitrate computed from (1) will need to be mapped to one of the available bitrate choices from

$$R_{i,j} = \arg\min_{r_{i,j}} [Q_i - r_{i,j}]$$

where $\{Q_i\}$ is the set of bitrates available and $R_{i,j}$ is the mapped bitrate for segment $j$ in session $i$. It is worth noting that the mapping function in (2) employs nearest-match rather than selecting an available bitrate not higher than $r_{i,j}$ to prevent introducing systematic errors into the system that will impair streaming performance predictability.

Given the bitrates selected for all video segments we can then determine the amount of video data consumed at any given time $t$ seconds after session begins from

$$B_i(t) = \begin{cases} 0, & \text{for } t < d_i \\ \sum_{j=0}^{(t-d_i)/\Delta} R_{i,j} \Delta, & \text{otherwise} \end{cases}$$

Similarly, the accumulated amount of video data received at time $t$ can be computed from the throughput trace function:

$$A_i(t) = \int_{0}^{t} C_i(t) dt$$

Hence playback rebuffering will occur when $A_i(t)<B_i(t)$. In the next section we present the post-streaming analysis framework to extract the statistical correlations between playback rebuffering and various control parameters.

B. Post-Streaming Analysis

We first illustrate the intuition behind post-streaming analysis using the system model developed in the previous section, and then present the framework to extract statistical correlations between key system parameters, namely startup delay $d_i$ and rate adaptation parameter $\gamma$.

**Fig. 1.** Preventing playback rebuffering with: (left-side figurea) more conservative bitrate selection (i.e., smaller $\gamma$) and (right-side figureb) with longer startup delay.

Fig. 1a plots the video data consumption curve $A_i(t)$ and consumption curve $B_i(t)$ versus time for a streaming session. We first consider the data consumption curve with $\gamma=\gamma_1$ which shows that it began playback after a startup delay of $d_i$ seconds and switched into a higher video bitrate at time $t_1$. At time $t_2$ the curve coincided with the data reception curve, indicating that it ran into playback rebuffering. Playback is suspended until time $t_3$ where it resumed playback till the end of the video.

In conventional adaptive streaming systems the primary tool to prevent playback rebuffering is to make the video bitrate selection more conservative. Indeed this can also be done here by means of adopting a smaller value for the rate control parameter $\gamma$ as illustrated by the green curve in Fig. 1a. We can observe that the lower video bitrate selected now prevented playback rebuffering from occurring under the exact same throughput conditions.

There is one problem though – the selected video bitrate may fall below the target minimum specified by the service provider and thus failing the goal to provide video quality guarantee. The key to resolving these two conflicting objectives is to adapt not only the rate control parameter $\gamma$ but also the startup delay $d_i$ as well.

This is further illustrated in Fig. 1b where the startup delay is now extended from $d_i$ to $d_2$. As a result the whole data consumption curve is shifted to the right and even though the rate control parameter $\gamma$ is unchanged at $\gamma_1$ we can still prevent playback rebuffering from occurring. The tradeoff is in longer startup delay and for this reason startup delay must be adapted in conjunction with the rate control parameter $\gamma$ to balance between startup delay and average video quality.

In the above analysis we were able to choose a set of operating parameters $\{d_2, \gamma_1\}$ to prevent any playback rebuffering from occurring. However there are two problems. First, we assumed that the data reception curve $A_i(t)$ is completely known a priori, i.e., before streaming commences, which is clearly not possible in practice. Second, the set of operating parameters is not necessary unique and thus we need a method to choose the best operating parameters among the feasible ones.

Both problems can be tackled by the post-streaming analysis framework. The principle is that mobile networks do exhibit consistent statistical properties across long timescales, e.g., over many days [8], and thus it is possible to derive
statistical properties from past throughput data to inform the selection of operating parameters for future streaming sessions. This was first proposed by Liu and Lee [8] for achieving predictable streaming performance and in the following we extend the framework to provide video quality guarantee in addition to predictable streaming performance.

First, the streaming server (often just a web server) records the actual HTTP/TCP throughput data for all streaming sessions. Many service providers are already recording these data for billing and analytic purposes so can easily be done as part of the logging process at the server. The trace data then capture the actual throughput behavior of the mobile network. This can be done on a per-mobile-cell basis to capture individual cell’s network characteristics. Using past throughput data we can then run the system model as described in Section III-A to derive the statistical relationships between the key system parameters for past streaming sessions.

Specifically, with the throughput trace data the data reception curve $A(t)$ is now known. For a given startup delay $d$ we can then determine the maximum $γ$ which does not result in playback rebuffering, i.e.,

$$γ_{\max} = \max \{ γ | A(t) ≥ B(t), 0 ≤ t ≤ T + d \} \tag{5}$$

Naturally the choice of startup delay $d$ affects the result and it is also possible that there is no feasible solution to (5), e.g., small $d$ and high target minimum bitrate. Therefore we compute (5) not just for one value of startup delay, but for a range of startup delay values in discrete steps.

Finally we repeat the above computation for a given set of past throughput data as if the streaming session were carried out in a back-to-back manner. Each streaming session generates a set of $\{γ_{\max}^{\prime} \}$ and together we can then generate the empirical cumulative distribution function (CDF) of $γ_{\max}^{\prime}$ for each discrete value of $d$, denoted by $F_d(\cdot)$. Its physical interpretation is the distribution of the maximum rate control parameter $γ$ that can be used without causing playback rebuffering at the given startup delay. Thus if the mobile network behavior is consistent in the future then we can make use of these empirical distributions to select the set of operating parameters for future streaming sessions.

### C. Operating Parameters Prediction

The adaptive streaming algorithm described in Section III-A is controlled by the operating parameters $\{d, γ, \}$, i.e., startup delay and rate control parameter. Post-streaming analysis generates the empirical distribution function for the maximum value of $γ$ without playback rebuffering under a range of discrete startup delays. In practice, aiming for zero playback rebuffering probability is likely to be too conservative and may result in excessive startup delay. Therefore the proposed framework offers a configurable parameter $α$ which is the target average rebuffering probability for future streaming sessions. A service provider can then control the tradeoff between startup delay, playback rebuffering probability, and average video bitrate.

Given a startup delay $d$ and a target rebuffering probability $α$, we can determine $γ_{\max}$ from its distribution function as follows:

$$γ_{\max} = F^{-1}_d(α) \tag{6}$$

where $F^{-1}_d(α)$ is the inverse distribution function which returns the value of $γ_{\max}$ that will result in rebuffering probability of $α$. If there is no feasible $γ_{\max}$ (e.g., rebuffering cannot be prevented even if $γ_{\max}=0$) then it will return a negative value. This can be done for all discrete values of startup delay $d$. Let $\{d | i=0,1,…,W-1\}$, be the set of $W$ discrete startup delays used in generating the distribution functions $F_i(\cdot)$. Then the set of operating parameters is obtained from

$$\{γ_{\max}^{\prime}, d_{\min}^{\prime}\} = \left\{ F^{-1}_d(α), \min_{i=0,…,W-1}\left\{ d | F^{-1}_d(α) ≥ 0 \right\} \right\} \tag{7}$$

Effectively we are selecting the minimum startup delay that offers a feasible $γ$ for the target rebuffering probability $α$. Note that there is an alternative selection method, i.e., selecting the maximum $γ$ among all startup delays. Intuitively this will increase average video bitrate at the expense of longer startup delay. We also tested this alternative version and found that the gain in video quality is insignificant but the resultant startup delay can be considerably longer.

### D. Discussions

The proposed framework can be deployed as two components. First, the post-streaming analysis module can be executed periodically with new throughput trace data to update the distribution functions $F_d(\cdot)$. In our experiments, executing it once a day offers a good balance between prediction accuracy and computation requirement.

Second, armed with the distribution functions, the rate adaptation module can then adapt the video bitrate for new streaming sessions as described in Section III-A. This can either be implemented at the server, e.g., by transparently switching between video segments of different encoding bitrates, or at the client as is commonly implemented in current adaptive streaming systems.

A further refinement of the framework is to divide the analysis and prediction processing into multiple throughput levels. The intuition is that the average throughput is correlated with the network condition, i.e., a low throughput is often caused by poor signal condition, and thus by segregating the throughput data into multiple throughput levels and then apply the framework to each throughput level separately we can potentially achieve higher prediction accuracy [8]. In our experiments described in the next section a total of 10 throughput levels were used.

### IV. Performance Evaluation

We evaluate the performance of the proposed A-PSPBA framework and compare it against the constant bitrate version PSPBA [9] and two existing adaptive streaming systems, namely A-HLS [14] and BBA [3]. A key challenge to comparative performance evaluation is the ability to recreate the exact same network environment for each of the algorithms under test so that the results are quantitatively comparable.
To this end we developed a trace-driven simulator where the mobile network link's throughput varies according to throughput trace data captured from a production 3G/HSPA mobile network. We developed a custom measurement platform to measure the actual TCP throughput available in mobile networks by means of sending data over TCP to a mobile receiver at the maximum rate allowed by TCP. The packet arrival times are then recorded into a trace file which is then fed into the simulator to replicate the exact TCP throughput variations as in the real network. At the time of writing we collected a total of 11 weeks of throughput trace data. The trace data have an overall average throughput of 5.73Mbps which suggests good streaming performance potential. However, it also varies substantially over short timescales, e.g., their 10s-averaged throughput can vary from 0.8kbps to 11.3Mbps.

It is worth noting that these throughput trace data are captured in a production mobile network and as such already incorporated the impact of various environmental (e.g., fading, interference, etc.), network (e.g., multiple competing users, etc.), and end-systems dynamics (e.g., TCP protocol behavior, operating system overheads, etc.). Thus the results, although simulated, do offer more fidelity to the algorithms’ performance in real mobile networks.

A. Experiment Setup

In all the experiments the video was encoded into 9 quality levels as listed in Table I. Two sets of video duration, namely 5 mins and 2 hrs were tested to evaluate performance for short and long videos. For the proposed A-PSPBA framework the post-streaming analysis was performed once a day using throughput trace data of the past 14 days. The generated distribution functions were then applied to the prediction of the operating parameters for the next day (i.e., 24 hrs) and the process is repeated for 63 days. We adopted 10 throughput levels: [0,1), [1,2), …, [10,∞) Mbps and a series of 400 startup delays for 5-min videos: {1, 2, …, 400} s. For 2-hr videos we employ a three-tiered division to reduce the number of discrete startup delays needed: {1,2,3,4, 5, …, 10 s}, {15,20, …, 1000 s}, and {1100, 1200, …, 16300 s}, for a total of 361 steps.

The service provider can control two system parameters: (a) target rebuffering probability \( \alpha \); and (b) bitrate switching interval \( \tau \). The primary performance metrics are average playback rebuffering probability, average startup delay, average video bitrate, and bandwidth utilization.

B. Performance Guarantees

The proposed A-PSPBA is designed to provide deterministic guarantee on video quality and statistical guarantees on streaming performance. The former guarantee is achieved by the rate adaptation algorithm in (1) where the selected video bitrate will not be lower than the target minimum \( R_{min} \).

Next we investigate the streaming performance in Fig. 3 by plotting the actual average video bitrate achieved by A-PSPBA versus the target minimum bitrate \( R_{min} \). We can observe that the actual average bitrate is significantly higher than \( R_{min} \). In fact the differences between different \( R_{min} \) are relatively small because A-PSPBA is inherently adaptive and is designed to utilize available bandwidth even if the target \( R_{min} \) is low. The slightly higher average bitrate for larger \( R_{min} \) is due to the higher initial bitrate (which equals to \( R_{min} \), c.f. Section III-A) and the constraint that the chosen bitrate cannot be lower than \( R_{min} \).

Next we investigate the streaming performance in Fig. 3 by plotting the actual rebuffering probability versus the target one, i.e., \( \alpha \), set by the service provider. The target \( R_{min} \) is 720p and two cases are tested, one for video duration of 5 mins and the other 2 hrs. We observe that the actual rebuffering probabilities tracked the target ones remarkably closely for the 5-min case. The 2-hr video case did exhibit more deviations from the target. We conjecture that the slightly larger deviations is due to the longer video duration, which weakened the correlation between the initial throughput measured during startup and the actual throughput in the latter part of the video session. As the initial measured throughput is used to determine the video session’s
throughput level, errors in its selection will likely result in corresponding increase in deviations in the actual rebuffering probability. Whether it is possible to reduce such error in long video sessions is an open problem and warrants further research.

C. Startup Delay

The key to A-PSPBA’s ability to control both video quality and streaming performance is its dynamic configuration of the startup delay. In A-PSPBA the startup delay is determined after the first $H$ segments are downloaded so the user can be informed of the exact time playback can commence. By contrast, existing streaming protocols often rely on downloading a certain amount of video data before commencing playback, resulting in unpredictable startup delay that is determined by the bandwidth available at the time. We argue that having a known and fixed startup delay can improve user experience and increase user’s tolerance of the delay.

We analyze A-PSPBA’s startup delay in Table III for four streaming configurations. In all four cases the target rebuffering probability is $\alpha=5\%$. The startup delays for 5-min videos were less than 10s which are similar to existing streaming systems in practice. The real challenge again is in long videos and 2-hr videos did require substantial startup delays – 222.5 and 666.2 seconds on average for target minimum quality of 480p and 720p respectively. While the average may appear to be large it is inflated by small number of outliers having very long startup delays, i.e., cases where the network throughput is low compared to the target minimum bitrate. In fact the 80-percentile startup delay is only 2 and 15 seconds for 480p and 720p respectively. This suggests that in actual service provisioning it may be desirable to handle the extreme cases separately, e.g., by suggesting the user to switch to a lower minimum video quality, to improve the user experience.

D. Bitrate Switching Interval

The proposed A-PSPBA streaming algorithm decouples bitrate switching interval from video segment duration and thus offers an additional tool for service provider to tradeoff between video quality variations (due to bitrate switches) and other performance metrics.

We first investigate the tradeoff in rebuffering probability predictability in Fig. 4 for bitrate switching intervals ranging from 10s (equal to the segment duration) to 7200s (equal to the video duration, i.e., no further adaptation once the first bitrate decision is performed). We can observe that the rebuffering probability was generally insensitive to the bitrate switching interval, even for period as large as 1800s. By contrast, the 7200-second case did deviate more significantly from the target, especially at higher target rebuffering probabilities. In fact in this case the bitrate is selected only once – at the beginning of the video session and is not changed afterwards, i.e., equivalent to non-adaptive streaming. This shows that rate adaptation is still essential to achieving consistent playback rebuffering performance in the presence of bandwidth fluctuations in the mobile network.

The low sensitivity to bitrate switching interval is somewhat surprising as it suggests that frequent bitrate switching may not bring any advantages. This is, however, not the whole picture. We plot in Fig. 5 the actual average bitrate for different bitrate switching intervals. This set of results clearly shows the real gain in shorter bitrate switching intervals – more frequent bitrate switches enables the system to achieve higher average bitrates. The difference did narrow rapidly for periods 300s or shorter and thus suggests that bitrate switching can in fact be done in periods far longer than is commonly done in conventional adaptive streaming systems (typically 10s or even shorter). Longer switching periods will reduce variations in the video quality which in itself is an important factor to the user’s perceived video quality [15].

E. Comparisons

To put A-PSPBA’s performance into perspective we further compare it to two existing algorithms, namely Android HLS [14] and the Buffer-based algorithm (BBA) [3]. Note that we extended the existing algorithms to provide minimum video quality guarantee as described in Section II-A. Table IV gives a comprehensive comparison of the four algorithms in streaming 2-hr video at a minimum quality guarantee of 720p.

First, both A-HLS and BBA exhibited very high levels of rebuffering probability which are unlikely to be acceptable in a
premium service. In fairness neither of these two algorithms were designed for video quality guarantee and thus the removal of the lower bitrate video versions significantly impaired their ability to cope with short-term throughput degradations.

For A-PSBBA and PSBBA the target rebuffering probability is the same at 5%. A-PSBBA was able to closely track the target rebuffering probability (at 5.26%) and yet was able to achieve an average video bitrate of 5Mbps, significantly outperforming the non-adaptive PSBBA at only 3.2Mbps. Moreover, the non-adaptive PSBBA algorithm achieved an actual rebuffering probability of 7.61% versus the target of 5%. The deviation is larger than the results reported in the earlier study [9] so we investigated further to isolate the cause. Turns out it is due to the wider bitrate selections at the high end – in this work three additional top-end bitrates {6Mbps, 8Mbps, 10Mbps} were provided compared to the previous study [9]. If we remove these additional bitrate choices, i.e., limit the maximum one to 4.2Mbps, then the resultant rebuffering probability became 5.78% which is very close to the 5% target. We conjecture that in non-adaptive PSBBA, video bitrate is only selected once at the beginning of the streaming session and hence a sub-optimal selection, e.g., selecting a bitrate higher than the actual throughput available, cannot be remedied later in the streaming session. In contrast, A-PSBBA can adapt the bitrate depending on throughput and buffer occupancy information and thus is able to compensate for errors in the initial bitrate selection.

V. CONCLUSION AND FUTURE WORK

This work exploited the vast amount of trace data generated in mobile video streaming to enable service providers to provision streaming video with deterministic quality guarantee and statistical streaming performance guarantee. The proposed A-PSBBA framework is very efficient in utilizing the available bandwidth to achieve high average bitrate and can operate with longer bitrate switching intervals, both factors contribute to perceived video quality positively.

For future work there are several open problems. First, for longer videos the actual streaming performance did deviate more significantly from the target. This is clearly undesirable and further work is warranted to investigate the root cause of this deviation. Second, the rate-adaptation algorithm employed in this work is only one of many possibilities and hence different algorithm designs, e.g., a less aggressive one, could achieve different tradeoff, e.g., reduce startup delay at the expense of lower bandwidth utilization. Finally, users may also perform interactive playback controls such as fast forward and rewind and hence an extended model is needed to account for such operations.

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 HK SAR Research Grant Council and the Research Committee of CUHK.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Adaptive PSPBA</th>
<th>PSBBA</th>
<th>PSPBA (max 4.2 Mbps)</th>
<th>A-HLS 720p</th>
<th>BBA 720p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual rebuffering probability</td>
<td>5.26%</td>
<td>7.61%</td>
<td>5.78%</td>
<td>18.21%</td>
<td>14.74%</td>
</tr>
<tr>
<td>Avg. bitrate (Mbps)</td>
<td>5.004</td>
<td>3.154</td>
<td>2.976</td>
<td>3.537</td>
<td>5.403</td>
</tr>
<tr>
<td>Bandwidth utilization</td>
<td>87.4%</td>
<td>58.3%</td>
<td>54.1%</td>
<td>65.0%</td>
<td>97.8%</td>
</tr>
</tbody>
</table>

REFERENCES