A Control Theoretic Approach to ABR Video Streaming: A Fresh Look at PID-based Rate Adaptation

Yanyuan Qin*, Ruofan Jin*, Shuai Hao†, Krishna R. Pattipati*, Feng Qian‡,
Subhabrata Sen†, Bing Wang*, Chaoqun Yue*
*University of Connecticut, †AT&T Labs Research, ‡Indiana University

***Abstract*—Adaptive bitrate streaming (ABR) has become the *de facto* technique for video streaming over the Internet. Despite a flurry of techniques, achieving high quality ABR streaming over cellular networks remains a tremendous challenge. ABR streaming can be naturally modeled as a feedback control problem. There has been some initial work on using PID, a widely used feedback control technique, for ABR streaming. Existing studies, however, either use PID control directly without fully considering the special requirements of ABR streaming, leading to suboptimal results, or conclude that PID is not a suitable approach. In this paper, we take a fresh look at PID-based control for ABR streaming. We design a framework called PIA that strategically leverages PID control concepts and incorporates several novel strategies to account for the various requirements of ABR streaming. We evaluate PIA using simulation based on real LTE network traces, as well as using real DASH implementation. The results demonstrate that PIA outperforms state-of-the-art schemes in providing high average bitrate with significantly lower bitrate changes (reduction up to 40%) and stalls (reduction up to 85%), while incurring very small runtime overhead.**

I. Introduction

Video streaming has come to dominate mobile data consumption today. As per Cisco's 2016 Visual Network Index report [2], mobile video traffic now accounts for more than half of all mobile data traffic. Despite much effort, achieving good quality video streaming over cellular networks remains a tremendous challenge. In fact, a report in late 2014 indicates significant stalling (10% to 40% depending on video quality) in both Europe and North America's cellular networks [3].

Most video content are currently streamed using Adaptive Bit-Rate (ABR) streaming over HTTP, the *de facto* technology adopted by industry. In ABR streaming, a video is encoded into multiple resolutions/quality levels. The encoding at each quality level is divided into small chunks, each containing data for some time intervals' worth of playback (e.g., several seconds). A chunk at a higher quality level requires more bits to encode and is therefore larger in size. During playback, the video player determines in real-time which quality level to fetch according to its adaptation algorithm.

Various user engagement studies [10], [13], [15], [17] indicate that satisfactory ABR streaming needs to achieve three conflicting goals simultaneously: (1) maximize the playback bitrate; (2) minimize the likelihood of stalls or rebuffering; and (3) minimize the variability of the selected video bitrates for a smooth viewing experience. Reaching any of the three goals alone is relatively easy – for instance, the player can simply stream at the highest bitrate to maximize the video quality; or it can stream at the lowest bitrate to minimize the stalls. The challenge lies in achieving all three goals simultaneously, especially over highly varying network conditions, typical of the last-mile scenarios in cellular networks.

Rate adaptation for ABR streaming can be naturally modeled as a control problem: the video player monitors the past network bandwidth and the amount of content in the playback buffer to decide the bitrate level for the current chunk; the decision will then affect the buffer level, which can be treated as feedback to adjust the decision for the next chunk. PID (named after its three correcting terms, namely "proportional", "integral", and "derivative" terms) is one of the most widely used feedback control technique in practice [6]. It is conceptually easy to understand and computationally simple.

There has been initial work on using PID control theory for ABR streaming. However, the message so far has been mixed and inconclusive. The studies [8], [9] directly apply the standard PID controller to ABR streaming with no modifications. As a result, video bitrate may fluctuate significantly as shown in the evaluation results. The studies [20], [22] conclude that PID control is not a suitable approach since the goal of PID control is misaligned with the goals of ABR streaming.

A. Contributions

In this paper, we adopt a contrarian perspective and take a fresh look at the potential of PID control for ABR video streaming. We start by pointing out that a recent heuristic technique with commercial deployment, BBA [11], in fact can be shown to be, in effect, using a simplified form of PID control. We then conduct an in-depth exploration of using PID control for ABR streaming. We design *PIA* (PID-control based ABR streaming), a novel control-theoretic video streaming scheme that strategically leverages PID control concepts as the base framework, and further incorporates domain knowledge of ABR streaming to improve the robustness and adaptiveness of video playback. Our main contributions include the following.

- We take a fresh look at PID-based control for ABR streaming. We strategically leverage PID control concepts as the base framework for PIA. The *core controller* in PIA differs from those in [8], [9] in that we define a control policy that makes the closed-loop control system linear, and easy to control and analyze. The core controller maintains the playback buffer to a target level, so as to reduce rebuffering.
- We add domain-specific enhancements to further improve the robustness and adaptiveness of ABR streaming. Specifi-

cally, PIA addresses two key additional requirements of ABR streaming, i.e., maximizing playback bitrate and reducing frequent bitrate changes. It also incorporates strategies to accelerate initial ramp-up and protect the system from saturation.
• We explore parameter tuning. K_p and K_i (defined in Section II) are the two fundamental and most critical parameters that guide the PIA controller's behavior. We develop a methodology that systematically examines a wide spectrum of network conditions and parameter settings to derive their (K_p, K_i) configurations that yield satisfactory quality of experience (QoE). A service provider can use this approach to appropriately tune (K_p, K_i) for their environment.

We conduct comprehensive evaluations of PIA using a large number of real cellular network traces with diverse network variability characteristics. The traces were collected from two commercial LTE networks at diverse geographic locations. Our key findings include the following.
• PIA achieves comparable bitrates as two state-of-the-art schemes, BBA [11] and MPC [22], while substantially reducing bitrate changes (49% and 40% lower respectively) and rebuffering time (68% and 85% lower respectively). Overall, PIA achieves the best balance among the three QoE metrics.
• PIA has low computation overheads (e.g., comparable to BBA and only 0.5% of MPC based on our simulation). Our emulation results also show that the execution time of PIA is less than 2 seconds for a 15-minute video.
• We also found that a common set of (K_p, K_i) values exist that have good performance across all the traces. This is important as it suggests that a single appropriately selected (K_p, K_i) pair can be used across a wide range of network conditions, facilitating easy deployment of our streaming scheme.

B. Related Work

Overall, our work differs from prior efforts that directly apply PID controller to video streaming without any adaptation. Instead, we show a somewhat surprising high-level finding: *by applying control theories in an explicit and adaptive manner, our ABR streaming algorithm substantially outperforms state-of-the-art video streaming schemes.* Besides the initial work on using control theory for ABR streaming [8], [9], [20], [22] as described earlier, we briefly review other recently proposed schemes for ABR streaming. BOLA [19] improves BBA by selecting the bitrate to maximize a utility function considering both rebuffering and video quality. FESTIVE [12] and PANDA [14] both consider scenarios with multiple video flows. piStream [21] is designed specifically for LTE networks and uses PHY-layer information to improve the bandwidth prediction. As another related work, [7] proposes a network-based scheduling framework for adaptive video delivery over cellular networks. The work [23] demonstrates the benefits of knowing network bandwidth on the performance of ABR streaming. None of them focuses specifically on leveraging control theory for improving the video streaming QoE.

II. Motivation

For a satisfactory user-perceived QoE, ABR streaming needs to optimize several conflicting goals, including maximizing average playback rate, minimizing stalls (or rebuffering), and reducing sudden and frequent quality variations [10], [13], [15], [17]. We next formulate ABR streaming as a control problem and describe the motivation for our study. Table I summarizes the main notation used in this paper.

C_t	Network bandwidth at time t
x_t	Buffer level at time t
x_r	Target buffer level
R_t	Selected video bitrate for time t
Δ	Video Chunk Size
δ	Startup latency
u_t	PID controller output
K_p, K_i, K_d	PID controller parameters
ζ	Damping ratio
ω_n	Natural frequency
β	Setpoint weighting parameter

TABLE I: Key notation.

A. ABR streaming as a control problem

Deciding which bitrate level to choose can be modeled as a control problem. Specifically, let x_t be the buffer level of the video player at time t, C_t the real-time network bandwidth at time t, and R_t the bitrate of the video chunk that is being downloaded at time t. Further, let Δ denote the video chunk size (i.e., the duration of its playback time), δ denote the startup delay, i.e., how long it will take for the player to start playing. Then the player's buffer dynamics can be written as

$$\dot{x}_t = \begin{cases} \frac{C_t}{R_t}, & \text{if } t \le \delta \\ \frac{C_t}{R_t} - \mathbf{1}(x_t - \Delta), & \text{otherwise} \end{cases} \quad (1)$$

where $\mathbf{1}(x_t - \Delta) = 1$ if $x_t \ge \Delta$; otherwise, $\mathbf{1}(x_t - \Delta) = 0$. In other words, the playback of a chunk is only started after the entire chunk has been downloaded (a chunk contains meta data and hence the player needs to wait until the entire chunk is downloaded).

In (1), $\dot{x}_t$ is the rate of change of the buffer at time t. Here, C_t/R_t models the relative buffer filling rate. If $C_t > R_t$, i.e., the actual network bandwidth is larger than the bitrate of the video chunk being downloaded, the buffer level will increase. Otherwise, the buffer level will be at the same level (if $C_t = R_t$) or decrease (if $C_t < R_t$).

One simple control strategy is to select the video bitrate for each chunk based on the prediction of real-time link bandwidth, $\widetilde{C}_t$. Specifically, it simply chooses the highest bitrate that is less than $\widetilde{C}_t$. This is an open-loop control (there is no feedback; the decision is based only on the current state and the model of the system). It is not robust against network link bandwidth estimation errors. As an example, it may choose a high video bitrate if the estimated bandwidth, $\widetilde{C}_t$, is high, even if the current playback buffer level is very low. If it turns out that $\widetilde{C}_t$ is an overestimate of the actual network bandwidth, the buffer can be further drained and become empty, causing stalls. Closed-loop control (or feedback control) is more effective in dealing with network link bandwidth estimation errors.

B. PID control

As mentioned earlier, PID control is by far the most common way of using feedback in engineering systems. A PID

controller works by continuously monitoring an "error value" defined as the difference between the setpoint and measured process variable [6]. Specifically, let u_t represent the control output, and e_t the error feedback at time t. Then

$$u_t = K_p e_t + K_i \int_0^t e_\tau d\tau + K_d \frac{de_t}{dt}, \tag{2}$$

where the three parameters K_p, K_i and K_d are all non-negative, and denote the coefficients for the proportional, integral and derivative terms, respectively. As defined above, a PID controller takes account of the present, past and future values of the errors through the three terms, respectively. Some applications may require using only one or two terms to provide the appropriate system control. This is achieved by setting the other parameters to zero. A PID controller is called a PI, PD, P or I controller in the absence of the respective control actions [6].

In the video streaming scenario, the real-time buffer level is the measured process variable, and the reference buffer level is the setpoint. We next show that a recent state-of-the-art buffer based scheme, BBA [11], can be mapped to a P-controller (though the paper does not claim any control-theoretic underpinnings). In BBA, the video player maintains a buffer level, and empirically sets two thresholds, $\theta_{\text{high}} > \theta_{\text{low}}$. If the buffer level is below θ_{low}, the video player always picks the lowest bitrate, R_{min}; if the buffer level is above θ_{high}, the video player picks the highest bitrate, R_{max}; otherwise, the video player picks the video bitrate proportionally to buffer level. The selected bitrate, R_t, can therefore be represented as

$$R_t = \begin{cases} R_{\text{min}}, & x_t < \theta_{\text{low}}, \\ \frac{R_{\text{max}} - R_{\text{min}}}{\theta_{\text{high}} - \theta_{\text{low}}} (x_t - \theta_{\text{low}}) + R_{\text{min}}, & \theta_{\text{low}} \le x_t \le \theta_{\text{high}} \\ R_{\text{max}}, & x_t > \theta_{\text{high}} \end{cases}$$

Comparing the above with (2), we see that it is equivalent to a P-controller when $x_t \in [\theta_{\text{low}}, \theta_{\text{high}}]$ with $K_p = \frac{R_{\text{max}} - R_{\text{min}}}{\theta_{\text{high}} - \theta_{\text{low}}}$, $K_i = 0$ and $K_d = 0$.

The above scheme has been tested successfully in a large-scale deployment [11], indicating that a PID-type control framework has potential for ABR streaming. On the other hand, P-controller only considers the present error (i.e., the proportional term), and ignores the other two terms. It is well known that the absence of an integral term in a system may prevent the system from reaching its target value [6]. This is especially true for video streaming where inaccurate network bandwidth estimations may cause the error to accumulate over time. Therefore, including the integral term can potentially further improve the performance of [11]. PID's ability to address accumulative errors is advantageous compared to model predictive control (MPC) based approach in [22], which does not consider accumulative errors and requires accurate network bandwidth prediction. In addition, MPC is much more computationally intensive than PID.

We investigate PID-based control for ABR streaming in this paper, motivated by the widespread adoption of PID control in practice and BBA. Using PID for ABR streaming, however, has several challenges. First, the goal of PID control is to maintain a target buffer level that is only indirectly related to QoE. Indeed, while maintaining the buffer at a target level can help in preventing rebuffering, it does not help with the other two metrics on playback quality and bitrate variation. Second, PID is often used in continuous time and state space, while video streaming is a discrete-time system where the decisions are made at chunk boundaries and the video bitrate levels are discrete. Finally, while PID is conceptually simple, the parameters (K_p, K_i and K_d) need to be tuned carefully. Here important questions are how to choose these parameters, and to find out whether there exists a parameter set that is applicable to a wide range of settings. Some of the above challenges have been pointed out in [20], [22], which take the position that PID is not suitable for ABR streaming. As we shall show, none of the above challenges is a fundamental hurdle for using PID-based control for ABR streaming.

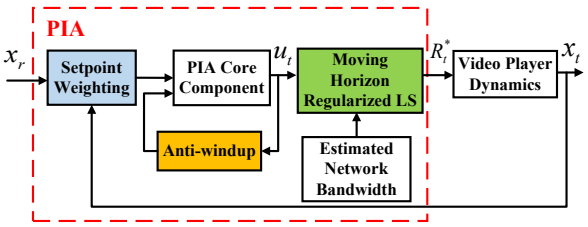


Fig. 1: PIA main components.

III. Adapting PID Control for ABR Streaming

We propose PIA, a PID based rate adaption algorithm for ABR streaming. As shown in Fig. 1, it contains a PI-based core control block as well as three mechanisms to address specific requirements for ABR streaming. We first describe the core component, and then the three performance enhancing mechanisms.

A. PIA core component

The core component of PIA adjusts the standard PID control policy in equation (2) so that the resultant closed-loop system is linear, and hence easier to control and analyze. We next define the controller output, analyze the system behavior, and provide insights into how to choose the various parameters.

Recall the dynamic video streaming model in (1), where x_t is the video player buffer level at t, C_t is the network bandwidth at time t, and R_t is the video bitrate chosen for time t. We define the controller output, u_t, as

$$u_t = \frac{C_t}{R_t}, \tag{3}$$

and set the control policy as

$$u_t = K_p(x_r - x_t) + K_i \int_0^t (x_r - x_\tau)d\tau + \mathbf{1}(x_t - \Delta) \tag{4}$$

where K_p and K_i denote, respectively, the parameters for proportional and integral control, x_r denotes target buffer level, and Δ is the chunk size. The choice of x_r depends on system constraints, a point we will come back to in Section IV.

The above control policy differs from the standard PID control policy (2) in the last term $\mathbf{1}(x_t - \Delta)$, which is a novel

aspect of our design. As we shall see, it provides linearity, making the closed-loop control system easier to control and analyze. In our control policy, the parameter for derivative control $K_d = 0$ (hence strictly speaking, our controller is a PI controller). This is because derivative action is sensitive to measurement noise [6] and measuring network bandwidth in our context is prone to noise.

Intuitively, u_t defined in (3) is a unitless quantity representing the relative buffer filling rate. With u_t selected, based on (3), the player can select the corresponding bitrate as

$$R_t = \frac{\widetilde{C}_t}{u_t}, \tag{5}$$

where $\widetilde{C}_t$ is the estimated link bandwidth at time t. Since video bitrate levels are discrete, we can choose the bitrate to be the highest that is below $\widetilde{C}_t/u_t$. This choice of R_t can increase, decrease or maintain the buffer level.

We next analyze the system to provide insights into its behavior as well as providing guidelines in choosing parameters. Combining equations (1) and (4) yields

$$\dot{x}_t = u_t - \mathbf{1}(x_t - \Delta) = K_p(x_r - x_t) + K_i \int_0^t (x_r - x_\tau)d\tau, \tag{6}$$

when the video starts playback (i.e., when $t \geq \delta$). We see that it is a linear system. Taking Laplace transform on both sides of (6) yields

$$sx(s) = K_p(x_r(s) - x(s)) + \frac{K_i}{s}\left(x_r(s) - x(s)\right), \tag{7}$$

where s is a complex variable. Let $T(s)$ be the system transfer function, which describes the relationship of the input and output of a linear time-invariant system. From (7), we have

$$T(s) = \frac{x(s)}{x_r(s)} = \frac{K_p s + K_i}{s^2 + K_p s + K_i}, \tag{8}$$

which is a second-order system. From (8), we have

$$2\zeta\omega_n = K_p, \quad \omega_n^2 = K_i\,, \tag{9}$$

where ζ and ω_n are damping ratio and natural frequency, respectively, two important properties of the system. Solving the above two equations, we have

$$\zeta = \frac{K_p}{2\sqrt{K_i}}, \quad \omega_n = \sqrt{K_i}\,. \tag{10}$$

Damping ratio represents the system's ability of reducing its oscillations. In our context, it measures how the buffer will oscillate around the target buffer level — small damping will cause the buffer to change rapidly, while large damping will cause the buffer to change slowly. Natural frequency represents the frequency at which a system tends to oscillate in the absence of any driving or damping force. Empirically it is found that ζ in the range $[0.6, 0.8]$ yields a very good system performance [18]. As a result, K_p and K_i should be chosen so that $\zeta \in [0.6, 0.8]$. We will discuss how to tune those parameters in Section IV-B.

B. PIA performance enhancing techniques

Based on the core component of PIA, we now add three domain-specific enhancements to further improve the robustness and adaptiveness for ABR streaming.

Accelerating initial ramp-up. At the beginning of the video play, the buffer level x_t can be much smaller than the target buffer level x_r. In this case, observe from (4) that u_t will be large, which will result in low video bitrate, R_t. To address this issue, we include a setpoint weighting parameter [6], $\beta \in (0, 1]$, into the control policy as

$$u_t = K_p(\beta x_r - x_t) + K_i \int_0^t (x_r - x_\tau)d\tau + \mathbf{1}(x_t - \Delta). \tag{11}$$

Note that β is only included in the proportional term; it does not affect the steady-state behavior of the control system [6]. When $\beta = 1$, the above control policy reduces to (4). When $\beta < 1$, it can lead to smaller u_t, and hence faster initial ramp-up in video bitrate. However, very small β can lead to aggressive choice of video bitrate, and hence rebuffering at the beginning of the playback. We explore how to set β in Section IV-B. The control function corresponding to the above control policy is

$$T(s) = \frac{x(s)}{x_r(s)} = \frac{\beta K_p s + K_i}{s^2 + K_p s + K_i}\,. \tag{12}$$

Therefore, both damping ratio and natural frequency remain the same as before.

Minimizing bitrate fluctuations. The simple choice of R_t in (5) mainly tracks the network bandwidth and does not take into account the smoothness of the video; it may lead to frequent and/or abrupt bitrate changes. To address the above issue, we develop a regularized least squares (LS) formulation that considers both video bitrate and the changes in video bitrate to achieve a balance between both of these metrics. Specifically, it minimizes the following objective function

$$J(R_t) = \sum_{k=t}^{t+L+1} \left(u_k R_t - \hat{C}_k\right)^2 + \eta\left(R_t - R_{t-1}\right)^2, \tag{13}$$

where R_{t-1} is the video bitrate for chunk $t-1$ (i.e., the previous chunk), u_k is the controller output for the k-th chunk (based on the decision of using R_t as the bitrate), $\hat{C}_k$ is the estimated link bandwidth for the k-th chunk, and η is the weight factor for bitrate changes. To reduce the number of video bitrate changes, the formulation assumes that the bitrate for the next L chunks to be the same, all equal to R_t. In (13), the first term in the sum aims to minimize the difference between $u_k R_t$ and the estimated network bandwidth $\hat{C}_k$ (so as to maximize R_t under the bandwidth constraint and selected u_k); the second term aims to minimize the bitrate changes compared to R_{t-1}; the weight factor η can be set to reflect the relative importance of these two terms. We use $\eta = 1$ (i.e., equal importance) in the rest of the paper.

The above formulation takes into account both the history (i.e., R_{t-1}) and future time slots (through a moving horizon of L chunks into the future). In each moving horizon, the control output u_k is updated according to the control policy

(11), base on the estimated x_k when choosing R_t as the video bitrate. Note that the above formulation does not need to consider rebuffering explicitly because PID already maintains the playback buffer to the target level (so as to avoid rebuffering).

Let $\mathcal{R}$ denote the set of all possible bitrates. The solution of (13) is

$$R_t^* = \arg \min_{R_t \in \mathcal{R}} J(R_t). \quad (14)$$

We can find R_t^* easily by plugging in all possible values of R_t, $R_t \in \mathcal{R}$, into (13), and find the value that provides the minimum objective function value in (13). For every $R_t \in \mathcal{R}$, obtaining $J(R_t)$ requires computation of L steps. Therefore, the total computation overhead is $O(|\mathcal{R}|L)$. This is significantly lower than the complexity of $O(|\mathcal{R}|^L)$ in [22].

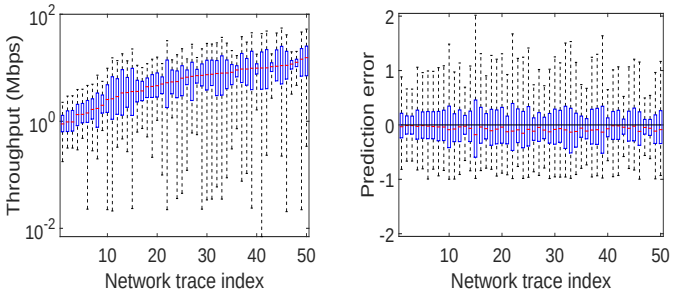
Dealing with bitrate saturation. Following the control policy, u_t may become negative (e.g., when the current buffer level exceeds the target buffer level). In this case, solving (13) will lead R_t to be the minimum bitrate. During this time period, if we continue using the integral term, $I_{\text{out}} = K_i \int_0^t (x_r - x_\tau)d\tau$, u_t may remain negative for an extended period of time, causing R_t to stay at the minimum bitrate level for an extended period of time (so called system saturation [6]), and causing the buffer level to continue to grow. To deal with the above scenario, we incorporate an anti-windup technique (to deal with integral windup, i.e., integral term accumulates a significant error) for negative u_t, or more specifically, when $u_t \leq \epsilon$, $0 < \epsilon \ll 1$. Many anti-windup techniques have been proposed in the literature [6]. We adopt a simple technique which sets R_t to the maximum value, $u_t = \epsilon$, and does not change I_{out} when $u_t \leq \epsilon$. This corresponds to turning off the integral control when u_t is below ϵ. We set ϵ to a small positive value, 10^{-10}, in the rest of the paper.

C. PIA parameter tuning

Three important parameters in PIA are K_p, K_i and β, where K_p and K_i determine the system behavior and β is used for faster initial ramp-up. For a given network setting (e.g., cellular networks), since β does not affect the steady-state behavior [6], we can first assume a fixed β (e.g., $\beta = 1$) and tune K_p and K_i to achieve a desirable steady-state behavior (i.e., jointly maximize the three metrics in QoE). Once K_p and K_i are fixed, we then tune β for the initial stage of the video playback. The values of K_p and K_i need to be tuned so that the resultant system behavior is compatible with the network setting. Taking cellular networks as an example, since the bandwidth is highly dynamic, it is reasonable to tune the system so that the buffer level does not fluctuate drastically. Otherwise, the buffer can suddenly become very low, making the system vulnerable to stalls. We describe this approach using a set of network traces from commercial cellular networks in Section IV-B.

D. Putting it altogether

We summarize the workflow of PIA depicted in Fig. 1. PIA takes the target buffer level x_r and current buffer level x_t as input, and computes the selected bitrate R_t^*, which is then fed into the Video Player Dynamics block to update the buffer level x_t. PIA considers both present and past estimation errors, as well as incorporates all the three QoE metrics in the control loop. PIA also includes an anti-windup mechanism to deal with bitrate saturation, and a setpoint weighting technique to provide faster initial ramp-up.



(a) (b)

Fig. 2: Characteristics of the network bandwidth traces.

IV. Performance Evaluation

In this section, we evaluate the performance of PIA using simulation and real implementation on a video player. Simulation allows us to evaluate a large set of parameters in a scalable manner. Real implementation provides insights under various system constraints. In both cases, the network conditions are driven by a set of traces captured from commercial LTE networks that allow reproducible runs as well as apple-to-apple comparison of different schemes. We first describe the evaluation setup, and then compare PIA against several state-of-the-art schemes using simulation. Last, we present real implementation results on a DASH player.

A. Evaluation setup

Network bandwidth traces. We focus on LTE networks that dominate today's cellular access technology. For evaluation under realistic LTE network environments, we collected 50 network bandwidth traces from two large commercial LTE networks in the US. These traces were collected under a wide range of settings, including different times of day, different locations (in three U.S. states), and different movement speed (stationary, walking, local driving, and highway driving). Each trace contains 30 minutes of one-second measurement of network bandwidth. The bandwidth was measured as the throughput of a large file downloading from a well provisioned server on a mobile device. Fig. 2(a) is a boxplot that shows the minimum, first quartile, median, third quartile, and maximum bandwidth of each trace, where the traces are sorted by the median bandwidth. We see that the network bandwidth is indeed highly dynamic. For some traces, the maximum bandwidth is tens of Mbps while the minimum bandwidth is less than 10 Kbps.

Video parameters. We use three video bitrate sets: $\mathcal{R}_1 = [0.35, 0.6, 1, 2, 3]$ Mbps, $\mathcal{R}_2 = [0.35, 0.6, 1, 2, 3, 5]$ Mbps and $\mathcal{R}_3 = [0.2, 0.4, 0.6, 1.2, 3.5, 5, 6.5, 8.5]$ Mbps. The first set is based on the reference for YouTube video bitrate levels

(corresponding to 240p, 360p, 480p, 720p and 1080p respectively) [5]. The second set adds a higher bitrate level of 5 Mbps to the first set. The third set is based on Apple's HTTP Live Streaming standard [1]. For each bitrate set, we further consider three variants with chunk size of 2, 4, and 8 s.

ABR Schemes. We compare PIA against three other schemes.

• RB: The bitrate is picked as the maximum possible bitrate that is below the predicted network bandwidth. This is a simple open-loop controller (see Section II) serving as baseline.

• BBA [11]: This is a state-of-the-art buffer based scheme. We use BBA-0 (BBA-1 deals with variable bitrate (VBR) while for simplicity, we use constant bitrate (CBR) in simulation). The lower and upper buffer thresholds are $\theta_{\text{low}} = 10$ s and $\theta_{\text{high}} = 60$ s, respectively. The upper threshold is chosen to accommodate chunk size of 8s (for chunk size of 2 and 4 s, setting it to 30 s leads to similar results). The corresponding buffer size is at most 60 MB (considering the highest bitrate of 8 Mbps), reasonable even on mobile devices. We have empirically verified the above buffering setting works well on our dataset.

• MPC [22]: The video bitrate is chosen by solving a discrete optimization problem by looking ahead for a horizon of 5 chunks (as suggested by the paper).

• PIA: We set the target buffer level $x_r = 60$ s that is compatible with the setting of BBA. The look-ahead horizon is set to 5 chunks (i.e., $L = 5$ in (13)).

For all the schemes, the start-up playback latency δ is set to 5, 10 or 15 s. For BBA and MPC, their parameters are either selected based on the original papers, or configured by us based on the properties of the videos (e.g., chunk size and encoding rates) as justified above.

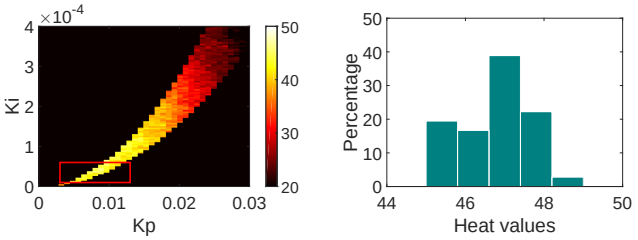
Network bandwidth prediction. For the schemes that require network bandwidth estimation, it is set as the harmonic mean of the network bandwidth of the past 20s. Harmonic mean has been shown to be robust to measurement outliers [12]. Fig. 2(b) shows the boxplot of the bandwidth prediction of the network traces. Each box in the plot corresponds to the distribution of all prediction instances within a particular trace. The prediction is at the beginning of every second. As shown, the median prediction error is 20% to 40%, highlighting the challenges of accurate bandwidth prediction in LTE networks.

B. PIA: Choice of parameters

Following the methodology outlined in Section III-C, we first tune K_p and K_i, and then tune β for PIA. One question is whether there exist a set of K_p and K_i values that work well in a wide range of settings. This is an important issue related to the practicality of PIA because if the choice of K_p and K_i were sensitive to the settings, then tuning them for different settings would require more efforts. While as described earlier, the quality of experience (QoE) is affected by three metrics (average video bitrate, the amount of bitrate changes and rebuffering) jointly, comparing the QoE under different choices of K_p and K_i is much simpler when using a single combined metric. There is no consensus on such a metric. One approach is using a weighted sum of the three metrics as in [22]. Specifically, for a video of N chunks,

$$\text{QoE} = \sum_{t=1}^{N} R_t - \mu \sum_{t=1}^{N-1} |R_{t+1} - R_t| - \lambda \sum_{t=1}^{N} S_t. \qquad (15)$$

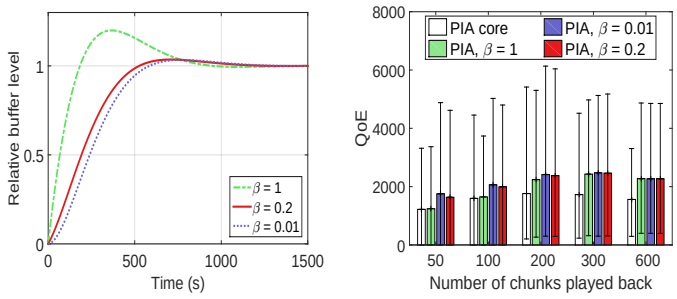
where R_t is the bitrate of the t-th chunk and S_t is the amount of stalls for the t-th chunk, and μ and λ are weights that represent respectively the importance of the middle and last terms (i.e., bitrate changes and rebuffering) relative to the first term (i.e., average bitrate) in the sum. There is no well agreed upon settings for μ and λ; we vary μ and λ to multiple values for sensitivity test.



(a) Heatmap. (b) Histogram of "heat" values.

Fig. 3: Region of K_p and K_i and the corresponding "heat" values in one setting (bitrate level set $\mathcal{R}_3$, chunk size 2s, video length 20 min, startup latency 10s, $\mu = 1$, $\lambda = 8.5$ (when bitrate is represented in the unit of Mbps)).

The trace-driven simulation allows us to conveniently consider a wide range of settings by varying video bitrate level set, video length, chunk size, startup latency, and μ and λ in (15). Specifically, the video bitrate level set is either $\mathcal{R}_1$, $\mathcal{R}_2$, or $\mathcal{R}_3$, video length is 5, 10 or 20 minutes, chunk size is 2, 4 or 8 s, startup latency is 5, 10 or 15 s, and μ is 1 or 2, and λ is the maximum bitrate level of a video (e.g., 3 Mbps in $\mathcal{R}_1$) or twice as much. The choice of of μ and λ is based on the settings in [22]. In each setting (i.e., after fixing the above parameters), we consider each of the 50 network bandwidth traces individually. For the k-th network trace, we vary the values of K_p and K_i in a large range to find the pair of K_p and K_i that maximizes the QoE (note that as described in Section III, we only consider valid combinations of K_p and K_i values, i.e., those so that the damping ratio is in $[0.6, 0.8]$). Once the maximum QoE, denoted as Q_k^*, is determined, the QoE under each valid (K_p, K_i) pair is compared to Q_k^* to see whether it is within 90% of Q_k^*. Specifically, we define a binary function $f_k(K_p, K_i)$ for the k-th network bandwidth trace, where $f_k(K_p, K_i) = 1$ if the resulting QoE under K_p and K_i is within 90% of Q_k^*, and otherwise $f_k(K_p, K_i) = 0$. We then consider all the network bandwidth traces, and create a heat map with the "heat" for each valid pair of K_p and K_i values as $\sum_k f_k(K_p, K_i)$. Clearly, a larger "heat" value for a K_p and K_i pair means that it leads to good performance for more network bandwidth traces. Fig. 3a shows an example heat map for one set of parameters (described in the caption of the figure). The black region represents invalid K_p and K_i pairs (i.e., those causing the damping ratio out of $[0.6, 0.8]$). For the valid K_p and K_i pairs, the "heat" value varies, with



(a) Step response. (b) QoE.

Fig. 4: Choosing β in PIA.

the highest values in the bottom left region, marked by the rectangle. Fig. 3b is the histogram of the "heat" values in the rectangle area (excluding those corresponding to invalid K_p and K_i pairs). It shows that majority of the values are close to 50 (i.e., the maximum "heat"), indicating that the valid K_p and K_i pairs marked by the rectangle provide good performance across almost all network traces.

We repeat the above procedure for all the settings, and find the following region of K_p and K_i values leads to good performance for all the settings

$$\begin{aligned} K_p &\in \left[1 \times 10^{-3}, 14 \times 10^{-3}\right] \\ K_i &\in \left[1 \times 10^{-5}, 6 \times 10^{-5}\right] \\ \text{s.t. } &\zeta\left(K_p, K_i\right) \in [0.6, 0.8]\,. \end{aligned} \tag{16}$$

Specifically, under the above range of values, the average "heat" for the different settings varies from 31 to 50, and the standard deviation varies from 0.25 to 3.45. The results in the rest of the paper use $K_p = 8.8 \times 10^{-3}$, approximately the middle of the range of K_p in (16), and $K_i = 3.6 \times 10^{-5}$ so that the damping ratio is $1/\sqrt{2}$, a widely recommended value for damping ratio [18], [16].

The finding that a small set of K_p and K_i values work well under a wide range of settings is encouraging. Comparing video characteristics (chunk size, bitrate levels) and network conditions, network conditions represent the environment in which PIA operates, and hence play a more important role in determining K_p and K_i values. Considering that the network traces were collected under a wide range of settings and that they exhibit significantly different characteristics (see Fig. 2), our results show that K_p and K_i can be tuned to accommodate the large variations among individual traces. Fundamentally, this indicates that we can find a range of K_p and K_i values to make the system capable of dealing with the rapid bandwidth variations, which are known to be one of the predominant characteristics of cellular networks, despite the differences across individual network conditions.

Once K_p and K_i are determined, we tune β for the initial stage of the video playback. Specifically, we set β to 0.01, 0.2, 0.4, 0.6, 0.8, and 1.0. Fig. 4(a) shows the step response of the control policy (only the results for β=0.01, 0.2 and 1 are shown for better clarity). When β = 1, the buffer becomes full much more quickly than when β=0.2 and 0.01. This is because, as explained in Section III-B, lower bitrate tends to be selected when β=1, causing the buffer to fill up more quickly. To examine the three QoE metrics jointly, Fig. 4(b) plots the QoE when playing up to the i-th chunk of a video of 600 chunks (the setting is the same as that for Fig. 3) when β=0.01, 0.2 or 1. We see that β indeed affects the QoE for the initial playback, and β=1 leads to lower QoE compared to β=0.01 and 0.2. Further investigation reveals that β=0.01 leads to more rebuffering than β=0.2. Results in other settings show similar trends. Since rebuffering has very detrimental effects on viewing quality, we use β=0.2 in the rest of the paper.

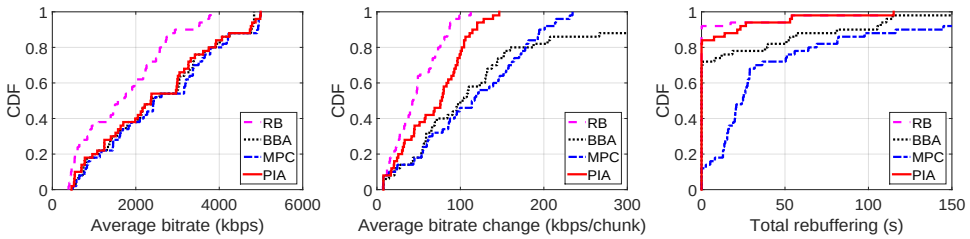
Last, the results of PIA core (i.e., without the three enhancing techniques) are also shown in Fig. 4(b). We see that PIA core indeed leads to lower QoE compared to the full-fledged PIA, indicating the benefits of our three enhancing techniques.

C. Performance comparison

We first present the performance of PIA in the default setting, i.e., chunk size 2 s, video bitrate set $\mathcal{R}_2$, video length 20 minutes, startup latency 10 s. After that, we evaluate the impact of various parameters on the performance of PIA.

Fig. 5 plots the CDF of the three QoE metrics over all network bandwidth traces in the default setting. The performance of four schemes, RB, BBA, MPC and PIA, are also plotted. We see that while the amount of bitrate change and rebuffering is low under RB, its average bitrate is significantly lower than those of the other schemes. PIA achieves comparable average bitrate as BBA and MPC, with significantly less bitrate change and rebuffering. Specifically, the average bitrate of PIA is 98% and 96% of that of BBA and MPC, respectively, while the average amount of bitrate change is 49% and 40% lower, and the average amount of rebuffering is 68% and 85% lower than BBA and MPC, respectively. Overall, *PIA achieves the best balance among the three conflicting metrics*. As described earlier, the inferior performance of RB is because it uses open-loop control without any feedback. The superior performance of PIA compared to BBA is because BBA implicitly uses one form of P-control (Section II) that only takes into account present error, while PIA considers both the present and past errors. PIA's approach of applying PID in an explicit and adaptive manner further facilitates the design and improves the performance. The performance of MPC is sensitive to network bandwidth estimation errors [22]: it solves a discrete optimization problem in each step; when network bandwidth estimation is inaccurate, the input to the optimization problem is correspondingly inaccurate, leading to suboptimal performance.

To provide further insights, Fig. 6 plots the bitrate selection and the buffer level over time for BBA, MPC and PIA when using one network trace. For reference, it also plots the network bandwidth of the trace. We clearly see that BBA has significantly more bitrate changes, and MPC tends to be more aggressive in choosing higher bitrates, which can lead to excessive rebuffering. The bitrate selection under PIA matches well with the network bandwidth without frequent bitrate changes. In terms of the buffer level shown in the



(a) (b) (c)

Fig. 5: Performance comparison in default setting (chunk size 2 s, bitrate set $\mathcal{R}_2$, video length 20 mins, startup latency 10s).

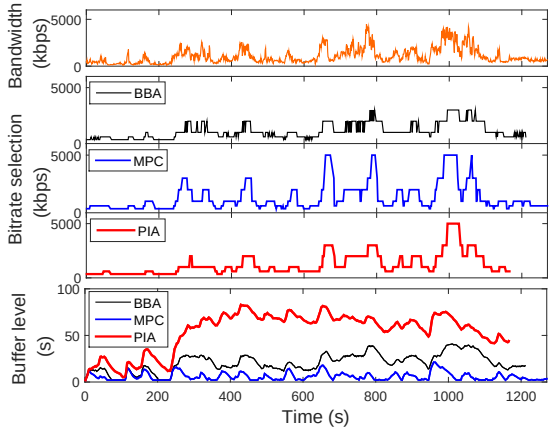


Fig. 6: Comparison of different schemes for one trace under the default setting (chunk size 2 s, video bitrate set $\mathcal{R}_2$, video length 20 minutes, startup latency 10 s).

bottom plot in Fig. 6, MPC is lower than that of BBA and PIA due to its aggressive choice of bitrate; the buffer level of PIA reaches steady state at around 300 s, and then stays around the target level of 60 s; the buffer level of BBA is in between that of MPC and PIA.

Impact of video length. The above results are for video length of 20 mins. We can vary the ending time of the video to investigate PIA's performance for shorter videos. When the ending time is larger than 5 mins (i.e., video length longer than 5 mins), we observe similar results as before; for much shorter videos, PIA has lower average bitrate compared to BBA and MPC (but still outperforms BBA and MPC on the other two metrics). This is because of the initial transient period of PIA to reach the target buffer level. Further improving the performance during the transient period is left as future work.

Impact of video bitrate sets. Recall that the video bitrate set $\mathcal{R}_2$ has one higher bitrate level of 5 Mbps compared to $\mathcal{R}_1$. We further investigate two more video bitrate sets $\mathcal{R}_4 = [0.2, 0.35, 0.6, 1, 2, 3]$ Mbps, which has one lower bitrate of 0.2 Mbps compared to $\mathcal{R}_1$; and $\mathcal{R}_5 = [0.2, 0.35, 0.6, 1, 2, 3, 5]$ Mbps, which has one lower and one higher video bitrate levels (of 0.2 and 5 Mbps) compared to $\mathcal{R}_1$. Fig. 7 shows the results (the results of RB are omitted for better clarity). The X axis is the video bitrate set ID, and each subplot corresponds to a metric. For each metric, we plot the average value over all the network traces with 95% confidence interval. We observe consistent trend for all three schemes (BBA, MPC, and PIA). Comparing the results under $\mathcal{R}_1$ and $\mathcal{R}_2$, we see adding one higher bitrate level leads to higher average video bitrate, more bitrate changes, and more rebuffering; comparing the results under $\mathcal{R}_1$ and $\mathcal{R}_4$, we see adding one lower bitrate level maintains the average video bitrate while reduces bitrate changes and rebuffering; comparing $\mathcal{R}_1$ and $\mathcal{R}_5$, we see adding both one lower and higher bitrate levels increases the average video bitrate and bitrate changes, while reduces the rebuffering. In general, adding more bitrate levels helps improve at least one of the three metrics. Across the settings, PIA has the lowest bitrate switches, the lowest rebuffering, and comparable average bitrate compared to BBA and MPC.

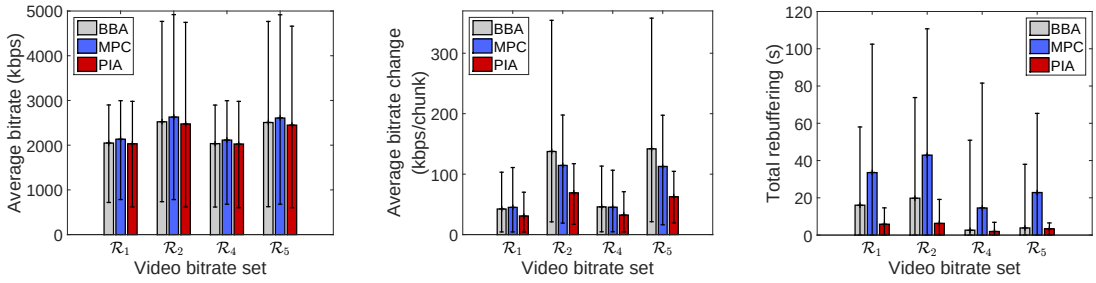
Impact of video chunk size. We vary the video chunk size by setting it to 2, 4, and 8 s. We found that for all chunk sizes, PIA consistently outperforms MPC and BBA in balancing the tradeoffs incurred by the three metrics. For example, for chunk size of 8 s, PIA's average playback bitrate differs from BBA and MPC only by 0.1% and 3.2%, respectively, while PIA reduces the rebuffering duration by 67% and 68% compared to those of BBA and MPC, respectively.

D. Computational overhead

As described earlier, the computational overhead of PIA is much lower than that of MPC: for m bitrate levels and horizon L, the complexity of MPC is $O(m^L)$, while the complexity of PIA is $O(mL)$. On a commodity laptop with Intel i5 2.6GHz CPU and 16GB RAM, for 600 chunks (2-second chunk with bitrate set $\mathcal{R}_2$), the CPU time for MPC is 36.04 s, while the CPU time for PIA is 0.17 s, comparable to that of BBA (0.08s).

E. Evaluation using DASH implementation

We have implemented PIA using `dash.js` (version 2.0), a production quality open source framework [4]. We create an emulation environment that consists of two computers: one is a Linux machine running Apache httpd as the video server and the other is a Windows machine as the client (a laptop with i7-5700HQ 3.50 GHz CPU and 16GB memory). They are connected by a 100Mbps link. We then apply the Linux `tc` tool on the server side to emulate the bandwidth of the download link based on the LTE bandwidth traces we collected. We set the latency between the client and the server to 70ms as it is the average latency reported by OpenSignal's latency report. The client uses Chrome browser to run `dash.js`. Under the `dash.js` framework, we implemented a new ABR streaming rule (about 400 LoC) to realize PIA.



(a) (b) (c)

Fig. 7: Impact of video bitrate levels on performance (chunk size 2 s, video length 20 minutes, startup latency 10 s.)

Comparing Simulation and Real Implementation Results. We compare the results obtained from our `dash.js` implementation with those from the simulation, and confirm that the results are consistent. Specifically, for average bitrate, 90% of the relative differences are within 6.5%; for bitrate changes and and rebuffering duration, 90% of the absolute differences are within 15 Kbps/chunk and 3 s, respectively. Our further investigation indicates that such differences are mainly due to VBR introduced by the x.264 encoder, a setting that differs from our constant bitrate assumption in simulation. The results also indicate PIA works well for VBR videos.

Runtime Overhead. We use a video of 903 seconds encoded using the bitrate set $\mathcal{R}_1$ with the chunk size of 2 s. We record the CPU execution time of the ABR logic in the JavaScript code when the video is being played. The execution time of the default ABR logic in the `dash.js` player is 1.2 s for the entire 15-min video; the execution time of our PIA logic is only slightly longer (1.9 s). The results indicate PIA incurs very small runtime overhead despite its non-trivial decision process shown in Fig. 1.

V. Conclusion and Future Work

By strategically applying PID controller in an explicit and adaptive manner, PIA considerably outperforms state-of-the-art video streaming schemes in balancing the complex tradeoffs incurred by key QoE metrics, as demonstrated by extensive evaluations. PIA is also lightweight and easy to deploy. We believe the same high-level principle can be applied to other multimedia applications with content quality adaptation such as live video conferencing. In our future work, we plan to port our implementation to mobile devices to better assess PIA's performance in the wild, and also conduct deeper exploration of other network and streaming settings.

References

[1] Best Practices for Creating and Deploying HTTP Live Streaming Media for Apple Devices (Apple Technical Note TN2224). https://developer.apple.com/library/content/technotes/tn2224/_index.html.

[2] Cisco VNI: Global Mobile Data Traffic Forecast Update, 2015-2020. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

[3] Ctrix Mobile Analytics Report, 2014. https://www.citrix.com/products/bytemobile-adaptive-traffic-management/tech-info.html#reports.

[4] Dash-Industry-Forum/dash.js. https://github.com/Dash-Industry-Forum/dash.js.

[5] YouTube live encoder settings, bitrates and resolutions. https://support.google.com/youtube/answer/2853702?hl=en.

[6] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[7] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *Proc. of ACM MobiCom*, 2013.

[8] L. D. Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proc. of ACM MMSys*, 2011.

[9] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo. ELASTIC: a client-side controller for dynamic adaptive streaming over HTTP (DASH). In *Proc. of Packet Video Workshop (PV)*. IEEE, 2013.

[10] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. *ACM CCR*, 41(4), 2011.

[11] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. of ACM SIGCOMM*, pages 187–198, 2014.

[12] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with FESTIVE. In *Proc. of ACM CoNEXT*, pages 97–108, 2012.

[13] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. *IEEE/ACM Transactions on Networking*, 21(6):2001–2014, 2013.

[14] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. Begen, and D. Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE JSAC*, 32(4):719–733, 2014.

[15] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. Deriving and validating user experience model for DASH video streaming. *IEEE Transactions on Broadcasting*, 61(4), December 2015.

[16] N. H. McClamroch. *State Models of Dynamic Systems: A Case Study Approach*. Springer, 1980.

[17] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *Proc. of ACM Multimedia*, 2011.

[18] K. Ogata. *Modern Control Engineering*. Prentice Hall, 2010.

[19] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: near-optimal bitrate adaptation for online videos. In *INFOCOM*. IEEE, 2016.

[20] G. Tian and Y. Liu. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proc. of ACM CoNEXT*, 2012.

[21] X. Xie, X. Zhang, S. Kumar, and L. E. Li. piStream: physical layer informed adaptive video streaming over LTE. In *Proc. of ACM MobiCom*, 2015.

[22] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proc. of ACM SIGCOMM*, pages 325–338, 2015.

[23] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha. Can accurate predictions improve video streaming in cellular networks? In *Proc. of HotMobile*, 2015.