Toward a Principled Framework to Design Dynamic Adaptive Streaming Algorithms over HTTP

Xiaoqi Yin Vyas Sekar Bruno Sinopoli
Electrical and Computer Engineering Department, Carnegie Mellon University
Pittsburgh, PA, USA
{xiaoqiy,vsekar}@andrew.cmu.edu, brunos@ece.cmu.edu

ABSTRACT

Client-side bitrate adaptation algorithms play a critical role in delivering a good quality of experience for Internet video. Many studies have shown that current solutions perform suboptimally, and despite the proliferation of several proposals in this space, both from commercial providers and researchers, there is still a distinct lack of clarity and consensus w.r.t. several natural questions: (1) What *objectives* does/should such an algorithm optimize? (2) What environment *signals* such as buffer occupancy or throughput estimates should an algorithm use in its control loop? (3) How sensitive is an algorithm to operating conditions (e.g., bandwidth stability, buffer size, available bitrates)? This work attempts to bring clarity to this discussion by casting adaptive bitrate streaming as a model-based predictive control problem. We demonstrate the initial promise of shedding light on these questions using this control-theoretic abstraction.

Categories and Subject Descriptors
C.2.4 [**Distributed Systems**]: Distributed applications; C.4 [**Performance of systems**]: Modeling techniques

General Terms: Algorithms; Design; Performance

1 Introduction

Many recent studies have highlighted the critical role that user-perceived quality-of-experience (QoE) plays in Internet video applications, which ultimately ties in to revenues for content providers [14, 19]. Specifically, metrics such as the duration of rebuffering (i.e., the player's playout buffer does not have content to render), startup delay (i.e., the lag between the user clicking vs. the time to begin rendering), the average playback bitrate, and the stability of the bitrate delivered have emerged as key factors.

Given that there is little, if any, support in the network for optimizing such measures, bottlenecks could occur at every point in the video ecosystem. For instance, Content Distribution Networks (CDN) servers may be chosen poorly or the data may not be in the content server's cache. Similarly, there may be intermittent or persistent congestion in the ISPs, or potential throttling of video flows. There may even be bottlenecks in the home networks, with the shared WiFi network being a point of contention.

Given this complex ecosystem and presence of diverse bottlenecks, the *bitrate adaptation logic* in the video player is critical to ensure a rich user experience. Thus, delivering high QoE requires intelligent *adaptation* algorithms that can dynamically adapt the quality to the current operating conditions. Historically, video streaming over IP has relied on custom servers and protocols to implement adaptation algorithms. However, the actual widespread deployment of Internet video has taken a different (and rather surprising) trajectory with HTTP being the dominant and converged protocol for delivering video content. The reasons for the departure from custom protocols are pragmatic; e.g., repurposing existing commodity CDN solutions and server deployments and avoiding interactions with middleboxes [24].

This bitrate adaptation logic is also referred to as Dynamic Adaptive Streaming over HTTP (DASH) algorithms.[1] Our specific interest here is in the implementation of *client-side* DASH. At a high-level, DASH approaches work as follows. The video content is divided into chunks and each chunk is encoded at several discrete bitrate values. The DASH problem is to choose the bitrate level for future chunks to deliver the highest possible QoE; e.g., maximizing bitrate while minimizing the likelihood of buffering and avoiding too many bitrate switches. As recent efforts have pointed out, this problem is challenging because the adaptation algorithm has to be robust to network conditions as well as interactions with lower-layer control loops [16].

Recognizing the growing importance of video QoE, several proposals have emerged (e.g., [18, 18, 8, 17]). Despite the proliferation of several algorithms, however, there appears to be a lack of clarity and consensus across these solutions on several fronts; e.g., some argue for better bandwidth estimation [25], while others suggest improving chunk scheduling [18]. Some researchers even argue for avoiding *rate-based* approaches that rely on throughput estimates

[1] In this paper, we use DASH to refer to the class of bitrate adaptation algorithms rather than the specific standard [7].

HotNets-XIII, October 27–28 2014, Los Angeles, CA, USA

http://dx.doi.org/10.1145/2670518.2673877

from previous chunk downloads and make the case for purely *buffer-occupancy based* algorithms [17].

This paper is an attempt to bring rigor to this problem space. To this end, we formulate the DASH problem through the "lens" of *model predictive control* from the domain of control theory. This abstraction provides a general framework to objectively compare different *classes of algorithms* and analyze their sensitivity to operating conditions rather than evaluate specific artifacts.

Using this framework, we present an initial attempt to clarify some fundamental questions about the design space of DASH algorithms:

- What is the objective function that these algorithms try to optimize?
- Are rate-based algorithms fundamentally flawed or can they be competitive w.r.t. buffer-based algorithms?
- Is there an optimal way to combine rate- and buffer-based control approaches?
- How far away from the optimal solutions are pure rate-based or pure buffer-based approaches?
- How sensitive are the algorithms to parameters such as network variability, bitrate levels, and buffer size?

Our model-based results though preliminary are promising to shed light on this problem space. Our initial results suggest that 1) buffer-based (BB) algorithms outperform rate-based (RB) algorithms, however, by systematically combining buffer occupancy and bandwidth predictions, model predictive control (MPC) further drives the performance closer to optimal; 2) the advantage of MPC over BB depends on bandwidth prediction accuracy, in cases where future bandwidths are hard to predict, dropping bandwidth information by adopting pure BB approach can in turn be beneficial; 3) high bitrate variability and finer-grained bitrate levels will increase the benefits of MPC/BB over pure RB algorithms.

We acknowledge several challenges and open questions remain. First, we need to analyze the implementation complexity of such optimal control algorithms and see if they can be practically implemented in client-side players. Second, we do not yet have a good understanding of the bandwidth variability of video clients. Third, we have not fully explored the sensitivity to the full spectrum of operating parameters. Finally, we need to extend this framework to also consider issues of fairness and efficiency when multiple players and applications compete for shared resources.

2 Background and Motivation

We begin with a high-level overview of how HTTP-based adaptive video streaming works, before describing the key challenges and shortcomings of state-of-art solutions today.

Internet video technologies such as Microsoft SmoothStreaming [5], Apple's HLS [22], and Adobe's HDS [1] rely on *HTTP-based* adaptive streaming. We refer to this class of protocols as Dynamic Adaptive Streaming over HTTP or DASH. In DASH systems, each video consists of multiple segments or "chunks" (corresponding to a few seconds

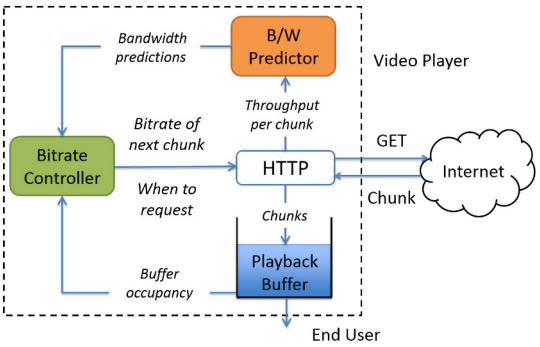


Figure 1: Abstract model of DASH algorithms

of play time) and each chunk is encoded at multiple *discrete* bitrates. The chunks from different bitrate streams are aligned so that the video player can switch to a different bitrate if necessary at a chunk boundary. This approach has several pragmatic advantages over custom streaming protocols such as Real-Time Messaging Protocol (RTMP). The use of HTTP enables providers to seamlessly bypass middleboxes. Furthermore, it can use existing commodity CDN servers without requiring custom modifications. Finally, by making the server stateless, we can implement better load balancing as well as fault tolerance mechanisms using multiple servers and CDNs [21, 20].

Figure 1 shows an abstract model of the adaptive video player. The player uses some inputs (e.g., buffer occupancy or estimates of the network bandwidth) in its decision logic to choose the bitrate level for the next chunk to be downloaded. In making this decision, there are many (and potentially conflicting) quality considerations a player must account for: minimizing rebuffering events where the playback buffer is empty and cannot render the video, deliver as high a playback bitrate as possible within the bandwidth constraints, minimize startup delay so that the user does not quit while waiting for the video to load, and keeping the playback as "smooth" as possible by avoiding frequent or large bitrate jumps [14, 19].

To see why these objectives are conflicting, let us consider two extreme solutions. A trivial solution to minimize rebuffering and the startup delay would be to *always* pick the smallest available bitrate, but it conflicts with the goal of delivering high bitrate. Conversely, picking the highest available bitrate may lead to many rebuffering events. Similarly, the goal of minimizing rebuffering and maintaining a smooth playback may also conflict if the optimal choice w.r.t. buffering and high average bitrate is to switch rapidly.

Many measurement studies have shown the poor performance of state-of-art video players with respect to these QoE measures [10, 18, 16]. These studies show that these problems are not artifacts of specific players but manifest across all state-of-art players such as SmoothStreaming [5], Netflix [4], Adobe OSMF [2], and Akamai HD [3]. For brevity we do not reproduce these results here but refer interested readers to prior work [10, 18, 16].

To alleviate these problems, there have been several recent

proposals in the research literature [25, 18, 9, 17]. At a high level, these solutions can be roughly divided into two categories: (1) *rate-based algorithms* (e.g., [18]) and (2) *buffer-based algorithms* (e.g., [17]). Video players with rate-based methods essentially pick the highest possible bitrate based on the estimated available bandwidth. However, as shown in prior work bandwidth estimation on top of HTTP suffers from significant biases [16], which leads to problems with traditional rate-based approaches. Some solutions try to work around these biases by either smoothing out throughput estimates [25] or choosing better scheduling strategies [18]. On the other hand, recent work makes a case for buffer-based algorithms [17]. Rather than using throughput estimates, this class of algorithms uses buffer occupancy as the feedback signal, and designs mechanisms to keep the buffer occupancy at a desired level, essentially discarding all information coming from the bandwidth estimation.

Despite the broad interest in this topic from academia and industry, what is critically lacking today is a principled understanding of the performance of bitrate adaptation algorithms on several dimensions. Each solution offers "point" heuristics that work in specific environments. While each approach seen in isolation has been shown to outperform the commercial players, there is little effort to systematically compare how different *classes of algorithms* stack up against each other or which of these technical components are *critical* or how robust these algorithms are to different operating regimes (e.g., bandwidth stability, buffer size, number of bitrate levels). Furthermore, many of these algorithms fail to formally state what *objective* they seek to optimize making it harder to conduct a meaningful comparison.

Our motivation in this work is to bring some clarity to this space. Rather than design yet another point solution, we seek to develop a general framework to *reason about classes of algorithms*. To this end, we resort to control-theoretic tools to formally define the control problem and optimization underlying DASH systems as discussed in the next section.

3 Control-Theoretic Model

In this section, we develop a mathematical model of the HTTP video streaming process and formally define the bitrate adaptation problem. This gives us a framework to compare and evaluate existing algorithms, and it also serves as the foundation for potential improvements.

3.1 The Video Streaming Model

We model a video as a set of consecutive *video segments* or chunks, $\mathcal{V} = \{1, 2, \cdots, K\}$, each of which contains L seconds of video and encoded with different bitrates. Thus, the total length of the video is $K \times L$ seconds. The video player can choose to download video segment k with bitrate $R_k \in \mathcal{R}$, where $\mathcal{R}$ is the set of all available bitrate levels. The amount of data in segment k is then $L \times R_k$. The higher bitrate is selected, the higher video quality is perceived by the user. Let $q(\cdot) : \mathcal{R} \to \mathbb{R}_+$ be the function which maps selected bitrate R_k to video quality perceived by user $q(R_k)$. We assume $q(\cdot)$ to be increasing.

The video segments are downloaded into a *playback buffer*, which contains downloaded but as yet unviewed video. Let $B(t) \in [0, B_{max}]$ be the *buffer occupancy* at time t, i.e., the play time of the video remained in the buffer. The *buffer size* B_{max} depends on the policy of the service provider, as well as storage limitations.

At time t_k, the video player starts to download segment k. The downloading time depends on the selected bitrate R_k as well as average download speed C_k. At time t_{k+1}, when segment k is completely downloaded, the video player immediately starts to download the next segment $k + 1$. If we denote by C_t the bandwidth at time t, then we have:

$$t_{k+1} = t_k + \frac{LR_k}{C_k} \tag{1}$$

$$C_k = \frac{1}{t_{k+1} - t_k} \int_{t_k}^{t_{k+1}} C_t \, dt. \tag{2}$$

The buffer occupancy evolves while the video is being downloaded and played. The buffer occupancy increases by L seconds after segment k is downloaded. Meanwhile, after the start-up phase, the buffer occupancy decreases as the user watches the video. The buffer dynamics can then be formulated as:

$$B_{k+1} = \left(B_k - \frac{LR_k}{C_k} \right)_+ + L \tag{3}$$

Here, $B_k = B(t_k)$ and $(x)_+ = \max\{x, 0\}$. Note that if $B_k < \frac{LR_k}{C_k}$, the buffer becomes empty while the video player is still downloading segment k, leading to rebuffering events.

3.2 Goal of Bitrate Adaptation: Maximize QoE

The ultimate goal of bitrate adaptation is to improve the QoE of the users, so as to achieve higher long-term user engagement [14]. While the definition of QoE may differ across users, the key elements of QoE are the following:

1. *Average Video Quality* : $\frac{1}{K} \sum_{k=1}^{K} q(R_k)$;
2. *Average QualityVariations*: $\frac{1}{K} \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)|$;
3. *Total Rebuffer Time*: $\sum_{k=1}^{K} \left(\frac{LR_k}{C_k} - B_k \right)_+$,
   or *Number of Rebufferings*: $\sum_{k=1}^{K} \mathbf{1} \left(\frac{LR_k}{C_k} > B_k \right)$.

For brevity, we assume a fixed startup latency. However, the model can be extended to incorporate the startup delay into the objective function as well.

As users may have different preferences on which of three components is more important to them we define the QoE of video segment 1 through K by a weighted sum of the aforementioned components:

$$QoE_1^K = \sum_{k=1}^{K} q(R_k) - \lambda \sum_{k=1}^{K-1} |q(R_{k+1}) - q(R_k)| - \mu \sum_{k=1}^{K} \left(\frac{LR_k}{C_k} - B_k \right)_+ \tag{4}$$

Here λ, μ are positive weighing parameters corresponding to video quality variations and rebuffering time, respectively. A relatively small λ indicates that the user is not particularly concerned about video quality variability; the large λ is, the more effort is made to achieve smoother changes of bitrates. A large μ, relatively to the other parameters, indicates that a user is deeply concerned about rebuffering. This definition of QoE 1) allows customization so it can easily take into account user's preference, 2) can be extended as needed to incorporate other factors, such as start-up time [14].

We are now ready to formulate the problem of bitrate adaptation for QoE maximization in the following way:

$$\begin{aligned} \max_{R_1,\cdots,R_K} \quad & QoE_1^K \\ s.t. \quad & t_{k+1} = t_k + \frac{LR_k}{C_k}, \\ & C_k = \frac{1}{t_{k+1}-t_k}\int_{t_k}^{t_{k+1}} C_t \, dt, \\ & B_{k+1} = \left(B_k - \frac{LR_k}{C_k}\right)_+ + L, \\ & R_k \in \mathcal{R}, \quad B_k \in [0, B_{max}], \\ & \qquad\qquad \forall k = 1, \cdots, K. \end{aligned} \tag{5}$$

We denote problem (5) as $QoE\_MAX_1^K$. The bandwidth trace $C_t, t \in [t_1, t_{K+1}]$ serves as input to the problem. Outputs of $QoE\_MAX_1^K$ are bitrate decisions $R_1, \cdots, R_K$, along with corresponding downloading time $t_1, \cdots, t_K$ and buffer occupancy $B_1, \cdots, B_K$.

3.3 Classes of Algorithms

In this section we wish to characterize problem (5) and describe existing bitrate adaptation algorithms within this framework to provide better insights on how they relate to one another and allow easier comparison.

Problem (5) is a finite-horizon stochastic optimal control problem. The source of randomness is the bandwidth C_t: At time t_k when the video player chooses bitrate R_k, only the past bandwidth $\{C_t, t \le t_k\}$ is available while the future one $\{C_t, t > t_k\}$ is not known with certainty, as it varies randomly.

However a *bandwidth predictor* can be used to obtain predictions defined as $\{\hat{C}_t, t > t_k\}$. Based on such prediction and on buffer occupancy information (which is instead known precisely), the *bitrate controller* selects bitrate of the next segment k:

$$R_k = f\left(B_k, \{\hat{C}_t, t > t_k\}\right). \tag{6}$$

While the choice of a bandwidth predictor can influence the overall QoE, in this study we solely focus on bitrate adaptation algorithms, and regard bandwidth predictors as given and characterized by their prediction error. Namely, we focus on the design of $f(\cdot)$ and on the effect of the prediction error on the performance of the compared control algorithms.

Different bitrate adaptation algorithms essentially adopt different functions $f(\cdot)$. Two main categories of algorithms appear in the literature: *rate-based (RB)* and *buffer-based (BB)* algorithms.

Video players with RB strategies essentially choose bitrate only based on bandwidth information, i.e.,

$$R_k = f\left(\{\hat{C}_t, t > t_k\}\right). \tag{7}$$

For example, a typical RB strategy is to choose the maximum possible bitrate below the predicted bandwidth.

On the other hand, BB strategies advocate decision making based merely on buffer occupancy, namely:

$$R_k = f\left(B_k\right), \tag{8}$$

while regarding bandwidth variations as unmodeled disturbances. For example, Huang et al. illustrate one roadmap for designing BB algorithms [17].

Note, however, that both algorithms are discarding possibly useful information, and consequently are in principle suboptimal. Ideally, we want to use both buffer occupancy and bandwidth prediction as shown in (6), to enable a broader design space of bitrate adaptation algorithms. Specifically, in this paper we propose to use model predictive control (MPC) [13], also known as receding horizon control or dynamic optimization algorithms to explore this design space. MPC algorithms are widely used to solve similar problems in different domains, ranging from industrial control to navigation. In our context, MPC algorithms essentially choose bitrate R_k by looking h steps ahead, and solve QoE maximization problem $QoE\_MAX_k^{k+h}$ with bandwidth predictions $\{\hat{C}_t, t \in [t_k, t_{k+h}]\}$. The first bitrate R_k is applied by using feedback information and the optimization process is iterated at each step k. The main advantage of MPC is that it incorporates the bandwidth predictions, buffer occupancy and buffer dynamics into the bitrate adaptation process, making use of the available information to maximize QoE in a computationally efficient manner.

3.4 Performance Metric: Normalized QoE

We define a normalized QoE metric to compare the performance of algorithms to the theoretical optimum, which could be achieved if future bandwidth is known.

For a given bandwidth trace $\{C_t, t \in [t_1, t_{K+1}]\}$, the *offline optimal QoE*, denoted by $QoE(OPT)$, is the maximum QoE that can be achieved with perfect knowledge of future bandwidth over the entire time horizon. Technically, it is calculated by solving problem $QoE\_MAX_1^K$. While the assumption of knowing the entire future is not true in reality, the offline solution provides a theoretical upper bound for all algorithms for a particular bandwidth trace.

On the other hand, *online QoE* with bitrate selection algorithm A is calculated under the assumption that at time t_k, the bitrate controller only knows the past bandwidth $\{C_t, t \in [t_1, t_k]\}$, based on which it selects R_k. We denote the online QoE achieved by algorithm A by $QoE(A)$.

Because offline optimal solution assumes perfect knowledge about future, for any algorithms, the online QoE is always less than the offline optimal QoE. In other words, $QoE(OPT)$ is an upper bound of online QoE achieved by any algorithms. To this end, we define *normalized QoE of A (n-QoE(A))* as the performance metric for an algorithm A:

$$n\text{-}QoE(A) = \frac{QoE(A)}{QoE(OPT)} \quad (9)$$

The normalized QoE of a particular algorithm varies with different bandwidth trace cases. Based on this observation, we will, in the following section, compare different algorithms by 1) cdf of normalized QoE, 2) average normalized QoE over all bandwidth trace cases.

4 Preliminary Results

Simulation framework: We evaluate different algorithms using a custom simulation framework. The simulation takes as input a *bandwidth timeseries* and models the video downloading and playback process and the buffer dynamics. At time t_k when the bitrate of segment k is needed, the simulation calls the bitrate controller embedded with the RB/BB/MPC algorithms to get R_k. Given the lack of recent measurements that systematically study bandwidth stability, we use a simplistic synthetic bandwidth trace generator where the bandwidth is based on some hidden state $S_t \in \mathcal{S}$ capturing number of users sharing a bottleneck link. The actual bandwidth C_t follows a Gaussian distribution with mean $m(S_t)$ and variance $\sigma^2(S_t)$, which both depend on the hidden state S_t. We vary both how the state changes as well as the $m(\cdot)$, $\sigma(\cdot)$ to generate many synthetic traces.

We assume the video length is 5 minutes, consisting of 150 $2s$ chunks. As a baseline, we assume the video is encoded in the following bitrate levels: $\mathcal{R} = \{$400Kbps, 750Kbps, 1000Kbps, 2500Kbps, 4500Kbps$\}$ based on public data from YouTube [6]. We set the buffer size to $B_{max} = 30s$. We specify QoE by setting $\lambda = 1$, $\mu = 3000$. We assume that the video starts to play at $t = 10s$, which normalizes the start-up delay for all algorithms.

Algorithms: Ideally, we want to compare the optimal RB, BB and MPC algorithms. However, determining the optimal algorithm within each class is difficult as it involves optimizing over an infinite-dimensional functional space. To this end, we choose a widely adopted function form for each class of algorithms from prior work, and optimize the free parameters by empirical simulations based on randomly generated bandwidth traces. Specifically, for RB, we pick the maximum available bitrate below p times predicted bandwidth, and calculate the best p through offline simulations. Similarly, for BB, we employ the function suggested by Huang et al [17]. Here R_k is chosen to be the maximum available bitrate below r_k, where r_k is defined :

$$r_k = \begin{cases} R_{min} & B_k < r \\ R_{min} + \frac{B_k - r}{c}(R_{max} - R_{min}) & r \le B_k \le r + c \\ R_{max} & B_k > r + c \end{cases}$$

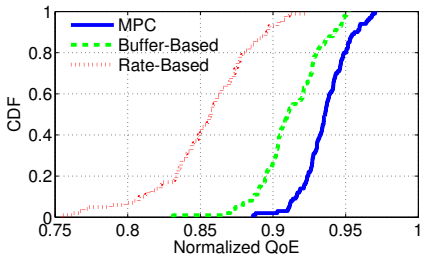


Figure 2: Cdf of normalized QoE

with r and c as free parameters. For MPC algorithms, we set the look-ahead horizon h as 5 segments as a baseline.

For RB and MPC, we assume the bandwidth predictions are true bandwidth plus zero-mean Gaussian noise with standard deviation σ_{err}=10% of actual bandwidth as baseline, and test algorithms with different prediction error levels.

Finally, we compute the offline optimal solution with perfect knowledge of the bandwidth trace. To make it tractable to compute this offline optimal, we assume it can pick bitrates from a continuous range [400Kbps, 4500Kbps].

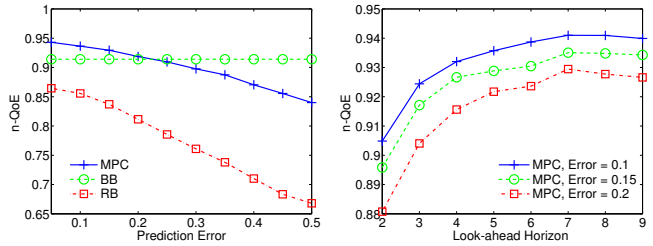
4.1 Comparison of Algorithms

Baseline results: Figure 2 shows the CDF of normalized QoE of the three algorithms over 100 simulation runs with the baseline setup. There are three main observations. First, this result confirms the hypothesis from prior work that BB is better than RB. Second, it also suggests that there is still room for improvement for BB as there is a 5% gap w.r.t. MPC. Finally, we observe that MPC is quite close to the optimal achievable QoE with the median value being 94% of optimal. Next, we evaluate sensitivity of these results to key parameters such as prediction accuracy, QoE weights, and the look-ahead horizon.

Impact of prediction error: Figure 3a shows how the bandwidth prediction errors influence the performance of bitrate adaptation algorithms. As expected BB is unaffected as it does not use any bandwidth information. When bandwidth predictions are accurate, MPC has larger advantage over BB algorithms. As prediction error grows beyond 25%, MPC can be even worse than BB. This suggests that if the actual prediction error is very large, then the video player should drop RB or MPC and use pure BB algorithms. One natural question for future work is to determine this tolerance level at which the control should shift to BB mode.

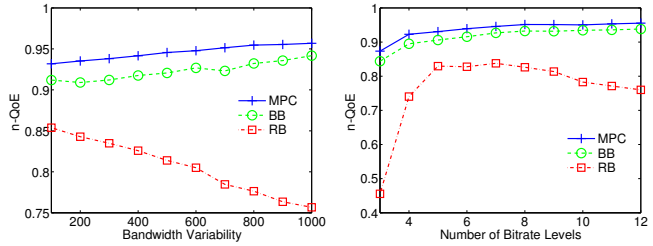
Impact of look-ahead horizon: Figure 3b answers the question of how far we should look ahead in MPC to achieve best performance. As the look-ahead horizon increases, more information of future bandwidth is taken into consideration, leading to increased MPC performance. However, since prediction accuracy reduces as we look farther into the future, the performance of MPC can drop if the horizon is too large.

Impact of QoE weights: We also studied how the QoE parameters λ and μ influence the performances of the algorithms (not shown). At a high level, we find that the normalized QoE of MPC algorithm remains much more stable than



(a) Prediction error (b) Look-ahead horizon

Figure 3: Performance of MPC vs. different configuration factors



(a) Bandwidth variability (b) Bitrate levels

Figure 4: Sensitivity to operating parameters

RB and BB as λ and μ changes. In particular, as we increase the weight on stability (i.e., as λ increases), the advantage of MPC over BB increases and BB over RB also increases. In terms of μ, both MPC and BB achieves near-zero rebuffering, their performance is unaffected whereas RB's optimality suffers significantly when we weigh rebuffering more.

4.2 Sensitivity to Operating Conditions

Figure 4a shows the sensitivity to bandwidth variability. While the performances of MPC and BB remain stable, RB algorithms suffer greatly from bandwidth variations. Because RB approaches try to closely track bandwidth changes, they incur large bitrate variations as well as rebuffer time as bandwidth becomes more variable.

Finally, Figure 4b shows how number of bitrate levels influences the algorithms. As video players with BB and MPC can select from a finer-grained set of bitrate levels, they achieve better and better performance. On the other hand, the performance of RB first grows with number of bitrate levels. However, when there are too many bitrate levels, the performance drops because this allows RB to change bitrate more frequently, leading to increased bitrate instability.

5 Discussion

> *Essentially, all models are wrong, but some are useful. [12]*

While our work provides a systematic method to objectively evaluate different classes of DASH that was critically lacking today, we acknowledge several modeling limitations and open questions for future work.

Full-spectrum sensitivity analysis: Even though we study a wide range of parameters, we acknowledge that this is still not comprehensive. For instance, we also need to analyze how other operating parameters such as buffer size, startup delay and startup bitrate impact our findings.

Bandwidth estimation and interaction with TCP: We currently assume a bandwidth estimator is given and characterized by estimation error. In practice, a real bandwidth estimator must be developed in order for MPC to be applied. Furthermore, our current model assumes the prediction error to be an independent process. However, as observed empirically in prior work [16], the error in predicting the available bandwidth might depend on the recent history of the bitrates of past segments due to the interaction between TCP and HTTP. An open question is to systematically characterize this relationship and how this coupling influences the design of bitrate adaptation algorithm.

Multi-player interactions: This study only modeled the QoE maximization problem for single player, and thus regards the bandwidth variations as fixed externalities. However, in the case of multiple video players competing for a bottleneck link, fairness among players becomes an objective in addition to single player QoE maximization, which is not yet formally defined. Given that there is a diversity of video players and providers, an interesting direction for future work is to study pairwise interactions across RB-, BB-, and MPC-based players. Similarly, another open question is whether we can suggest guidelines analogous to TCP-friendliness for MPC players.

Computational complexity: While RB and BB need relatively minor computations, the main limitation of MPC is its high computational complexity because it involves solving an optimization problem at each step. For video players with less computing power, such as mobile phone apps, we need to develop lightweight algorithms. Here, we believe that fast MPC algorithms are a promising alternative [26].

Characterizing bandwidth stability: While there is a rich history of Internet measurement to understand various properties of Internet paths [23, 28, 27], they do not specifically target video workloads. There are two key distinguishing characteristics of video that are relevant here. First, the key metric that impacts the control logic is not the available bandwidth or the capacity, but the *stability* of the bandwidth. Second, the specific target servers are not general end-hosts but CDN servers hosting videos. In fact, there has been surprisingly little work on understanding bandwidth stability—the three closest related works we are aware of focusing on the stability of Internet path properties focus largely on non-video workloads and are quite dated [15, 27, 11]. An interesting direction for future work is to develop a measurement-driven understanding of available bandwidth stability and predictability of clients to video servers to inform the design of the above control-theoretic models.

6 Acknowledgement

This work was supported in part by the National Science Foundation under Grant ECCS 0925964.

7 References

[1] Adobe http dynamic streaming. www.adobe.com/products/hds-dynamic-streaming.html.

[2] Adobe osmf player. http://www.osmf.org.

[3] Akamai hd network. www.akamai.com/hdnetwork.

[4] Netflix. www.netflix.com/.

[5] Smoothstreaming protocol. http://go.microsoft.com/?linkid=9682896.

[6] Youtube bitrate levels. https://support.google.com/youtube/answer/2853702?hl=en.

[7] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE Multimedia*, 2011.

[8] S. Akhshabi, L. Anantakrishnan, C. Dovrolis, and A. C. Begen. What Happens when HTTP Adaptive Streaming Players Compete for Bandwidth? In *Proc. NOSSDAV*, 2012.

[9] S. Akhshabi, L. Ananthakrishnan, A. Begen, and C. Dovrolis. Server-Based Traffic Shaping for Stabilizing Oscillating Adaptive Streaming Players. In *Proc. ACM SIGMM NOSSDAV*, 2013.

[10] S. Akhshabi, A. Begen, and C. Dovrolis. An Experimental Evaluation of Rate Adaptation Algorithms in Adaptive Streaming over HTTP. In *Proc. MMSys*, 2011.

[11] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing Stability in WideArea Network Performance. In *Proc. ACM SIGMETRICS*, 1997.

[12] G. E. P. Box and N. R. Draper. Empirical Model-Building and Response Surfaces, p. 424, Wiley. ISBN 0471810339.

[13] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer, 2013.

[14] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. A. Joseph, A. Ganjam, J. Zhan, and H. Zhang. Understanding the impact of video quality on user engagement. In *Proc. SIGCOMM*, 2011.

[15] Q. He, C. Dovrolis, and M. Ammar. On the predictability of large transfer TCP throughput. In *Proc. ACM SIGCOMM*, 2005.

[16] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proc. IMC*, 2012.

[17] 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. ACM SIGCOMM*, 2014.

[18] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proc. CoNext*, 2012.

[19] S. S. Krishnan and R. K. Sitaraman. Video Stream Quality Impacts Viewer Behavior: Inferring Causality using Quasi-Experimental Designs. In *Proc. IMC*, 2012.

[20] H. Liu, Y. Wang, Y. R. Yang, A. Tian, and H. Wang. Optimizing Cost and Performance for Content Multihoming. In *Proc. SIGCOMM*, 2012.

[21] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A Case for a Coordinated Internet Video Control Plane. In *SIGCOMM*, 2012.

[22] R. Pantos. Http live streaming. 2011.

[23] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM ToN*, 5(5):601–615, 1997.

[24] L. Popa, A. Ghodsi, and I. Stoica. HTTP as the narrow waist of the future internet. In *Proc. HotNets*, 2010.

[25] G. Tian and Y. Li. Towards Agile and Smooth Video Adaption in Dynamic HTTP Streaming . In *Proc. CoNext*, 2012.

[26] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, 2010.

[27] Y. Zhang and N. Duffield. On the constancy of Internet path properties. In *IMW*, 2001.

[28] Y. Zhang, V. Paxson, S. Shenker, and L. Breslau. The stationarity of Internet path properties: Routing, loss, and throughput. Technical report, ACIRI Technical report, 2000.