LAPAS: Latency-Aware Playback-Adaptive Streaming

Guanghui Zhang
Department of Information Engineering
The Chinese University of Hong Kong
Hong Kong
Email: ghzhang@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*—Live video streaming is one of the fastest growing applications in the mobile Internet. In addition to video quality and streaming performance, live video typically requires shorter user viewing latency or else the service quality will degrade significantly (e.g., live sports). This poses a significant challenge as latency and quality-of-experience (QoE) are inherently conflicting objectives. This work tackles this challenge by developing a novel LAPAS algorithm that employs playback rate adaptation in addition to bitrate adaptation to: (a) enable the service provider to explicitly control the tradeoff between latency and QoE; (b) reduce latency while maintaining reasonably high QoE. Extensive trace-driven simulation results showed that LAPAS can reduce latency by 21.8% to 75.1% at the same or higher QoE compared to leading algorithms. Moreover, it is the only algorithm that can achieve viewing latency shorter than 4 seconds and can work well even for latency as short as 2 seconds. Last but not least, LAPAS's performance is robust with respect to network locations, mobile operators, and time. It can be implemented entirely at the client side and is thus readily deployable in today's mobile Internet.**

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

I. INTRODUCTION

Mobile video streaming has grown tremendously in recent years and it is now one of the main applications in the mobile Internet [1]. In addition to video-on-demand, mobile live video streaming is rapidly gaining popularity over the last couple of years with live streaming services such as Periscope [2], YouTube Live [3], and Facebook Live [4].

Unlike on-demand streaming where the video is pre-encoded, in live streaming video capture, encoding, and streaming operate in tandem and in real-time. Moreover, in addition to video quality and streaming performance, live video typically requires shorter viewing latency or else the service quality will degrade significantly (e.g., live sports).

This last point presents a unique challenge as the primary means for streaming video systems to compensate for and absorb bandwidth fluctuations is by buffering up data at the video player so that playback can be sustained during sudden bandwidth droughts. This is why all streaming players prefetch a sizable amount of video data before commencing playback. However, buffered video data directly increases viewing latency in live streaming so the two performance metrics, i.e., QoE and latency, are inherently conflicting.

This work tackles the above-mentioned challenges by developing a novel Latency-Aware Playback-Adaptive Streaming (LAPAS) algorithm that: (a) enables the service provider to explicitly control the tradeoff between QoE and latency over a continuous spectrum; and (b) dynamically adapts not only the video bitrate, but also the playback rate to reduce latency while minimizing impact to the QoE.

Extensive trace-driven simulations based on TCP throughput trace data captured in production mobile networks show that LAPAS can reduce latency by 21.8% to 75.1% at the same or higher QoE compared to leading algorithms [5-8]. Moreover, the simulation results also revealed that the latencies incurred by existing streaming algorithms are substantial, ranging from 4.7 seconds to 8.5 seconds. By contrast, LAPAS supports tunable latency that can be tuned down to just 2 seconds while still maintaining reasonably high QoE.

The rest of paper is organized as follows: Section II reviews the background and related works; Section III evaluates existing adaptive streaming algorithms; Section IV presents the proposed LAPAS algorithm; Section V evaluates LAPAS's performance in live streaming; Section VI summarizes the study and outlines some future works.

II. BACKGROUND AND RELATED WORK

The majority of the works in the literature were focused on adaptive on-demand streaming, i.e., streaming videos which have been pre-encoded into multiple video bitrates. Some notable examples include the work by Yin *et al.* [8] (Robust-MPC), Spiteri *et al.* [9] (BOLA), Mao *et al.* [10] (Pensieve), and Akhtar *et al.* [11] (Oboe). These previous works, however, may not be directly applicable to live video streaming as they were not designed to reduce viewing latency.

Therefore researchers have developed separate algorithms specifically for live streaming services. A common technique is to control the client's buffer occupancy as the latter directly impacts viewing latency and QoE. For example, the studies by Wang *et al.* [5] and Cicco *et al.* [12] proposed algorithms employing PID feedback control to adapt the video bitrate to track a target buffer occupancy to maintain low latency while preventing playback rebuffering. Wang *et al.* [5] focused on RTMP [13] with the algorithm deployed at the server side while Cicco *et al.* [12] focused on DASH [14] with the algorithm deployed at the client. In another study, Xie *et al.* [6] proposed DTBB to determine video bitrate based on a dynamic buffer threshold adapted according to the estimated throughput.

In another work, Miller *et al.* [15] proposed LOLYPOP which employed video data skipping to reduce viewing latency. Zheng *et al.* [16] developed an online adaptive algorithm called OCTAD which employed Lyapunov optimization to assist service providers in live video transcoding decisions, bitrate assignment, and datacenter selection. For the video upload path, Siekkinen *et al.* [17] proposed an adaptive upload strategy for content providers to dynamically control bitrate levels and segments upload order to maximize the total video quality experienced by multiple users.

LAPAS differs from the existing works in two key aspects. First, while existing live streaming algorithms were designed with reducing viewing latency in mind, none of them offer any explicit control on the actual viewing latency. By contrast, LAPAS is the first algorithm that enables service providers to control the viewing latency explicitly.

Second, LAPAS is the first algorithm to integrate *playback rate adaptation* to reduce latency in live streaming. As opposed to skipping video data which will lead to visible quality degradation, changing the playback rate by a small fraction have been shown to be imperceivable to most users [18].

III. Adaptive Live Video Streaming Systems

In this section we first summarize the key components in a live video streaming system and then evaluate the performance of four leading adaptive live/on-demand streaming algorithms in the scenario of live video streaming.

A. Architecture

In a typical adaptive live streaming system, live video feed is first captured, encoded, and then uploaded to the service provider's receiving server where the video data will be further transcoded into multiple bitrate versions for the streaming server (or servers in a CDN).

Viewing latency, defined as the time difference between the current playback point and the live event, is a key performance metric in live streaming. In a live streaming system, latency will be incurred during capture, encoding, video segmentation, transcoding, streaming, as well as client buffering. In this work we focus on the download path where video data are requested and streamed to the client and so viewing latency refers to only the download path.

B. Comparison of Existing Streaming Algorithms

In this section we employ trace-driven simulation to evaluate and compare the performance of four existing live/on-demand adaptive streaming algorithms, namely PID [5] – a buffer-based adaptive live streaming algorithm based on RTMP, DTBB [6] – a buffer-based adaptive live streaming algorithm based on DASH, OSMF [7] – an open source commercial throughput-based algorithm developed by Adobe for both on-demand and live video streaming (supports both DASH and RTMP streams), and Robust-MPC [8] (henceforth called 'MPC') – a hybrid throughput-buffer-based algorithm originally designed for on-demand streaming based on DASH.

TABLE I. Simulation Setting

Streaming Parameters	Values
Bitrate profile	{0.2, 0.4, 0.8, 1.2, 2.2, 3.3, 5.0, 6.5, 8.6} Mbps
Segment duration	2 seconds
Frame rate	25 fps
Video duration	300 seconds
Session number	8,000
Buffer capacity	60 seconds

TABLE II. Comparison of Existing Streaming Algorithms

	Mean Latency (Seconds)	QoE[2] ($x10^3$)	Video Bitrate (Mbps)	Rebuffering Duration (Seconds)	Prefetch Duration (Seconds)
PID	4.7	2.0	2.1	1.6	4.0
DTBB	8.5	3.3	3.7	5.4	6.0
OSMF	6.0	2.7	3.7	4.6	4.0
MPC[1]	6.6	2.9	3.3	5.5	4.0
MPC-AR[1]	4.3	2.7	3.0	0.6	4.0

Notes: 1. MPC/MPC-AR were optimized using *Balanced*/*Avoid Rebuffering* QoE function. 2. The Balanced version of the QoE function is adopted to measure QoE performance for all five cases.

To offer insights into how streaming algorithms perform in real-world network environments we developed a platform to capture the achievable TCP throughput in production mobile networks. The captured throughput trace data incorporated all kinds of network impairments as well as resource contentions due to multiple users. At the time of writing our system has captured 4 months' worth of throughput trace data using a stationary client in 4 locations under 2 different mobile operators. The data are publicly available at [20].

As our focus is on the download path we assume the upload path is not the bottleneck. This will likely be the case for professionally captured and streamed contents such as large-scale live events. After the video data were uploaded they will be transcoded into multiple bitrate versions following Apple's recommended bitrate profile [21]. We applied CMAF [19] to DASH so that the streaming server can deliver video data on a frame-by-frame basis to minimize segmentation latency. We set the video framerate to 25 fps and segment duration to 2 seconds. The experiment simulated 30 days of streaming totaling 8,000 streaming sessions of duration 300 seconds each. The simulation settings are listed in Table I.

We measured five performance metrics:

- *mean latency* – defined as the average viewing latency (measured from the beginning of the video session) experienced in each video session;
- *QoE* – calculated from the QoE function proposed by Yin *et al.* [8], reproduced below:

$$Q=\frac{1}{K}\left(\sum_{k=1}^{K} r_k - \lambda\sum_{k=1}^{K-1}\left|r_{k+1}-r_k\right| - \mu Z\right) \quad (1)$$

  where Z is the total rebuffering duration experienced in a video session, r_k is the bitrate selected for segment k and K is the total number of segments. Note the QoE function can be configured into *Balanced* and *Avoid Rebuffering* versions via component weights (λ= 1, μ = 3000) and (λ= 1, μ = 6000) respectively;
- *video bitrate* – defined as the average bitrate selected;
- *rebuffering duration* – defined as the total rebuffering duration experienced in each session; and
- *prefetch duration* – the duration of video data buffered by the video player before playback commences.

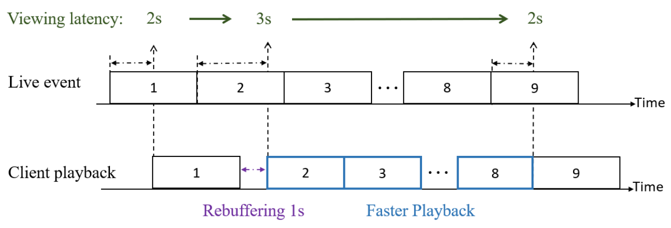

Fig. 1. Illustration of using adaptive playback to shorten viewing latency.

Table II summarizes the resultant average performance across all streaming sessions. Surprisingly, on-demand streaming algorithms do not necessary incur longer viewing latency than their live streaming counterparts. In fact both OSMF and MPC exhibited shorter viewing latency than DTBB. This is due to DTBB's choice of longer prefetch duration at 6 seconds. Nevertheless, PID – an algorithm specifically designed for live streaming, achieved shorter latency than OSMF/MPC. PID's shorter latency is due to its lower rebuffering duration (e.g., 1.6 seconds compared to MPC's 5.5 seconds).

We note that MPC can be optimized for any given QoE function and so far it was optimized and evaluated using the *Balanced* version of the QoE function in (1). This motivated us to optimize MPC using the *Avoid Rebuffering* version of the QoE function despite being evaluated using the *Balanced* QoE function. The results are shown in the last row of Table II (MPC-AR). Remarkably, MPC-AR not only achieved shorter latency than MPC, but also outperformed PID in both latency *and* QoE performance. This was achieved by reducing rebuffering duration from 5.5 seconds to 0.6 seconds, albeit with an unavoidable tradeoff in QoE (2.7×10^3 v.s. 2.9×10^3).

C. Discussions

There are three observations from the above results. First, viewing latency is primarily contributed by prefetch duration and rebuffering duration. This is intuitive as in both cases playback is suspended while the live event continues in real-time. Therefore to achieve short latency one needs a way to optimize these two factors against the conflicting QoE metric.

Second, rebuffering is an inevitable part of streaming. Whenever rebuffering occurs, the rebuffering duration will add to viewing latency. Worst-still, rebuffering duration *accumulates* so that the viewing latency can only *increase* with time. The above results were for video of 300 seconds. Longer live sessions will suffer even more from rebuffering.

Third, the latency performance of existing algorithms on one hand is an intrinsic property of the algorithms, and on the other hand is also affected by system parameters and network conditions. Therefore it is not possible to predict, let alone control, the latency performance for a live streaming session. Obviously this poses significant challenges to service providers as different live events demand different latency specifications and QoE preferences.

IV. Latency-Aware Playback-Adaptive Streaming

In this section we propose a novel Latency-Aware Playback-Adaptive Streaming (LAPAS) algorithm to tackle the three challenges discussed in Section III-C.

A. Adaptive Playback

A fundamental problem in live streaming is that once rebuffering occurs, the time *lost* in rebuffering will add to the viewing latency for the rest of the session. To tackle this problem we propose to adapt the *playback rate* in addition to adapting the video bitrate. The idea is to *increase* the video playback framerate slightly to enable playback to *catch-up* with the live event.

Fig. 1 illustrates the principle of playback adaptation. Assuming a latency threshold of 2 seconds and segment duration of 4 seconds the player ran into rebuffering for 1 second after segment 1, resulting in an increased viewing latency of 3 seconds. With adaptive playback the player then increased the playback framerate of segments 2 to 8 so that they took *shorter* time for playback, enabling the player to progressively shorten the latency back to 2 seconds.

The remaining question is how much one can change the playback rate without degrading the user experience. There are two aspects to this. First, changing video frame rate is relatively straightforward in video players as the latter were designed to support videos encoded with different framerates (e.g., common ones include 24fps, 25fps, 30fps, etc.).

According to Golubchik *et al.* [18] playback rate changes up to 5% are imperceivable to most users. We verified this by modifying the MPlayer [22] video player to support dynamic playback rate changes. Our own experiments with a wide range of playback rate changes agreed with [18]. In fact we found it difficult to spot even much higher playback rate changes but conservatively we adopted the 5% limit.

Second, most video also contains an audio stream multiplexed and synchronized to the video stream. Therefore changing the video playback rate alone will break audio-video synchronization, leading to unacceptable lip-sync issues. This can be addressed by resampling the audio to change its playback rate to match the video one. Moreover, one must also apply *pitch correction* to the resampled audio as some users are sensitive to pitch changes. As a proof-of-concept we successfully implemented audio resampling with pitch-correction together with video playback adaptation into MPlayer and verified their feasibility.

B. Adaptation Logics

LAPAS begins a streaming session by first prefetching ρ seconds of video data before commencing playback. Once playback starts the bitrate for the next segment to be requested is determined by its adaptation logics. Unlike existing algorithms, LAPAS implements two adaptation logics, namely, playback rate adaptation and bitrate adaptation.

The principle of LAPAS's playback rate adaptation is for the video player to continuously monitor the current actual latency and adapt towards a latency threshold, denoted by α. Let i be the session number and j be the next segment to be requested. As the adaptation logic executes whenever a new segment is requested, the following derivations are all executed at the time of requesting segment j in session i (hence the subscripts i, j). The *event time* – defined as the playback point when viewing latency is zero, is denoted by $\mu_{i,j}$, and can be calculated from the time elapsed since the beginning of the streaming session.

The current playback point, denoted by $\omega_{i,j}$, is known to the player so the viewing latency can be computed from

$$l_{i,j} = \mu_{i,j} - \omega_{i,j} \tag{2}$$

Let $g_{i,j}$ be the latency gap defined as

$$g_{i,j} = l_{i,j} - \alpha \tag{3}$$

where α is the *latency threshold*. The goal of playback adaptation is to adjust the playback framerate to reduce the latency gap to zero. We define a *playback rate multiplier*, denoted by $\theta_{i,j}$, to control the playback rate change, with $\theta_{i,j}=1$ for normal playback rate; $\theta_{i,j}>1$ for speedup; and $\theta_{i,j}<1$ for slowdown. We consider three cases below:

Case 1: $g_{i,j}=0$, i.e., latency threshold reached. In this case normal playback rate will be maintained, i.e., $\theta_{i,j}=1$.

Case 2: $g_{i,j}>0$, i.e., current viewing latency is longer than the target. In this case video playback will be sped-up by a factor of

$$\theta_{i,j}=1+\min\left\{g_{i,j}/\beta,\kappa_{\max}\right\},\ \text{if } g_{i,j}>0 \tag{4}$$

where $\kappa_{\max}$ is the maximum playback rate change allowed and β is a parameter to control the aggressiveness of playback rate adaptation. Intuitively, β is the time it takes to catch-up with the latency threshold, assuming it is not limited by $\kappa_{\max}$.

Case 3: $g_{i,j}<0$, i.e., current viewing latency is shorter than the target. In this case video playback will be slowed-down by a factor of

$$\theta_{i,j}=1+\max\left\{g_{i,j}/\beta,-\kappa_{\max}\right\},\ \text{if } g_{i,j}<0 \tag{5}$$

This serves two purposes: (a) to maintain a consistent latency throughout the streaming session; and (b) to exploit the shorter latency to improve video quality (see below).

For bitrate adaptation, LAPAS implements a hybrid throughput-buffer-based logic where measured throughput and client buffer occupancy are used in conjunction with playback rate change to determine the video bitrate. Let $c_{i,j}$ be the measured throughput in downloading the previous W segments, i.e., segment $j-(W+1)$ to $j-1$ and $d_{i,j}$ be the current buffer occupancy - the duration of video data (measured in playback time) buffered at the client at the time of requesting segment j. The bitrate to be selected for requesting segment j, denoted by $b_{i,j}$, is then computed from

$$b_{i,j} = \gamma c_{i,j}\frac{d_{i,j}+\left(1-\theta_{i,j}\right)\times\beta}{\tau} \tag{6}$$

where τ is the segment duration and γ is a *discount factor* for tuning the bitrate selection aggressiveness of the adaption logic. Intuitively, if $\gamma=1$ and the future throughput is exactly equal to $c_{i,j}$ then all buffered video data will be consumed by the time segment j is completely downloaded, i.e.,

$$\frac{b_{i,j}\tau}{c_{i,j}} \equiv d_{i,j}+\left(1-\theta_{i,j}\right)\times\beta \tag{7}$$

thereby maximizing video quality without causing rebuffering.

Clearly, there will be estimation errors in predicting future throughput based on past measurements so by tuning the discount factor γ one can optimize the algorithm according to the network's characteristics (c.f. Section IV-C).

TABLE III. OPERATING PARAMETERS AND TUNING RANGE IN LAPAS

Operating Parameters	Prefetch Duration ρ (Seconds)	Latency Threshold α (Seconds)	Playback Rate Control Parameter β (Seconds)	Discount Factor γ
Tuning Range	0.04 ~ 12	0.1 ~ 9.0	0.2 ~ 2.0	0.1 ~ 2.0

Finally, in practice there are a finite number of discrete video bitrate choices available so the computed video bitrate $b_{i,j}$ will be mapped to the closest bitrate level available in the encoding bitrate profile:

$$h=\arg\min_{h}\left|\tau_h - b_{i,j}\right| \tag{8}$$

where τ_h, $h=0,1,\ldots,H-1$ are the encoding bitrate of video bitrate level h.

C. Data-Driven Parametric Optimization

There are four operating parameters in LAPAS, namely prefetch duration ρ, latency threshold α, playback rate control parameter β and discount factor γ (c.f. Table III). One approach is to tune parameters via simulation and/or experiments. However, our investigations revealed that the optimal set of parameters inevitably change over time due to changing network conditions. To tackle this challenge we employed the Post-Streaming Quality Analysis (PSQA) framework by Liu and Lee [23] to automatically tune LAPAS's parameters through analyzing the throughput trace data of past streaming sessions.

PSQA comprises two phases: an offline analysis phase and an online prediction phase. The offline phase executes periodically (e.g., daily) to analyze past throughput trace data, captured as a by-product of actual streaming sessions, to optimize the operating parameters according to a given objective function. The optimized parameters will then be applied to new streaming sessions in the online phase until they are updated again in the next offline phase. Due to space limitation we refer the readers to Liu and Lee [23] for details of the PSQA framework.

We adopted an objective function to maximize the average QoE of all streaming sessions in the past N days (e.g., $N=7$), subject to a mean viewing latency constraint λ set by the service provider, i.e.,

$$\begin{aligned}&\max_{\rho,\alpha,\beta,\gamma} \mathrm{E}\left[Q_i(\rho,\alpha,\beta,\gamma)\,|\,\forall i\right]\\ &\text{s.t. } \mathrm{E}\left[L_i(\rho,\alpha,\beta,\gamma)\,|\,\forall i\right]\le\lambda\end{aligned} \tag{9}$$

where $Q_i(\cdot)$ and $L_i(\cdot)$ denote the achieved QoE and mean session latency for past streaming session i. The service provider through setting the desired viewing latency constraint λ can then control the tradeoff between viewing latency and QoE.

V. PERFORMANCE EVALUATION

In this section we employ trace-driven simulation to evaluate and compare the performance of LAPAS against four existing algorithms, namely PID [5], DTBB [6], OSMF [7], and MPC [8].

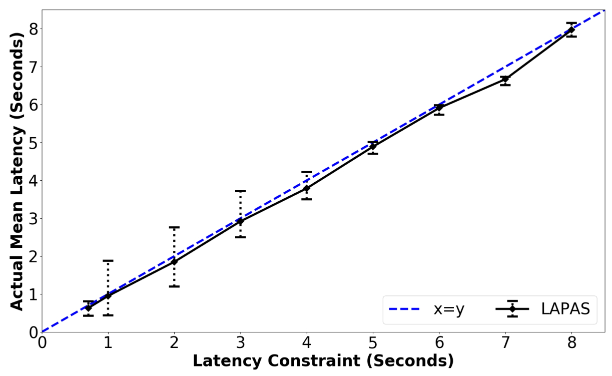

Fig. 2. The actual mean latency achieved by LAPAS versus the latency constraint set by the service provider (error bars span top/bottom 5%-tile).

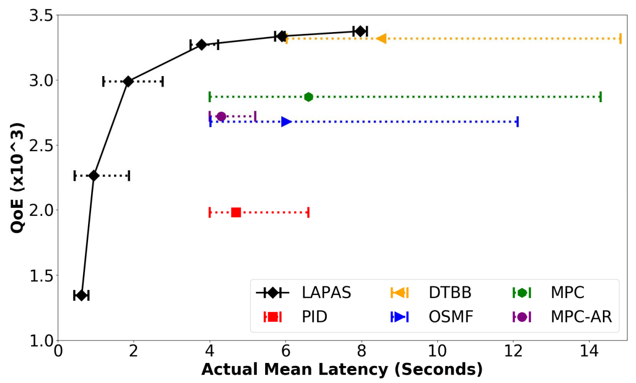

Fig. 3. Comparison of latency-QoE tradeoff for all algorithms (error bars span top/bottom 5%-tile).

A. Simulation Setup

We employed the same simulation setup as described in Section III. Performance results were obtained from simulating a total of 30 days with a total of 8,000 streaming sessions. LAPAS's offline analysis phase was configured to analyze the past 7 days' throughput trace data to obtain optimized parameters, namely prefetch duration ρ^*, latency threshold α^*, playback rate control parameter β^* and discount factor γ^* for use in the next 24 hours after which the process repeats. The choice of 7 days was based on experimental results and the intuition that it fully covers the day-of-the-week variations. The tuning range of the four operating parameters is listed in Table III. The maximum playback rate change is limited to κ_{max}=5%.

The comparison algorithms were configured according to their original studies except for MPC-AR which was optimized using the Avoid Rebuffering QoE function (c.f. Section III-B). All algorithms were evaluated using the Balanced QoE function in (1).

B. Latency-QoE Tradeoff

LAPAS is unique in that it offers the service provider direct control of the desired viewing latency. This is demonstrated in Fig. 2 which plots the actual mean latency achieved in all streaming sessions against the latency constraint λ set by the service provider. We can observe that the former closely tracked the latter.

TABLE IV. CHARACTERISTICS OF THROUGHPUT TRACE DATASETS

Dataset	#1	#2	#3	#4	Overall
Mean Throughput (Mbps)	7.2	5.5	2.6	1.6	4.2
Coefficient of Variation	0.44	0.39	0.74	0.53	0.53
Location	L1	L1	L2	L3	n/a
Mobile Operator	M1	M2	M1	M1	n/a

TABLE V. ACTUAL MEAN LATENCY (SECONDS) ACROSS FOUR DATASETS

Datasets	#1	#2	#3	#4
LAPAS	2.9	2.8	2.9	2.7
PID	4.7	4.9	4.8	4.6
DTBB	8.5	7.8	8.7	9.1
OSMF	6.0	5.7	5.9	7.4
MPC	6.6	10.4	4.6	4.2
MPC-AR	4.3	4.3	4.6	4.2

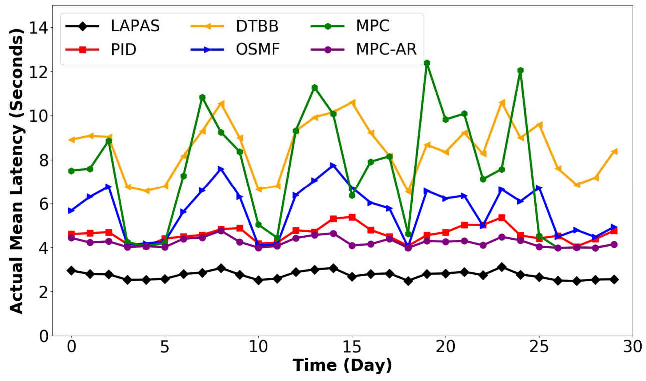

Fig. 4. Variation of daily mean latency over time.

Clearly reducing viewing latency will tradeoff QoE and this tradeoff is summarized in Fig. 3. Existing algorithms each achieved a different set of latency-QoE tradeoff which is not controllable. By contrast, LAPAS supports a controllable, continuous tradeoff between latency and QoE. Moreover, LAPAS's latency varied substantially less than the others (except MPC-AR). At higher latencies (e.g., 4 seconds or more) LAPAS outperformed the existing algorithms by achieving substantially higher QoE. At the other extreme LAPAS is the only algorithm that can achieve latency shorter than 4 seconds. LAPAS can reduce latency by 21.8% to 75.1% at the same or higher QoE compared to existing algorithms. More remarkably, LAPAS can reduce the latency to a mere 1.85 seconds while still outperforming all but DTBB (which has >8 seconds latency) in QoE performance.

C. Robustness

In this section we investigate the robustness of the streaming algorithms. Specifically, we first consider spatial robustness – latency variations over different network characteristics, e.g., geographical locations and mobile operators, which presumably may exhibit different network conditions. We conducted three additional simulations using three more datasets. Table IV summarized the key statistics for all four datasets. The resultant average latency across all streaming sessions are summarized in Table V.

We observe that PID and LAPAS (with λ= 3 seconds) maintained reasonably consistent latency levels across the four datasets. In contrast, DTBB, OSMF and MPC exhibited far greater variations due to the differing network conditions such as mean throughput and throughput variations (c.f. Table IV).

Interestingly, MPC-AR also achieved consistent latencies across the four datasets. This is because MPC-AR is more sensitive to rebuffering due to the use of the Avoid Rebuffering QoE function and with reduced rebuffering the latency is dominated by the fixed prefetch duration.

Next we consider temporal robustness – latency variations over time (i.e., days). Fig. 4 plots the daily mean latency achieved by each algorithm in dataset #1 from testing day 0 to day 30. Again, LAPAS, MPC-AR, and PID exhibited good temporal robustness while DTBB and OSMF varied more substantially across different days. MPC exhibited the highest temporal variations ranging from a low of 4.1 seconds to a high of 12.3 seconds.

Overall the results demonstrated that LAPAS not only outperformed existing algorithms in latency/QoE, but also achieved significantly higher robustness with respect to geographical locations, mobile operators, and time.

VI. Summary and Future Work

This is the first work employing playback rate adaptation to adaptive live video streaming. What is remarkable is that a mere imperceivable 5% change in playback rate can already improve latency performance significantly. The proposed LAPAS algorithm not only enables the service provider to control the viewing latency, it does so by achieving higher QoE than existing algorithms at the same or lower latency. For latency-sensitive contents LAPAS is the only algorithm that can achieve latency below 4 seconds and even all the way down to a mere 2 seconds while still maintaining reasonably good QoE performance. LAPAS can be implemented entirely at the client side, is compatible with the existing DASH/CMAF/HTTP infrastructure, and thus offer a practical solution to latency-sensitive live streaming services.

There are at least three directions for future research. First, LAPAS currently optimizes QoE subject to a given latency constraint. As latency is inherently part of the user experience it will be useful if latency can be unified and integrated into the existing QoE functions. Second, the principle of playback rate change is not limited to live video streaming but could also improve the performance of on-demand streaming. Finally, more work is warranted to explore more sophisticated ways to adapt the playback rate. Instead of applying a fixed playback rate for a period of time one could progressively ramp-up or down the playback rate to make it even less noticeable and thus would allow higher playback rate changes to be applied.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments and suggestions. This work was funded in part by a research grant (GRF14203714) from the HKSAR Research Grant Council.

References

[1] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecase Update, 2016-2021*, Mar 2017, Cisco Inc. [Online] http://www.cisco.com/c/en/us/solutions/collateral/service-provider /visual-networking-index-vni/mobile-white-paper-c11-520862.html

[2] *Periscope* [Online] https://play.google.com/store/apps/details?id=tv-.periscope.android&hl=en_US

[3] *Youtube Live* [Online] https://www.youtube.com/channel/UC4R8D-WoMoI7CAwX8_LjQHig

[4] *Facebook Live* [Online] https://live.fb.com/

[5] J. Wang, S. Meng, J. Sun, Z. Quo, "A general PID-based rate adaptation approach for TCP-based live streaming over mobile networks," *Proc. IEEE ICME*, Seattle, USA, Jul 2016, pp.1-6.

[6] L. Xie, C. Zhou, X. Zhang, "Dynamic threshold based rate adaptation for HTTP live streaming," *Proc. IEEE ISCAS*, Baltimore, MD, USA, May 2017, pp.1-4.

[7] *Adobe OSMF* [Online] https://sourceforge.net/projects/osmf.adobe/

[8] X. Yin, A. Jindal, V. Sekar and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," *Proc. ACM SIGCOMM*, London, United Kingdom, Aug 2015, pp.325-338.

[9] K. Spiteri, R. Urgaonkar and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," *Proc. IEEE INFOCOM*, San Francisco, CA, USA, Apr 2016, pp.1-9.

[10] H. Mao, R. Netravali, M. Alizadeh, "Neural adaptive video streaming with pensieve," *Proc. ACM SIGCOMM*, Los Angeles, CA, USA, Aug 2017, pp.197-210.

[11] Z. Akhtar, Y. S. Nam, R. Govindan, "Oboe: auto-tuning video ABR algorithms to network conditions" *Proc. ACM SIGCOMM*, Budapest, Hungary, Aug 2018, pp.44-58.

[12] L. De Cicco, S. Mascolo, V. Palmisano, "Feedback control for adaptive live video streaming," *Proc. ACM Multimedia Syst.*, San Jose, USA, Feb 2011, pp.145-156.

[13] *Adobe RTMP* [Online] https://www.adobe.com/devnet/rtmp.html

[14] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," *Proc. ACM Multimedia Syst.*, San Jose, USA, Feb 2011, pp.133-144.

[15] K. Miller, A. K. Al-Tamimi, A. Wolisz, "QoE-Based low-delay live streaming using throughput predictions," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13(1), Jan 2017, pp.1-24.

[16] Y. Zheng, D. Wu, Y. Ke, "Online cloud transcoding and distribution for crowdsourced live game video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27(8), Apr 2016, pp. 1777-1789.

[17] M. Siekkinen, E. Masala, J. K. Nurminen, "Optimized upload strategies for live scalable video transmission from mobile devices," *IEEE Transactions on mobile computing*, vol. 16(4), Apr 2017, pp.1059-1072.

[18] L. Golubchik, John C. S. Lui and R. R. Muntz, "Reducing I/O demands in Video-on-Demand Storage Servers," *ACM SIGMETRICS*, Ottawa, Canada, May 1995, pp.25-36.

[19] *Common Media Application Format (CMAF)* [Online] https://mpe-g.chiariglione.org/standards/mpeg-a/common-media-application-format

[20] *Mobile Throughput Trace Data.* [Online] http://sonar.mclab.info/-tracedata/TCP/3G/

[21] *Best Practices for Creating and Deploying HTTP Live Streaming Media for the iPhone and iPad,* Apple Inc, retrieved on Aug 2016 [Online] https://developer.apple.com/library/ios/technotes/tn2224/_index.html

[22] *MPlayer* [Online] https://mplayerhq.hu

[23] Y. Liu and J.Y.B. Lee. "A Unified Framework for Automatic Quality-of-Experience Optimization in Mobile Video Streaming," *Proc. IEEE INFOCOM*, San Francisco, CA, USA., Apr 2016, pp.1-9.