Video Through a Crystal Ball: Effect of Bandwidth Prediction Quality on Adaptive Streaming in Mobile Environments

Tarun Mangla*, Nawanol Theera-Ampornpunt†, Mostafa Ammar*, Ellen Zegura*, Saurabh Bagchi†

*Georgia Institute of Technology
{tmangla3, ammar, ewz}@cc.gatech.edu

†Purdue University
{ntheeraa, sbagchi}@purdue.edu

ABSTRACT

Mobile environments are characterized by rapidly fluctuating bandwidth and intermittent connectivity. Existing video streaming algorithms can perform poorly in such network conditions because of their reactive adaptation approach. Recent efforts suggest that bitrate adaptation using proactive accurate bandwidth prediction can help improve the quality of experience (QoE) of video streaming. However, highly accurate long-term predictions may be needed in mobile environments and those can be difficult to obtain. In this work, we examine the impact of bandwidth prediction quality on the QoE. We first characterize bandwidth profiles where bandwidth prediction-based adaptation can be useful. We then study the impact of prediction horizon and errors on the performance of Adaptive Bitrate (ABR) streaming. We observe that performance improves as the prediction horizon increases at first and then benefits start to diminish. We demonstrate that with proper error mitigation heuristic, even erroneous predictions can be useful in some scenarios. Finally, we study the role of video system parameters, namely buffer size and bitrate granularity on bandwidth prediction-based adaptation.

1. INTRODUCTION

Video streaming makes up a major portion of Internet traffic [1]. Clients stream video on a diverse set of devices and under a variety of network conditions. To support this diversity, most content providers such as Youtube, Hulu, Netflix use HTTP-based Adaptive bitrate (ABR) streaming. ABR streaming allows clients to have uninterrupted streaming by adapting the video quality to the changes in network bandwidth. The adaptation is made possible by dividing the video into chunks of constant length and then encoding each chunk into multiple bitrates at the server. Bitrate adaptation logic built into the client video player decides the quality of chunks to request from the server based on its estimate of the future bandwidth and/or playout buffer occupancy. Such adaptation logic aims to optimize (sometimes conflicting) Quality of Experience (QoE) metrics, namely average bitrate, rebuffering and bitrate switches.

Many bitrate adaptation algorithms have been proposed that adapt either based on past TCP-throughput (e.g., [5, 6]), or current buffer occupancy (e.g., [4]). These adaptation algorithms use information from the past or present to estimate the future bandwidth. This makes their approach reactive in the sense that they have to *experience* the fluctuation in the bandwidth in order to *estimate* it and then *adapt* to it. This kind of reactive adaptation can fail when the network conditions are fluctuating rapidly and the past is not a good indicator of future. Mobile clients connected to WiFi or cellular network are specifically characterized by such rapid fluctuations in the network bandwidth. For example, Figure 1a shows the bandwidth measured by a mobile client connected to the Georgia Tech campus WiFi while riding on a campus bus. The bandwidth trace is characterized by intermittent connectivity and, low and rapidly fluctuating bandwidth. To contrast, Figure 1b shows the bandwidth trace of a stationary client connected to the home WiFi of one of the authors. The trace has high average bandwidth, no disconnection and low bandwidth fluctuation. While a reactive adaptation approach can work well for stationary clients, the same approach, as will be shown in experiments later, can lead to high number of bitrate changes and excessive rebuffering in mobile environments.

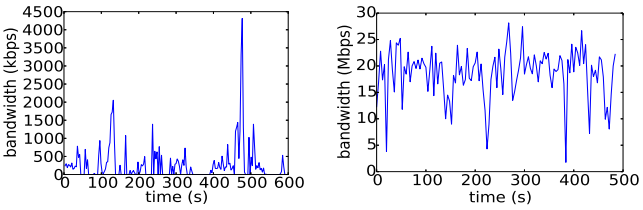
A recent study suggests that *accurate* bandwidth prediction can be useful to improve the performance of ABR streaming [16]. The study shows the gains in QoE achieved by using accurate, short time-horizon, bandwidth predictions for cellular networks. By implication, this study also argues that accurate, longer time-horizon bandwidth prediction will also be even more useful – especially for mobile clients that can experience high bandwidth fluctuation. However, long time-horizon predictions can be difficult to obtain with high accuracy and will be subject to errors. Motivated by this fact, in this paper, we ask the following question: "*How does bandwidth prediction quality, namely accuracy and horizon, impact the performance of prediction-aware ABR streaming*". Building an accurate bandwidth prediction system has an associated cost and hence we also ask "*Under what bandwidth profiles is it most useful to predict bandwidth?*" Finally, we study the role of video system parameters, specifically, bitrate granularity and buffer size on prediction-aware adaptation.

Our work here is not based on any specific bandwidth prediction technique. Rather we are interested in understanding the tradeoffs between prediction quality and ABR performance improvement so as to inform the design of bandwidth prediction techniques aimed at ABR video streaming. Prior work has suggested ways for bandwidth estimation. A cross-layer approach that utilizes information in the PHY layer can be used for bandwidth estimation in cellular networks [7]. Location-based network quality prediction for wireless-LANs has been suggested in [10]. The work in [13] uses real-time cellular data at the base station to predict the quality of the network in the future. Scheduling at the base station based on mobility [9] and shaping video traffic at the the network layer [3, 8] are also approaches that can be indirectly used to predict the future network bandwidth.

In this paper, we first design a bandwidth prediction-aware adaptation algorithm which we dub *CrystalBall*. We also design an error mitigation heuristic that alleviates the QoE degradation due to errors in the prediction. We decided not to use existing prediction-based adaptation algorithms in [16, 15], because the former only uses an average future bandwidth and does not explicitly consider the variations in bandwidth prediction, and the latter requires sig-

nificant offline optimization which increases as the prediction horizon and number of bitrate levels increase. Prediction-based adaptation using physical layer information [14] or location [12] in cellular networks has been also proposed, but these adaptation algorithms have been designed for the specific nature of the prediction and do not work with generic prediction schemes.

The rest of this paper is organized as follows: Section 2 describes the context in which prediction-aware algorithm operates and discusses a generic model for bandwidth prediction quality. Section 3 describes the CrystalBall algorithm including an enhancement to mitigate the effect of errors in the prediction. Section 4 describes results from a set of experiments aimed at understanding the interaction among prediction quality, video system parameters and ABR performance metrics. Section 5 summarizes our effort and conclusions.



(a) Bandwidth of client on a bus connected to campus WiFi

(b) Bandwidth of static client connected to home WiFi

Figure 1: comparison of throughput for static and mobile clients

2. SYSTEM CONTEXT

System Architecture: We suggest an architecture similar to one in Figure 2 where a bandwidth prediction system is feeding the bandwidth estimates to the adaptation algorithm. The bandwidth prediction system can be co-located with client, server or network.

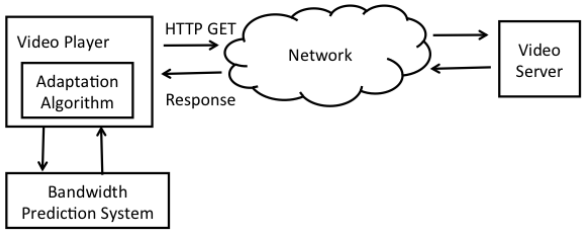


Figure 2: Prediction-based adaptation system

Prediction Model: We assume that the prediction system provides bandwidth estimates at time t_{cur} as the vector, $\hat{A}[i]$, for i $\in [t_{cur}, t_{cur} + W]$. Here, t_{cur} is the current time and W is the prediction *window*. Each prediction value is an average for a period of n seconds, with the vector predicting bandwidth for the next W seconds.

We quantify the prediction error $\xi[i]$ for sample i in the prediction vector as the difference between the actual value and the estimated value. Specifically, $\xi[i] = \hat{A}[i] - A[i]$ where $A[i]$ is the actual bandwidth average over the corresponding interval. We will assume $\xi[i]$ is randomly distributed for each i with increasing variance the farther away from t_{cur} the prediction is.

Video Model: We assume video **V** constituted of M chunks, each of length L seconds. Let B_{max} be the maximum buffer size and b_{cur} be the current buffer occupancy. **R** denotes the set of bitrates and r_k denotes the bitrate at which chunk k was streamed.

QoE Metrics: We use the following quality metrics to access the performance of streaming: average bitrate, rebuffering ratio and bitrate switches. Average bitrate, measured in kbps, is the time average of chunk bitrates during the entire playback session. Rebuffering ratio is the proportion of time video buffered out of the total time. Bitrate switches is the number of times the bitrate changed between two consecutive chunks. A high QoE is characterized by high average bitrate, low rebuffering ratio and low bitrate switches.

3. THE CRYSTALBALL ALGORITHM

3.1 Overview

The CrystalBall algorithm takes into account bandwidth prediction, the current buffer occupancy, the set of available bitrates and the number of chunks to be downloaded as input. It produces the bitrate the player should request future chunk downloads. The algorithm is run each time the bandwidth predictions are refreshed.

The algorithm solves the bitrate adaptation problem by considering the following factors:

- Rebuffering: Prior studies have shown that users are most sensitive to rebuffering in the video [2]. We avoid rebuffering by downloading every chunk before its playback time. As long as the average bandwidth available is not lower than the lowest bitrate, (and the bandwidth prediction is accurate), it is always possible to avoid rebuffering.
- Bitrate: The algorithm aims to maximize the minimum (max-min) bitrate across all chunks. We did not consider maximizing the average bitrate as it can lead to logically worse QoE. For example, a bitrate schedule decision of (in kbps) {1000, 1000, 1000} is better than {2000, 1000, 500}, though the latter has higher average bitrate.

We also would like the number of bitrate switches to be low, as they can be annoying to the user. Although this factor is not explicitly considered in the base algorithm, maximizing the minimum bitrate has the desirable side effect of producing solutions with lower number of bitrate switches. This number could be further lowered by the use of stability heuristics, which are designed to mitigate errors in bandwidth forecasts.

To solve for max-min bitrate, an optimization problem can be formulated with the same constraints as described in [16] and our max-min objective function. However, solving this optimization for every chunk download may not be feasible for a mobile client. A simple greedy approach for the max-min problem then could be to download the next N chunks at the largest bitrate less than the average bandwidth in the prediction window. This greedy approach, however, can lead to excessive rebuffering due to variations in network bandwidth. Below we describe a heuristic that accounts for these variations and gives an exact solution to the max-min problem when there is no limit on the buffer and a sub-optimal solution otherwise.

3.2 The Base Algorithm – *Clear CrystalBall (CCB)*

We first describe a base algorithm (shown in Algorithm 1) that assumes accurate bandwidth predictions. A heuristic to mitigate errors in prediction is described in the next subsection.

CCB first initializes a list $\psi_{[1,N]}$ with N slots, one slot corresponding to each chunk. Each slot is characterized by $(nc, bw)_i$, where nc refers to the number of chunks that will be downloaded in that slot and bw refers to the average bandwidth available for these nc chunks. *GetChunkBW* does this initialization by setting nc as 1 for every slot. For the first chunk, bw is set to be the total bandwidth between t_{cur} and its playback time t_{cur}+b_{cur}, where b_{cur} is the current playout buffer occupancy (measured in seconds of video). This is because we can afford a delay of b_{cur} seconds

while downloading the first chunk. For chunk i with playback time within the next W seconds, bw is the total bandwidth between the playback time of chunk $i-1$ and i. For the remaining slots, bw is set to be equal to zero. These are the chunks that do not have their playback deadline in the next W seconds, but we need to download them to maintain download rate equal or more than the playback rate. After initializing ψ, the algorithm iterates over this list and compares consecutive bw values. The algorithm utilizes the fact that bandwidth at time t can potentially be used for any chunk with playback time greater than t[1]. So, if the bandwidth in the current slot is greater than the next slot, then the two slots can be merged. *GetNewAverage* function merges the two slots into a single slot, by replacing bw with the weighted average of bw_i and bw_{i+1} and nc by the sum of nc values in the two slots. The algorithm terminates when there are no more merges possible i.e. $\psi[i][bw] < \psi[j][bw]$ $\forall i < j$. Then *AssignBitrate* greedily selects the highest bitrate just less than bw corresponding to each chunk and returns a bitrate list $r_{[1,N]}$ which is then used by the player to download the next chunk.

Algorithm 1: Base CrystalBall Algorithm

```
Input : Â_[t,t+W], b_cur, R, n
Output: r_[1,N]
ψ_[1,n] = GetChunkBW(Â_[t,t+W], n, b_cur);
modified = true;
φ = [];
while modified do
    modified = false;
    φ.append(ψ[0]);
    j = 0 ;
    for i = 1; i < len(ψ); i++ do
        if ψ[i][bw] ≥ φ[j][bw] then
            φ[j] = GetNewAverage(φ[j], ψ[i]);
            modified = true;
        else
            φ.append(ψ[i]);
            j++;
    ψ = φ;
    φ = [];
r_[1,n] = AssignBitrate(ψ,R);
return r_[1,n]
```

3.3 Error Mitigation Heuristic – *Foggy CrystalBall (FCB)*

Prediction errors can be of two types: overestimation error or underestimation error. Constant overestimation can lead to the wrong bitrate switch-up decisions and rebuffering. Similarly, constant underestimation can lead to wrong switch-down decisions and under-utilization of network bandwidth. To mitigate the errors we introduce a heuristic which gets called whenever there is a decision to switch the bitrate. The idea behind the heuristic is that a switch could happen because of errors in the prediction and under some conditions we can afford or choose not to switch.

A switch-up decision could be because of over-estimation and the heuristic is to check that the average bandwidth over the prediction window is sufficiently greater than the bitrate video player will switch to. The bitrate is, therefore, switched up only if the following condition is satisfied:

$$avg(\hat{A}_{[1,W]}) \geq (1+\alpha)R_k \quad (1)$$

α is a function of the overestimation error and its value increases as the overestimation error increases.

Similarly, a switch down decision can be because of underestimation, and we could stay at the same bitrate if the current buffer is greater than a threshold. The bitrate is switched down only if the following condition is satisfied:

$$b_{cur} \leq \beta * B_{max} \quad (2)$$

β is a function of the underestimation error and decreases when the underestimation error increases.

Both α and β can be dynamically adjusted by the prediction system based on the global error or by the client video player based on the past errors in the prediction.

4. EVALUATION

Our evaluation is driven by these questions:

- When are predictions useful?
- What is the effect of prediction quality on performance?
- Do video system parameters play any role in prediction-based adaptation?

Before answering these questions, we first describe our experimental methodology.

4.1 Evaluation Strategy

Video parameters and metrics: We assume 10 minute (M = 150) long video sessions in our experiments. The size of video chunk is 4s and maximum buffer size is 32s. We use the following six bitrate levels (in kbps): {150, 350, 600, 1000, 2000, 3000}. To quantify the performance of the adaptation algorithms, we use three video QoE metrics as defined in Section 2 namely, average bitrate, rebuffering ratio and total bitrate switches.

Adaptation algorithms: We implemented a simulator in python for the adaptive video player. We implement three other adaptation algorithms along with our Crystball algorithms. The first is a rate-based adaptive (RBA) player that decides the next chunk bitrate based on the harmonic mean of past 5 chunk download rates. The second player is a buffer-based adaptive (BBA) player where the next bitrate is a function of current buffer occupancy [4]. The third is the prediction-based adaptation player as suggested in [16], referred to as PBA.

Dataset: We use two sets of bandwidth traces: The first is a simple synthetic dataset designed to give us qualitative insight into how bandwidth profiles influence the effectiveness of adaptation-based algorithms. The second set contains bandwidth traces (40 traces) reported every 3s, collected by us when riding the campus bus while being connected to the campus wifi. Each trace in the second set is 15 min long. The trace in Figure 1a is from this second dataset.

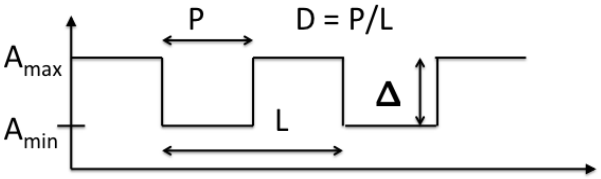


Figure 3: A rectangular waveform bandwidth trace

[1]In case of limited buffer, the bandwidth at t can be used only for chunk with playback time, $t_p \in$ (t,t+B_{max})

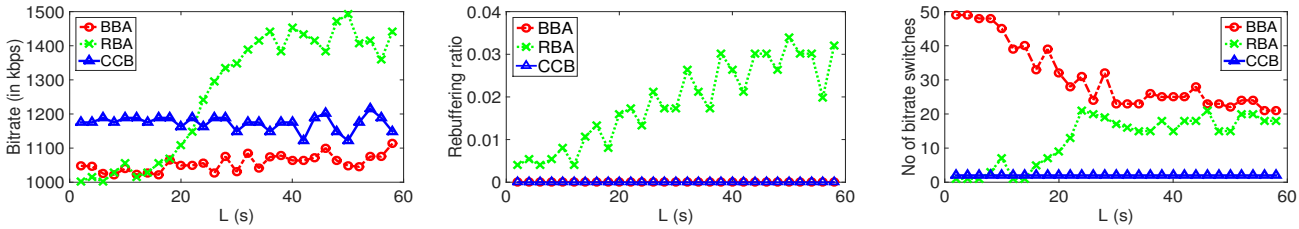


Figure 4: Performance as frequency of fluctuations changes

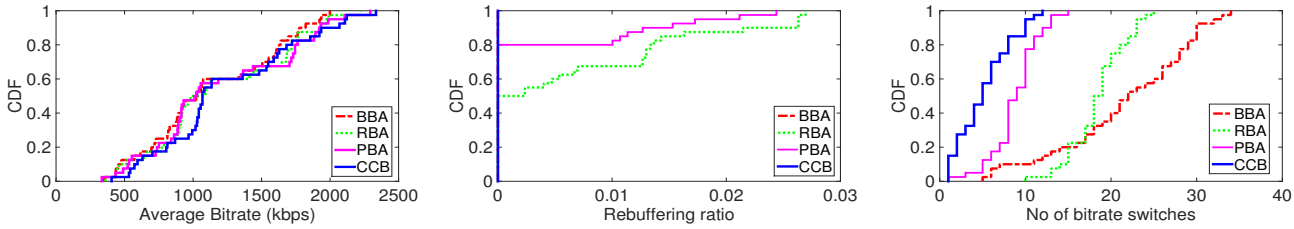


Figure 5: Performance for campus-wifi traces

4.2 When are predictions useful?

Here we qualitatively identify bandwidth profiles that can benefit from prediction-based adaptation. In order to simplify our analysis we focus on a synthetic bandwidth profile that allows us to change some basic properties of bandwidth fluctuations using some parameters. To that end we use a rectangular waveform, shown in Figure 3 as a representative of fluctuating bandwidth profile. A_{max} and A_{min} represent the maximum and minimum values of the bandwidth respectively and Δ denotes their difference. L refers to the period of the waveform and D is the proportion of time when bandwidth is A_{max}. We observe the response of the adaptation algorithms to the bandwidth traces generated by varying different parameters of this waveform. Unless specified otherwise, we use the following values: A_{min} = 0 kbps, A_{max} = 3000 kbps, L = 30s and D = 0.5. We assume that accurate bandwidth predictions are available for 60s horizon at a granularity of 1s.

Amplitude of fluctuations: In this experiment we vary Δ while keeping the sum of A_{min} and A_{max} to be constant and equal to 3000. Figure 6 shows the performance as the amplitude of the fluctuations increases. The rate-based adaptation algorithm performs poorly as the value of Δ increases. For buffer-based adaptation the average bitrate is high with no rebuffering. However, we see a large number of switches (15-50), especially when the bandwidth is constant. This is due to convergence issues with buffer-based adaptation algorithm. Due to discrete set of bitrates, the buffer keeps oscillating between two values and hence the bitrate also keeps switching. The high number of bitrate switches can adversely effect the QoE. CCB is able to adjust to the increasing amplitude of fluctuation.

Duration of low connectivity: Mobile environments often have periods of no or very low connectivty. Here we want to study the effect of duration of low connectivity. We vary the value of D and as shown in Figure 7, BBA again has a high number of bitrate switches and the rate-based adaptation algorithm encounters rebuffering for longer periods of disconnection. However, CCB avoids rebuffering by prefetching the chunks while keeping bitrate switching very low.

Frequency of fluctuations: Here we vary the time period of fluctuation, keeping D = 0.5, A_{min} = 500, A_{max} = 2500. As evident from Figure 4, very high frequency (low time period) fluctuations get averaged out and even reactive adaptation works well. However, fluctuations which have a time period greater than the chunk size lead to degradation in the QoE for rate-based adaptation. This is because, rate-based adaptation have inherent latency in adapting to the fluctuation due to its reactive nature. CCB is able to plan the bitrates taking into account the fluctuation and thus performs well across all QoE metrics.

The above experiments lead to the following insights about the usefulness of bandwidth prediction:

- Predictions can be useful when there are high amplitude fluctuations, as unnecessary bitrate changes can be avoided by proper bitrate planning.
- Predictions can be also be useful in case of intermittent connectivity, as rebuffering can now be avoided by prefetching the content.
- Very high frequency fluctuations can be handled well by reactive ABR schemes that are not prediction-based as they get averaged out. However, bandwidth fluctuations at time scale of chunk size can only be handled by prediction-based adaptation.

Performance on real traces: Here we compare the performance of adaptation algorithms with the campus-wifi traces. Figure 5 plots the CDF of the per-session values of QoE metrics under different adaptation algorithms. The results show similar trends to the synthetic traces. RBA suffers from rebuffering in 50% of the traces and BBA has high bitrate switches, more than 20 for 60% of the traces. Even PBA suffers from rebuffering in 20% of the traces because it does not consider the variations in the prediction and adapts based on the average. The CCB adaptation algorithm performs well as it completely avoids rebuffering, minimizes bitrate switches while maintaining comparable average bitrate.

4.3 What is the effect of prediction quality on performance?

Here we study the impact of prediction quality namely prediction window and accuracy on the QoE of video streaming. The results shown in this section are for campus-wifi dataset traces. Unless specified otherwise, we assume a prediction window of 60s.

Prediction window: We first examine the impact of prediction window on the QoE. Figure 8 plots the average of quality metrics across all traces vs prediction window. It is clear that as the pre-

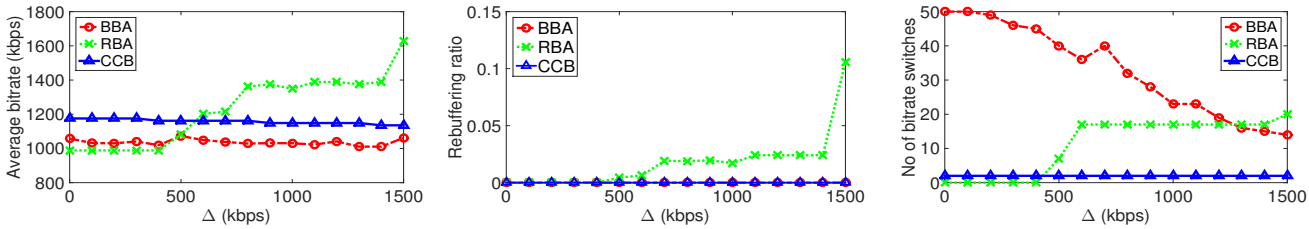


Figure 6: Performance as the amplitude of fluctuations changes

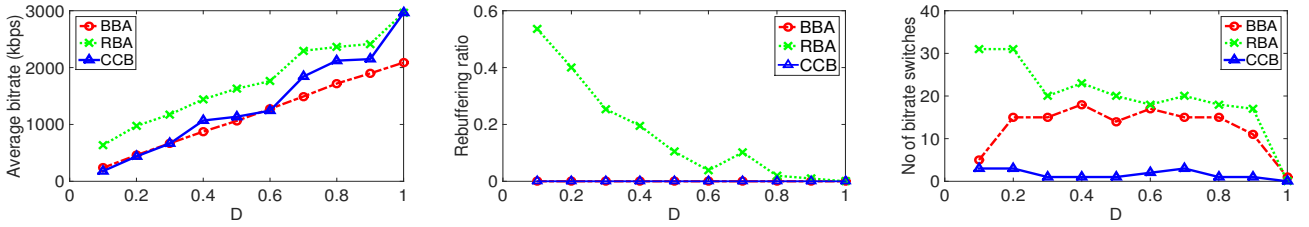


Figure 7: Performance as the duration of fluctuations changes

diction window increases, both rebuffering and bitrate changes decrease. The average bitrate also goes down, but this is because the player avoids unnecessary switch-ups due to temporary increase in the bandwidth. The benefits of prediction diminish after the window becomes more than 80s.

Errors: Until now we have been using accurate predictions, in this experiment we add errors in bandwidth predictions. Ideally, the nature of errors would depend on the bandwidth prediction system. In our experiments we consider an error model with time-varying errors inspired from the errors in weather prediction [11]. The model is based on the intuition that errors in prediction are more for farther times into the future. We first assign whether the current prediction is an overestimation (positive error) or underestimation (negative error) by a coin flip. Then we assign value to the errors that are distributed uniformly randomly between 0 to $\xi_{max}(t)$. Here $\xi_{max}(t)$ grows linearly with time t at a rate of m per second, starting from c. Thus the near future prediction values will be more likely to have less errors as compared to those farther in the future. The errors are recomputed using same process every time a new set of predictions are generated.

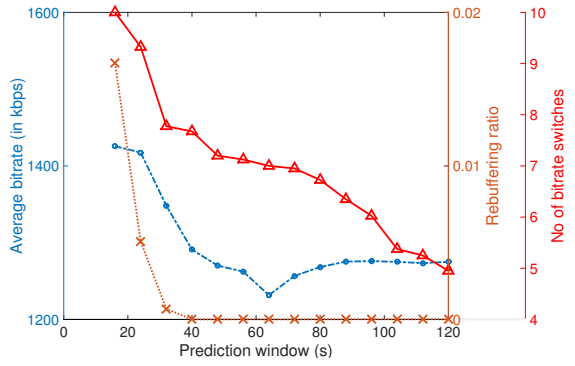


Figure 8: Effect of Prediction Window on CBA

In this experiment, we examine the performance of the error mitigation heuristic. We set m as 10kbps and c as 25kbps. We plot the results for four different scenarios, PBA without errors, PBA with errors, CCB without errors and FCB with errors. In these experiments, we assume α = 0.4 and β = 0.6 for the error mitigation heuristic used in FCB. In a real setting, either prediction system could recommend these values to the client or the client could adapt these values based on past errors in prediction. Figure 9 shows that performance of PBA degrades with errors. However, FCB (CCB with mitigation heuristic) almost behaves as good as CCB without error, despite inaccurate predictions.

In the second set of experiments we show the trade-off between accuracy and longer prediction window. We set the error growth rate (m) to 20kbps and plot the QoE under FCB for three different prediction windows, 16s, 32s and 60s. Figure 10 shows that QoE metrics for W=32 are better than W=16 and W=60. This is intuitive, as larger prediction window means more chances of errors in the prediction. A smaller prediction window, although has less errors, also has less information about the future. Thus, there is a trade-off between these two prediction quality metrics and any prediction system should consider this trade-off while making a bandwidth prediction.

4.4 Role of Video system parameters

Now we compare the role of video system parameters, namely buffer size and bitrate granularity in prediction-aware adaptation with rate-based adaptation. We run our experiments on campus-wifi traces with a 60s prediction window and note the average of QoE metrics across all traces.

Buffer size: Figure 11a shows the impact of varying the maximum buffer size on the QoE for RBA and CCB. For CCB, as the maximum buffer size increases, the number of switches decreases and the average bitrate increases. This is because larger buffers can absorb more fluctuations in the bandwidth and thus avoid unnecessary switch downs. Note that RBA is not able to fully utilize the potential of increasing buffer size as it does not consider buffer occupancy during adaptation. Although the bitrate increases a little, the number of bitrate switches and rebuffering (not shown here but non zero) remain the same.

Bitrate Levels: Here we compare the effect of granularity of bitrate levels on RBA and CCB. We construct a set of equally spaced bitrates between 300kbps and 3000kbps (both included) and vary the number of bitrate levels. Figure 11b shows the rebuffering ratio as the number of bitrate levels increases for RBA and CCB. Interestingly, rebuffering ratio increases for RBA as the number of

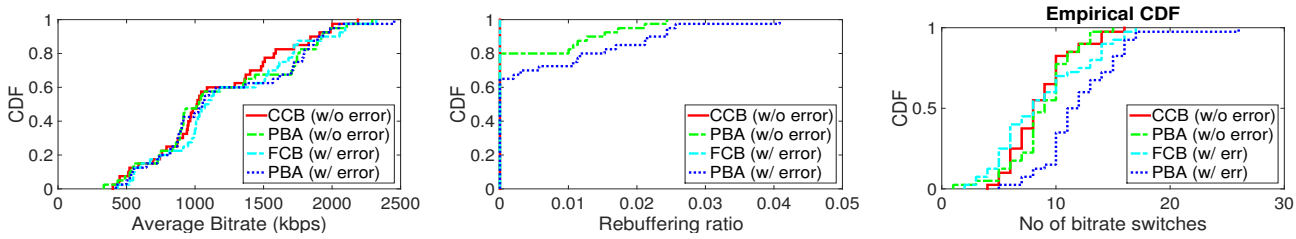


Figure 9: Performance of error mitigation heuristic

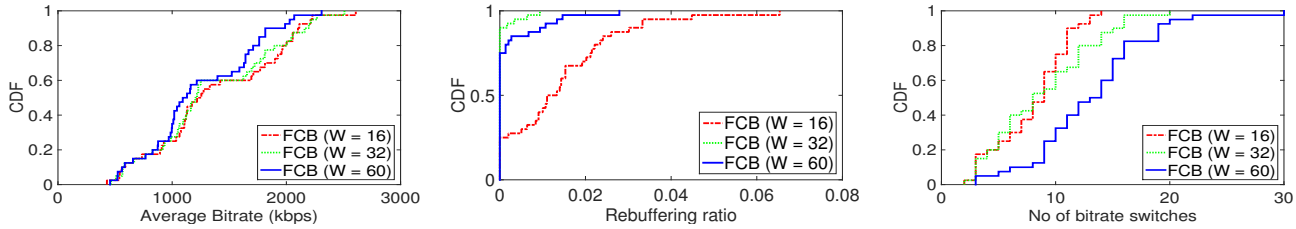
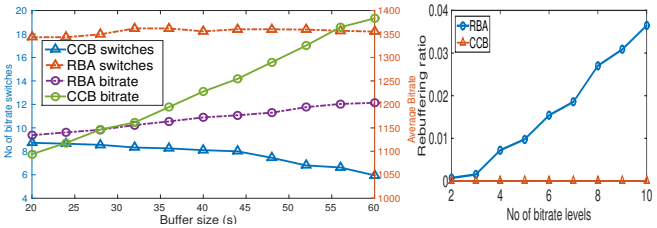


Figure 10: Trade-off between Accuracy and prediction horizon, W is the length of prediction window



(a) Buffer size (b) Bitrate granularity

Figure 11: Effect of video system parameters on QoE

bitrate levels increases. This is because now the bitrate selected by rate-based adaptation will closely match the bandwidth and there will be less video in the buffer at any time. During periods of disconnection, this can lead to rebuffering. The average bitrate and number of switches increase with increase in bitrate granularity for both adaptation approaches (not shown because of limited space).

5. CONCLUSION

In this paper we study the usefulness of bandwidth prediction in improving the performance of ABR video streaming in mobile environments. Prediction quality is characterized by the sometimes conflicting properties of accuracy and time-horizon. Our aim is to inform the design of prediction techniques that specifically target ABR video streaming. In order to make full use of longer-time horizon predictions and to help mitigate prediction errors, we design the CrystalBall (CB) adaptation algorithm. We then conduct an evaluation of the performance of CB and how it compares with other related algorithms in the literature. We observe that performance improves as the prediction horizon increases at first and then benefits start to diminish. We demonstrate that under some scenarios predictions up to some error can be useful with proper error mitigation heuristic. We also find that QoE is affected in unique ways when using prediction-based algorithms. Our future research will use the insights from this study to design bandwidth prediction techniques for mobile users that are specifically tailored to video streaming.

References

[1] Cisco visual networking index: Global mobile data traffic forecast update 2014–2019 white paper, feb 2015. *See:http://tinyurl.com/mokcut3*.

[2] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang. Developing a predictive model of quality of experience for internet video. In *SIGCOMM '13*.

[3] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *MobiCom '13*.

[4] 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. *ACM SIGCOMM Computer Communication Review*, 2015.

[5] J. Jiang, V. Sekar, and H. Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE. In *CoNEXT '12*.

[6] 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*, 2014.

[7] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis. CQIC: Revisiting cross-layer congestion control for cellular networks. In *HotMobile '15*.

[8] A. Mansy, M. Fayed, and M. Ammar. Network-layer fairness for adaptive video streams. In *IFIP Networking, 2015*.

[9] R. Margolies, A. Sridharan, V. Aggarwal, R. Jana, N. Shankaranarayanan, V. Vaishampayan, and G. Zussman. Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms. In *INFOCOM, 2014*.

[10] A. J. Nicholson and B. D. Noble. Breadcrumbs: Forecasting mobile connectivity. In *MobiCom '08*.

[11] D. Orrell, L. Smith, J. Barkmeijer, and T. N. Palmer. Model error in weather forecasting. *Nonlinear Processes in Geophysics*, 2001.

[12] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM TOMM*, 2012.

[13] N. Theera-Ampornpunt, S. Bagchi, K. R. Joshi, and R. K. Panta. Using big data for more dependability: A cellular network tale. In *HotDep '13*.

[14] X. Xie, X. Zhang, S. Kumar, and L. E. Li. piStream: Physical layer informed adaptive video streaming over LTE. In *MobiCom '15*.

[15] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *SIGCOMM '15*.

[16] 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 *HotMobile '15*.