applied sciences

MDPI

Article

Bit Rate Adaptation Using Linear Quadratic Optimization for Mobile Video Streaming

Young-myoung Kang [1] and Yeon-sup Lim [2,*]

[1] Network Business, Samsung Electronics, Suwon 16677, Korea; ym10.kang@samsung.com
[2] Department of Convergence Security Engineering, Sungshin Women's University, Seoul 02844, Korea
* Correspondence: ylim@sungshin.ac.kr; Tel.: +82-2-920-7144

Abstract: Video streaming application such as Youtube is one of the most popular mobile applications. To adjust the quality of video for available network bandwidth, a streaming server provides multiple representations of video of which bit rate has different bandwidth requirements. A streaming client utilizes an adaptive bit rate (ABR) scheme to select a proper representation that the network can support. However, in mobile environments, incorrect decisions of an ABR scheme often cause playback stalls that significantly degrade the quality of user experience, which can easily happen due to network dynamics. In this work, we propose a control theory (Linear Quadratic Optimization)-based ABR scheme to enhance the quality of experience in mobile video streaming. Our simulation study shows that our proposed ABR scheme successfully mitigates and shortens playback stalls while preserving the similar quality of streaming video compared to the state-of-the-art ABR schemes.

Keywords: dynamic adaptive streaming over HTTP; adaptive bit rate scheme; control theory

Citation: Kang, Y.-m.; Lim, Y.-s. Bit Rate Adaptation Using Linear Quadratic Optimization for Mobile Video Streaming. *Appl. Sci.* **2021**, *11*, 99. https://dx.doi.org/10.3390/app11010099

Received: 16 November 2020
Accepted: 21 December 2020
Published: 24 December 2020

1. Introduction

Video streaming constitutes a large fraction of current Internet traffic. In particular, video streaming accounts for more than 65% of world-wide mobile downstream traffic [1]. Commercial streaming services such as YouTube and Netflix are implemented based on Dynamic Adaptive Streaming over HTTP (DASH) to enable conventional HTTP Web servers to provide high-quality streaming of media content according to available network bandwidth [2]. Since network bandwidth in the Internet varies extremely over time, DASH video players utilize Adaptive Bit Rate (ABR) techniques [3–7] to adjust video chunk requests that ask the encoding rate of a video chunk. Based on the available throughput estimated at the application layer, the ABR schemes seek to select the encoding rates that the network can support while maintaining reasonable video quality.

The throughput estimation at the application layer tends to be inaccurate, thus, rate adaptation approaches solely based on the estimated throughput can cause undesired behaviors, resulting in high variability, low-quality streaming, and frequent rebufferings [8]. In particular, the throughput estimation at mobile devices is more challenging since the available network bandwidth in the mobile environments often changes dynamically. To resolve this problem, we propose a discrete feedback control design using the playback buffer level as a signal, in which the controller targets the buffer level to maintain a reference level. By applying the buffer level-based rate control, our proposed scheme becomes robust to incorrectly estimated throughputs. However, it is inappropriate to simply apply the control theory to mobile video streaming, e.g., solving complex control equations can incur infeasible computation overhead in mobile devices, which are usually constrained by low-power CPU and limited power supply. To reduce the overhead of the controller in mobile devices, we introduce several heuristics such as a pre-computed control gain table after formulating the optimization problem. Through a simulation study, we demonstrate that our proposed scheme successfully satisfies our design goal.

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. Section 3 provides the background context for our work. We present our approach in Section 4. Section 6 evaluates the performance of our controller with simulations. Future work is discussed in Section 7 and we conclude in Section 8.

2. Related Work

Several pieces of research have been done to address issues in video streaming under dynamic network conditions, such as incorrect bandwidth estimation, smoothness in streaming quality, and fairness across users.

Tian et al. [6] propose a feedback controller to drive the playback buffer length to a set-point, using adjustment factors to control bit rate according to network bandwidth and balance responsiveness and smoothness. While their controller utilizes the buffer status to determine the bit rate, the final decision strongly depends on bandwidth prediction.

Tian et al. [9] extend the approach in [6] to address large bandwidth variations and energy efficiency in cellular/WiFi networks. The proposed scheme chooses lower bit rates than an estimated throughput as a larger throughput variance becomes larger.

De Cicco et al. [10] also introduce a feedback control based on a Proportional-Integral (PI) controller using the playback buffer level as a feedback signal to the controller without adjusting control gains for optimal control.

Huang et al. [3] propose a bit rate adaptation mechanism based only on playback buffer status. The basic idea behind their approach is that buffer length increases if a requested video rate is lower than the available bandwidth, and decreases otherwise.

Li et al. [5] focus on undesirable streaming behaviors, such as bit rate fluctuation, when multiple streaming clients compete for bandwidth. They show that these problems are because clients cannot perceive their fair-share bandwidth due to the discrete nature of the video bit rates. To resolve these issues, the authors introduce a proactive probing mechanism for estimating the target average video bit rate.

Yin et al. [7] define a streaming QoE optimization problem and propose a bit rate selection using the solutions. However, their approach requires pre-computing the required information for a specific video setting.

Further, there are several approaches to utilize artificial intelligence (AI) algorithms for video streaming. Mao et al. [11] propose the first ABR scheme that utilizes Deep Reinforcement Learning (DRL) to select bitrate for the next video chunk. Huang et al. [12] suggest another RL-based ABR scheme. These approaches require training procedures to obtain an optimal DRL model in advance. Further, although RL can help a mobile client to choose the most proper bit rate according to network status, even inferences using a neural network requires large computation overhead with high energy consumption. This is inappropriate for mobile devices with low computing power and limited power supply; even though recent high-end mobile devices introduce an AI accelerator to support AI algorithms with low energy consumption, mid-range devices, which are popular in the market, still have a lack of such functionalities. Lee et al. [13] suggest PERCEIVE using an LSTM (Long Short Term Memory) model to predict throughput according to cellular channel status. Similar to the previous approaches [11,12], PERCEIVE can have concerns for computation overhead in mobile devices and it also targets WebRTC, not DASH video streaming.

3. Background

To stream videos with a bit rate appropriate for the available bandwidth, a DASH server provides multiple representations of a video content encoded at different bit rates. Each representation is fragmented into small video chunks that contain several seconds of video. Based on measured available bandwidth, a DASH client selects a chunk representation, i.e., bit rate, and requests it from a DASH server; this is called adaptive bit rate selection.

A DASH client player starts a streaming session with an initial buffering phase during which a player fills its playback buffer to the maximum level (B_{max}). When the buffer is filled with the minimum amount of video during this phase, the player starts playing the video, and continues to retrieve media chunks until the initial buffering completes. After completing the initial buffering phase, a player pauses downloading until the buffer level falls below the maximum buffer level by playback. This leads to an ON-OFF traffic pattern where the player downloads chunks for a while and then waits until a specific number of chunks is consumed [14]. If the amount of buffered chunks is less than the minimum required to play out the video, the player stops playback and fills its buffer until it has a sufficient amount of video to begin playback again, called the rebuffering phase. Note that the length of a video chunk is one of the parameters that can affect the duration of the rebuffering phase; assume a client player starts playback again once the playback buffer contains at least one video chunk. As the length of a video chunk is longer, the chunk size becomes larger, resulting in a longer rebuffering duration until one chunk download for playback completes.

Yin et al. [7] models such a behavior by a DASH client player in terms of buffer status. Suppose L is the chunk length in seconds. Let B_k be the buffer level measured in seconds when the client starts downloading the ith chunk with a particular encoding bit rate R_k. Assume that D_k is the download time for the kth chunk. The length of the OFF period after the kth chunk download, δ_k, is defined as:

$$\delta_k = \max(\max(B_k - D_k, 0) + L - B_{max}, 0)\,.$$

The rebuffering duration after the kth chunk download, γ_k, is:

$$\gamma_k = \max(D_k - B_k, 0)\,.$$

Then, we can define the next buffer level, B_{k+1}, as:

$$B_{k+1} = \max(\max(B_k - D_k, 0) + L - \delta_k, 0)\,.$$

4. Approach

4.1. Linear Quadratic Optimization

The optimal control theory seeks to operate a system with the minimum cost. If the system dynamics are described by a set of linear differential equations and the cost a quadratic function, the linear quadratic optimization provides the solution for the control problem. In this section, we propose an ABR selection scheme (LQ) based on the control theory using a discrete-time linear quadratic regulator, which uses the playback buffer level as a feedback signal. Based on the control outputs from the linear quadratic regulator, LQ targets the buffer level to maintain the reference buffer level.

Figure 1 presents the control loop diagram to design the basic LQ ABR scheme and Table 1 presents its terms. Let $\hat{R}_k$ be the estimated bit rate for kth chunk by the controller output u_k. Suppose that the set of available m bit rates is $V = \{v_1, v_2, \ldots, v_m\}$ where $v_1 < v_2 < \ldots < v_m$. The basic LQ ABR scheme works as follows:

$$\begin{aligned} e_i &= B_i - q_0 \qquad i = 1 \ldots k \\ S_k &= \sum_{i=1}^{k-1} e_i \\ u_k &= -K_P e_k - K_I S_k \\ \hat{R}_k &= F\left(\min\left(\frac{1}{1(u_k > 0)u_k}, v_m\right)\right) \end{aligned}$$

where q_0 is the reference buffer level, B_i is the buffer level measured in second when ith chunk download starts, e_i is the error between the buffer level, S_k is the sum of previous errors, $F(x) = argmax_{v_i \in V}(v_i \leq x)$, which is a function to quantize a controller output to

an available chunk bit rate, and $1(condition)$ is a function to set one if the condition is satisfied and zero otherwise.

Table 1. Notations.

Term	Descriptions
	when kth chunk download starts
q_0	Reference buffer level in seconds
B_k	Buffer level in seconds
e_k	Error between the buffer level and q_0
S_k	Sum of previous errors
u_k	Controller output
C_0	Expected mean throughput
$\hat{R}_k$	Controller estimated bit rate for kth chunk
K_P	Proportional gain
K_I	Integral gain

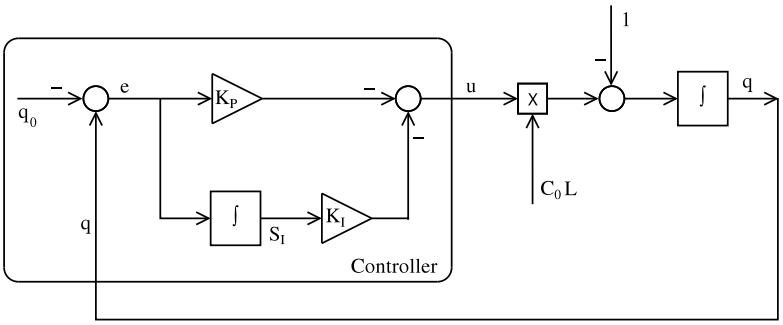


Figure 1. Control Loop.

In this controller system, we seek to keep e_k and S_k close to 0 while maximizing $\hat{R}_k$. We define a two dimensional state vector $\mathbf{x}_k = \begin{bmatrix} e_k \\ S_k \end{bmatrix}$ that evolves as follows:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k$$

where $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} L \times C_0 \\ 0 \end{bmatrix}$, and L is the chunk length measured in seconds.

Then, we formulate a quadratic cost optimization problem for an infinite-horizon where the goal is to minimize the cost function J, i.e., to keep $\mathbf{x}_k$ close to 0 using small u_k (large $\hat{R}_k$) as:

$$\min_{u_k} \quad J = \sum_{k=1}^{N} (\mathbf{x}_k^T \mathbf{Q} \mathbf{x}_k + \rho u_k^2) + \mathbf{x}_N^T \mathbf{Q} \mathbf{x}_N$$

where N is the number of chunks in an entire video, $\mathbf{Q}$ is a 2×2 dimensional symmetric positive definite *state* transition matrix, and ρ is a *control* weight.

Assuming $N \to \infty$, the gain matrix for optimal control, $\mathbf{K} = \begin{bmatrix} K_P & K_I \end{bmatrix}$, is given by [15]:

$$\mathbf{K} = (\rho + \mathbf{B}^T \mathbf{P} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}$$

where $\mathbf{P}$ is the solution of the discrete time algebraic Riccati equation

$$\mathbf{P} = \mathbf{A}^T \mathbf{P} \mathbf{A} - \mathbf{A}^T \mathbf{P} \mathbf{B} (\rho + \mathbf{B}^T \mathbf{P} \mathbf{B})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A} + \mathbf{Q}.$$

Finally, the control sequence of u_k minimizing the cost can be represented as:

$$u_k = -\mathbf{K}\mathbf{x}_k$$

where the obtained **K** represents the optimal values for K_P and K_I for the basic LQ ABR scheme.

Although our controller is based on the feedback control based on the level of playback buffer, it needs to estimate expected mean throughputs, which can dynamically fluctuate in the mobile environment. To predict mean throughput, our controller applies Holt-Winters time-series forecasting algorithm [16], which is known to be more accurate than formula-based predictors [17], on throughput samples measured while downloading each chunk.

4.2. Extending LQ for Improving QoE in Mobile Streaming

To avoid frequent bit rate fluctuation, which significantly degrades user quality of experience [18], and to continuously update the control gain parameters with optimal values while consuming feasible resources in mobile devices, we extend the basic LQ ABR, called LQE, as follows:

4.2.1. Adjustment Factor in the Error Term e_k

To prevent frequent bit rate switching in a short term, we redefine the error term e_k as:

$$e_k = B_k - q_0 - \sigma \times q_0(r_{k-2} - r_{k-1})$$

where r_k is the index of bit rate selected for kth chunk (increasing order according to bit rates), e.g., with six available bit rates the index of the minimum bit rate is one and the index of the maximum is six.

The intuition behind the adjustment factor is that a negative error drives LQ to select a lower bit rate and a positive error does LQ to select a higher bit rate. By switching to a lower bit rate, the error with the next feedback decreases (going to a positive error), thus, the client can shortly increase the bit rate, resulting in short-term bit rate fluctuation. It also occurs when LQ selects a higher bit rate. To mitigate the short-term fluctuations, the adjustment factor makes increases in the error in the negative direction than the actual error when LQE previously switches to a lower bit rate. Similarly, if the LQE observes a bit rate increase, the error increases in the positive direction, allowing LQE to continue with the selected higher bit rate for a longer time.

4.2.2. Counter-Based Switching Logic

In addition to the adjustment factor, we introduce a bit rate switching logic using a counter mechanism to more aggressively prevent short-term bit rate fluctuations. Since the LQ output is a continuous value, immediate switching based on the quantized value from the LQ output can incur oscillations at boundaries. The counter-based switching logic avoids such oscillations by introducing some hysteresis to the LQ ABR. To this end, LQE selects a bit rate R_k higher than R_{k-1} only if the estimated bit rates from the all controller outputs in previous m consecutive chunks ($\hat{R}_{i\in[k-m,k]}$) indicate to increase bit rate, that is, $\forall i \in [k-m,k]$, $\hat{R}_i > R_i \Rightarrow R_k = \hat{R}_k$. LQE uses the same counter-based switching for the case of switching-down bit rate.

4.2.3. Control Gain Table

The optimal control gains K_P and K_I depend on the chunk length L, the mean throughput C_0, and the control weight ρ. To enable LQ to continuously use optimal values for the parameters, LQE dynamically updates K_P and K_I according to chunk length and measured average bandwidth. However, solving the Riccati equation in a mobile device requires significant computational overhead. Therefore, we pre-compute K_P and K_I according to different chunk lengths and average bandwidths. The control gain table for a particular ρ represents this data as a two-dimensional array, indexed by the chunk length and the

measured average throughput, where each entry includes corresponding optimal values for K_P and K_I.

Table 2 shows entries in the control gain table for $\rho = 10^4$ indexed by every 0.5 Mbps. Note that the final output of the LQE is $\frac{10^6}{u_k}$ since we pre-compute this table according to bandwidth in Mbps. Figure 2 shows the flowchart of the LQE ABR logic to determine an appropriate bit rate for chunk downloading, where M_u and M_d are variables to count consecutive rate up/down selections, which is used for the counter-based switching logic described in Section 4.2.2.

Table 2. Control Gain Table ($\rho = 10^4$).

L (s)	C_0 (Mbps)	(K_P, K_I)
5	0.5	0.0297, 0.0010
	1.0	0.0219, 0.0009
	1.5	0.0185, 0.0009
...	...	...

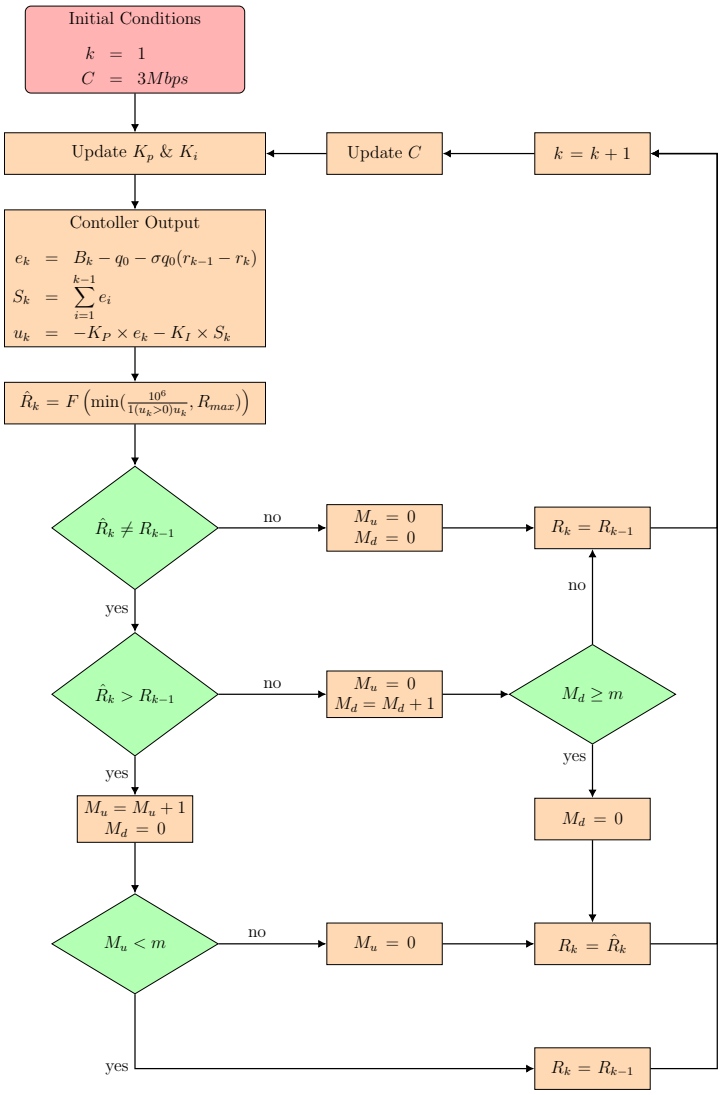


Figure 2. LQE Adaptive Bit Rate Selection.

4.2.4. Download Abandonment

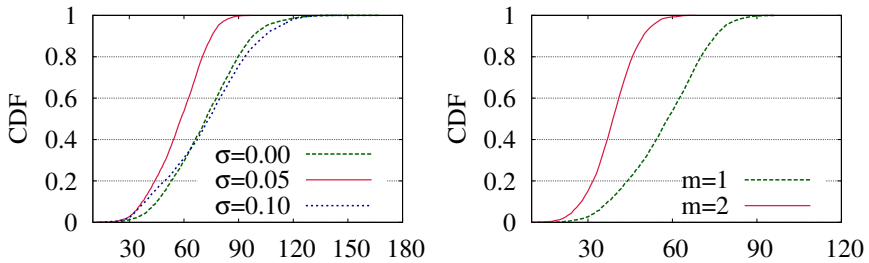
A sudden bandwidth drop in the mobile environments can cause buffer depletion while a player retrieves a chunk with a selected bit rate. To resolve this problem, if LQE

detects the possibility of buffer depletion while downloading a chunk, LQE abandons the current chunk download and chooses a new bit rate that can finish downloading before a buffer depletion, i.e., a download abandonment is triggered if the buffer level falls below a threshold while the expected remaining download time is longer than the buffer length. For the threshold in our implementation, we use $\frac{2}{3}B_k$ where B_K is the playback buffer level at the beginning of k-th chunk download.

5. Determining LQE Parameters

In this section, we explore the effect of LQE parameters, which are knobs to control the behavior of a client player according to different user demands. To this end, we examine LQE with trace-driven simulations with several parameter settings (See Section 6 for the simulation setup).

Figure 3a shows the CDFs of the number of bit rate changes in the 3570 scenarios according to different σ and m while we set q_0 & σ to 70s and 10^4, respectively. In these simulations, we use one for m, i.e., disable the count-based switching logic in LQE, so that we investigate the effect of the adjustment factor for reducing the number of bit rate changes. As shown in Figure 3a, the number of bit rate changes decreases until $\sigma = 0.05$ and starts to increase after σ becomes larger than 0.05. In the experiments, we observe that σ does not notably affect streaming quality such as the total rebuffering duration. Therefore, we select $\sigma = 0.05$, which yields the fewest number of bit rate changes without increasing the rebuffering duration.



(**a**) Effect of σ ($m = 1$) (**b**) Effect of m ($\sigma = 0.05$)

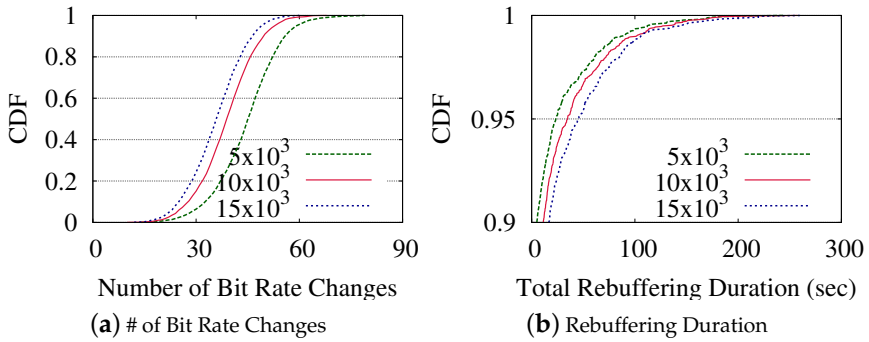
Figure 3. Effect of σ and m (LQE).

Given $\sigma = 0.05$, we explore the effect of the counter-based switching logic. As shown in Figure 3b, using $m \geq 2$ successfully enables LQE to reduce the number of bit rate changes together with the adjustment factor: $m = 2$ reduces the number of bit rate changes by almost half. However, as m increases, LQE response more slowly to network dynamics, resulting in a longer rebuffering duration: $m = 2$ triggers rebufferings in 3% more scenarios than $m = 1$ in our simulations. We use the counter threshold of $m = 2$, which significantly reduces the number of bit rate changes with the smallest impact on the rebuffering behavior.

Next, we investigate the effect of the control weight ρ on the number of bit rate changes and streaming quality. Figure 4 shows the CDFs of the number of bit rate changes of LQE with different values of ρ while we set σ & q_0 to 0.05 and 70 s, respectively. We observe that the number of bit rate changes decreases as ρ becomes larger, while a larger ρ results in longer total rebuffering durations. We choose $\rho = 10^4$ which compromises between the number of bit rate changes and the total rebuffering duration.

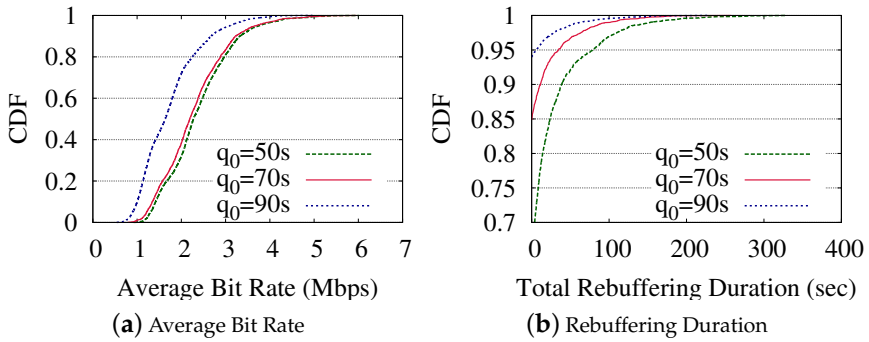
Figure 5 presents the CDFs of average bit rate and total rebuffering duration of LQE with different values of q_0 while σ and ρ are set to 0.01 and 10^4, respectively. We observe that the average bit rate and total rebuffering duration become smaller and shorter as q_0 increases, showing the trade-off between average bit rate and rebuffering. It is because a larger q_0 is likely to incur more negative error, resulting in lower bit rate selections, which can shorten the duration of rebuffering. For the rest of the experiments, we choose $q_0 = 70$

that yields similar average bit rates compared to $q_0 = 90$ while exhibiting moderately short rebuffering durations.



(**a**) # of Bit Rate Changes (**b**) Rebuffering Duration

Figure 4. Effect of ρ (LQE).



(**a**) Average Bit Rate (**b**) Rebuffering Duration

Figure 5. Effect of q_0 (LQE).

6. Trace-Driven Simulations

In this Section, we compare the performance of our streaming controller with following buffer-based ABRs:

- Tian: Tian et al. [6] use a feedback controller to drive the playback buffer level to a set-point by scaling predicted throughputs as a function of the buffer level and its trend. We use the control parameter $K_p = 0.1$ as described in [6].
- BBA: Huang et al. [3] use a rate adaptation that selects bit rate R_k based on a function of the playback buffer level. In our evaluation, the reservoir and cushion parameters for BBA are set to 20s and 70s, respectively.

We conduct trace-driven simulations using a set of 85 public available traces of cellular bandwidth [19]. Assuming that a mobile device obtains a sum of entries in two different traces as a bandwidth, we use combinations of the throughput traces to input bandwidth values for each interface. Note that if a trace is shorter than the simulation running time, we continue to repeat the trace from the beginning. This results in 3570 scenarios with which to investigate the performance of the streaming schemes, which is intended to reflect the bandwidth variability in real networks. In the simulation, we assume that the streaming server provides six representations of the video, of which length is 1800 s, with resolutions varying from 144 to 1080 p, of which bit rates are from 0.27 to 8.9 Mbps. Given a bandwidth scenario, our simulator calculates streaming bit rate, playback buffer, and rebuffering statuses based on the streaming behavior model [7] described in Section 3.

We use the following performance metrics to evaluate the ABR schemes:

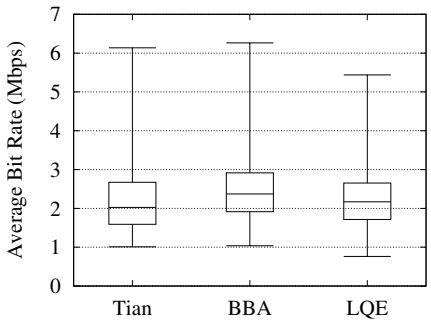
- Average bit rate: This is the average of the bit rate of all downloaded chunks, which is defined as $\frac{\sum_{k=1}^{K} R_k}{K}$ where K is the total number of chunks and R_k is the bit rate of the kth chunk.

- Average rebuffering duration: This is defined as the time spent in the rebuffering phase divided by the number of rebuffering occurrences during the entire playback.
- Number of bit rate changes: We count the number of times the bit rate increases or decreases during the entire playback, i.e., $\sum_{k=2}^{K} 1(R_{k-1} \neq R_k)$.

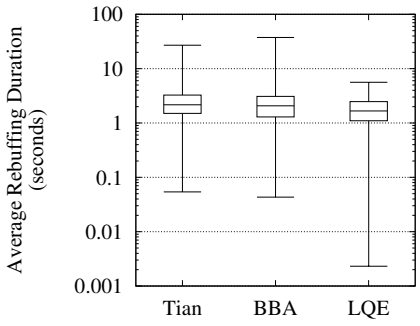
Figure 6 presents Whisker plots showing the minimum, first quartile (Q1), median, third quartile (Q3), and maximum of average bit rate and average rebuffering duration and CDF of the number of bit rate changes. As shown in Figure 6a, in terms of median, LQE yields a higher average bit rate (2.17 Mbps) than Tian (2.02 Mbps) and slightly lower than BBA (2.37 Mbps) while min/max of LQE is lower than others. It is because LQE is likely to use lower bit rates when LQE expects low available bandwidth status so that it can more properly mitigate rebufferings as shown in Figure 6b.

Note that Figure 6b exhibits the Whisker plot of collected samples only in traces where rebufferings occur. In our simulation, rebufferings did not happen in 93% of traces with Tian, 88% with BBA, and 91% with LQE. Although LQE experiences rebufferings in 2% more traces than Tian, LQE exhibits the shortest average rebuffering duration than others as shown Figure 6b. In particular, its maximum of average rebuffering duration is 5.6 s while others' are larger than 25 s. In terms of median, LQE yields the shortest rebuffering duration (1.67 s) followed by BBA (2.07 s) and Tian (2.16 s). This shows that LQE successfully shortens the duration of rebuffering when it happens by appropriately choosing bit rates in advance, of which average can be higher than what Tian chooses and should be slightly lower than what BBA does.

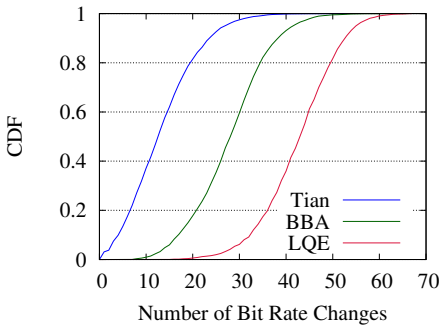
In Figure 6c, we observe that LQE exhibits the largest number of bit rate changes followed by BBA and Tian. This is because LQE is more adaptively switching bit rates according to available network bandwidth than the other ABRs. Although BBA changes bit rates more often than Tian, BBA suffers from longer rebufferings than Tian while LQE does shorter rebufferings as shown in Figure 6b. This shows that BBA switches bit rate at incorrect timing for mitigating rebufferings while LQE does at appropriate moments that can shorten rebuffering duration.



(**a**) Average Bit Rate



(**b**) Average Rebuffering Duration

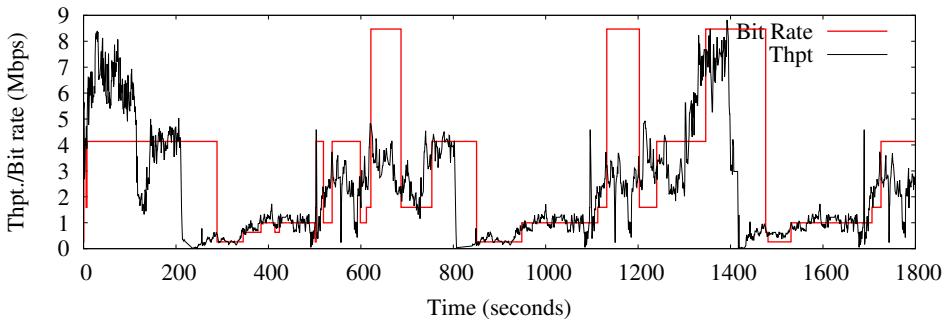


(**c**) Number of Bit Rate Changes

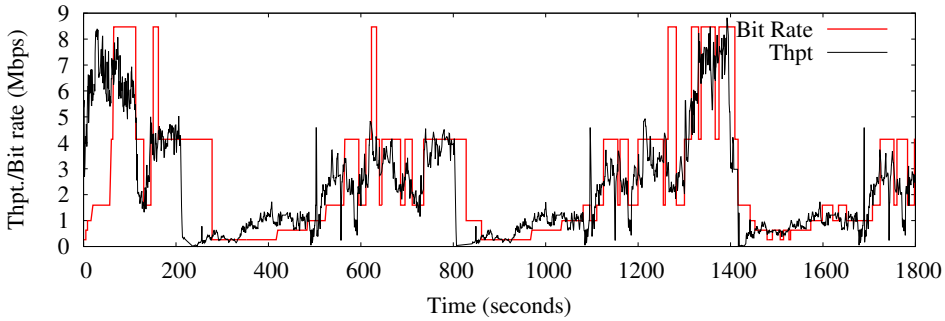
Figure 6. Comparison of ABRs.

In sum, by proactively switching bit rates, LQE exhibits up to 22% shorter average rebuffering duration than the other ABRs while achieving up to 8% lower average bit rate; this is an inevitable bit rate degradation to successfully mitigate rebufferings in the experiment scenarios.

Figure 7 exhibits example traces showing how Tian and LQE select bit rates according to available throughput in a selected scenario. In this figure, we compare two extreme cases in terms of the number of bit rate changes: Tian with the least number of changes and LQE with the largest number of changes. As we can expect in Figure 6c, LQE changes bit rates more adaptively than Tian: for example, LQE switches bit rates more often during the period between 1100 and 1500 s while Tian tends to stick with one selected bit rate. In particular, during the periods of 1150–1200 and 1400–1450, Tian preserves the higher bit rate than the available throughput, which can result in rebufferings, while LQE accordingly changes bit rates.



(**a**) Tian



(**b**) LQE

Figure 7. Examples of Bit Rate Changes according to Throughput.

7. Future Work

Although our simulation study proves that LQE successfully fulfills the design goal while exhibiting better performance compared to the existing ABRs, it is important to investigate its performance in the real world by using an implementation in an off-the-shelf mobile device. For the experiments in the real world, we have implemented an Android DASH client players based on Google Exoplayer [20]. Our implementation includes LQE as well as the other ABR schemes such as Tian and BBA using the reference V1 of the ExoPlayer, in which FormatEvaluator class takes charge of adaptive video streaming. We are currently porting our implementation to the latest version (V2) of the ExoPlayer. In future work, we will construct a test environment that consists of a testing video streaming server, our latest DASH client player, and UI to collect feedback from users. We will examine the performance metrics in real experiments together with the additional quality of service metrics such as Mean Opinion Score (MOS), which is a measure representing the overall perceived quality of streaming based on user feedback.

8. Conclusions

This paper proposes and evaluates LQE, a Linear Quadratic Optimization-based ABR scheme, which chooses a proper video bit rate to reduce playback stalls due to rebufferings. We introduce several tweaks to utilize the complex control theory appropriately for mobile video streaming. Our simulation results show that LQE successfully mitigates rebufferings by decreasing their duration when they occur compared to the existing buffer-level based ABR schemes, while still preserving similar video streaming quality.

Author Contributions: Conceptualization, Y.-m.K. and Y.-s.L.; methodology, Y.-m.K. and Y.-s.L.; software, Y.-s.L.; validation, Y.-m.K.; formal analysis, Y.-m.K.; writing—original draft preparation, Y.-m.K.; writing—review and editing, Y.-s.L.; visualization, Y.-m.K.; supervision, Y.-s.L.; project administration, Y.-s.L.; funding acquisition, Y.-s.L. Both authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Sungshin Women's University Research Grant of H20200096.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sandvine. The Mobile Internet Phenomena Report. 1H 2020. Available online: https://www.sandvine.com/download-report-mobile-internet-phenomena-report-2020-sandvine (accessed on 10 November 2020).
2. Stockhammer, T. Dynamic Adaptive Streaming over HTTP—Standards and Design Principles. In Proceedings of the ACM MMSys, Santa Clara, CA, USA, 23–25 February 2011; pp. 133–144.
3. Huang, T.Y.; Johari, R.; McKeown, N.; Trunnell, M.; Watson, M. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In Proceedings of the ACM SIGCOMM, Chicago, IL, USA, 17–22 August 2014; pp. 187–198.
4. Jiang, J.; Sekar, V.; Zhang, H. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Trans. Netw.* **2014**, *22*, 326–340. [CrossRef]
5. Li, Z.; Zhu, X.; Gahm, J.; Pan, R.; Hu, H.; Begen, A.; Oran, D. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 719–733. [CrossRef]
6. Tian, G.; Liu, Y. Towards Agile and Smooth Video Adaptation in Dynamic HTTP Streaming. In Proceedings of the ACM CoNEXT, Nice, France, 10–13 December 2012; pp. 109–120.
7. Yin, X.; Jindal, A.; Sekar, V.; Sinopoli, B. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In Proceedings of the ACM SIGCOMM, London, UK, 17–21 August 2015; pp. 325–338.
8. Huang, T.Y.; Handigol, N.; Heller, B.; McKeown, N.; Johari, R. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In Proceedings of the ACM IMC, Boston, MA, USA, 14–16 November 2012; pp. 225–238.
9. Tian, G.; Liu, Y. On Adaptive HTTP Streaming to Mobile Devices. In Proceedings of the 2013 20th International Packet Video Workshop (PV), San Jose, CA, USA, 12–13 December 2013; pp. 1–8.
10. De Cicco, L.; Caldaralo, V.; Palmisano, V.; Mascolo, S. ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP DASH. In Proceedings of the 2013 20th International Packet Video Workshop (PV), San Jose, CA, USA, 12–13 December 2013; pp. 1–8.
11. Mao, H.; Netravali, R.; Alizadeh, M. Neural Adaptive Video Streaming with Pensieve. In Proceedings of the ACM SIGCOMM, Los Angeles, CA, USA, 21–25 August 2017; pp. 197–210.
12. Huang, T.; Zhou, C.; Yao, X.; Zhang, R.X.; Wu, C.; Yu, B.; Sun, L. Quality-Aware Neural Adaptive Video Streaming With Lifelong Imitation Learning. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 2324–2342. [CrossRef]
13. Lee, J.; Lee, S.; Lee, J.; Sathyanarayana, S.D.; Lim, H.; Lee, J.; Zhu, X.; Ramakrishnan, S.; Grunwald, D.; Lee, K.; et al. PERCEIVE: Deep Learning-Based Cellular Uplink Prediction Using Real-Time Scheduling Patterns. In Proceedings of the ACM MobiSys, Toronto, ON, Canada, 15–19 June 2020; pp. 377–390.
14. Rao, A.; Lim, Y.S.; Barakat, C.; Legout, A.; Towsley, D.; Dabbous, W. Network Characteristics of Video Streaming Traffic. In Proceedings of the ACM CoNEXT, New York, NY, USA, 6–9 December 2011; pp. 25:1–25:12.
15. Goodwin, G.C.; Graebe, S.F.; Salgado, M.E. *Control System Design*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2000.
16. Rockwell, P.J.; Davis, R.A. *Introduction to Time Series and Forecasting*; Springer-Verlag: New York, NY, USA, 2002.
17. He, Q.; Dovrolis, C.; Ammar, M. On the Predictability of Large Transfer TCP Throughput. In Proceedings of the ACM SIGCOMM, Philadelphia, PA, USA, 22–26 August 2005; pp. 145–156.
18. Mok, R.K.P.; Luo, X.; Chan, E.W.W.; Chang, R.K.C. QDASH: A QoE-aware DASH System. In Proceedings of the ACM MMSys, Chapel Hill, NC, USA, 22–24 February 2012; pp. 11–22.

19. Riiser, H.; Vigmostad, P.; Griwodz, C.; Halvorsen, P. Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications. In Proceedings of the ACM MMSys, Oslo, Norway, 27 February–1 March 2013; pp. 114–118.
20. Google. ExoPlayer. Available online: https://github.com/google/ExoPlayer (accessed on 10 November 2020).