QoE-Aware Adaptive Bitrate Video Streaming over Mobile Networks with Caching Proxy

Kai Dong, Jun He, and Wei Song
Faculty of Computer Science
University of New Brunswick
Fredericton, NB, Canada
Emails: {kdong, jhe2, wsong}@unb.ca

***Abstract*—As a widely used over-the-top (OTT) video technique, adaptive bitrate video streaming has attracted considerable research attention and efforts. Traditional bitrate adaptation schemes rely on quality of service (QoS) to assess video delivery quality. However, in this paper, we argue that QoS may not accurately reflect the user-perceived video quality, especially in time-varying wireless networks. Instead, we adopt the concept of quality of experience (QoE) and propose a novel caching-based adaptation scheme to improve the overall QoE, which can more comprehensively gauge the subjective user satisfaction of video quality. Experiments show that the proposed method outperforms existing adaptation schemes in terms of QoE in various network scenarios. This observation is mainly attributed to the merits of our scheme in properly leveraging channel bandwidth prediction and proxy-based content prefetching.**

***Index Terms*—Adaptive bitrate streaming, quality of experience (QoE), prefetching, bitrate adaptation, wireless networks**

I. Introduction

The development of mobile networks and the Internet has led to the growing popularity of over-the-top (OTT) video services, in which video content is delivered over the Internet. The most common video delivery uses the hypertext transfer protocol (HTTP) from conventional Web servers given the benefits of fast deployment and network address translation (NAT) traversals. Adaptive bitrate streaming (ABS) is an effective technique to improve HTTP-based video delivery by exploiting network dynamics, especially in wireless networks. ABS has been implemented in many OTT video services, such as Adobe's HTTP Dynamic Streaming, Apple's HTTP Live Streaming, and Microsoft's Smooth Streaming.

Fig. 1 shows the ABS structure for a wireless network. As seen, video content is stored at the remote server and provides playback to the mobile device through the mobile network, which further includes the inter-connected backbone network and access point (AP). For adaptation purpose, each video files is fragmented into a series of segments of the same playout duration (typically 2s to 10s) in the remote server. Each segment is further encoded into different quality levels (bitrates) and resolutions. Mobile clients access ABS video information, such as quality levels, URL links to segments, codecs, and resolutions, by requesting a manifest file from the server before transmission. Clients then requests video segments in a chronological order during the transmission. To provide users with the best available video quality in corresponding network condition, ABS intends to determine the quality level of each segment by adopting bitrate adaptation.

This research was supported in part by Atlantic Canada Opportunities Agency (ACOA) and Mariner Partners.

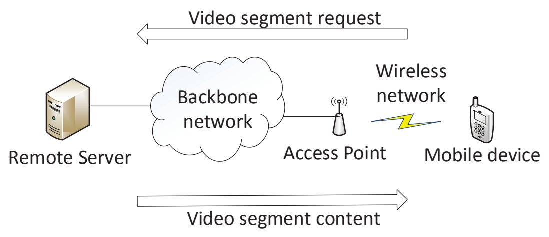

Fig. 1. Structure of adaptive bitrate streaming over wireless networks.

Many bitrate adaptation schemes have been developed to improve the quality of service (QoS) of ABS [1]–[5]. These schemes usually work at mobile clients and estimate current network condition by calculating the average throughput during past segments of the video session associated with the corresponding client, e.g., the throughput between the remote server and the mobile device in Fig. 1 w.r.t. an individual video session. However, the transmission bottleneck of existing mobile networks mainly resides in the radio access networks, i.e., from APs to mobile devices. Therefore, the throughput history record of past segments may not be able to accurately reflect the dynamics of wireless channels. In addition, most existing works on bitrate adaptation focus on the traditional QoS, which is not sufficient to gauge user-perceived service quality. In contrast, quality of experience (QoE) provides a more comprehensive assessment of user-perceived quality.

In this paper, we propose a novel bitrate adaptation scheme based on the prediction of wireless channels at APs. Caching proxy is further deployed at APs to make better use of fast backbone networks and mitigate the impact of the wireless channel variation by prefetching video segments to be requested. From clients' perspective, such a caching mechanism effectively decreases the segment delivery time, and therefore, results in higher average segment quality and lower video freeze levels, compared to traditional adaptation schemes which directly fetch segments from remote servers. In this manner, we can improve the QoE of video services.

The rest of this paper is organized as follows. Section II introduces some existing bitrate adaptation schemes and analyzes the challenges of improving QoE for adaptive streaming in wireless networks. Section III presents the proposed proxy-based bitrate adaptation method. In Section IV, we evaluate and compare the performance of the proposed method to several representative schemes. Section V concludes this paper.

II. Related Works

In this section, we review some conventional bitrate adaptation schemes with the aim to improve the QoS of video streaming. A smooth adaptation scheme (SA) was proposed and evaluated in [1] for different players. The scheme estimates current network condition from historical records as follows:

$$\widehat{A}(t_i) = \begin{cases} \delta \widehat{A}(t_{i-1}) + (1-\delta) A(t_i) & i > 0 \\ A(t_0) & i = 0 \end{cases} \tag{1}$$

where $\widehat{A}(t_{i-1})$ is the bitrate requested for the $(i-1)^{th}$ segment, $A(t_i)$ is the throughput of the i^{th} segment, and δ is the smoothing weight. To achieve smooth bitrate switching under dynamic network conditions, it tries to adapt bitrates to the network condition and meanwhile decrease the standard deviation of video quality level so as to resist traffic jitters. However, due to the smoothing effect, the bitrate may not be able to quickly adapt to network changes. In fact, experiments therein show that this scheme reacts slowly towards bandwidth changes, which, we comment that, can decrease the amount of data in client buffer and hence result in video freezes.

An instant adaptation (IA) scheme was proposed in [2], which solves the problem of the smooth scheme in [1] by adjusting the weight in (1). The weight approaches 1 when the network condition is significantly changed, so that the client can react quickly. Otherwise, it is set to a small value to provide smooth adaptation. Compared to the smooth adaptation scheme, such instant throughput based scheme reacts more quickly to the network changes and decreases the probability of video freezes. However, the standard deviation of the video quality level increases as a consequence, especially in wireless networks with time-varying channel conditions.

Buffer based adaptation (BBA) schemes [3], [4] use the amount of data in the client buffer, instead of the segment throughput, as the metric for bitrate switching. Higher segment bitrates are chosen as the amount of data in corresponding client buffer is large, and vice versa. In [3], a list of buffer thresholds are set to trigger decisions of different bitrates. In [4], the client buffer occupancy is mapped into the segment bitrate via a algorithm-defined function. Such buffer based bitrate adaptation schemes focus on the requirement of the client buffer since a large buffer is needed to accommodate data variation in the buffer due to bitrate adaptation.

The work in [6] unveiled that video freezes have the largest impact on user engagement. In order to decrease freezes, [5] imitates the congestion control mechanism in the transport control protocol (TCP) and proposes a TCP-like conservative bitrate adaptation (CA) scheme: once the preceding segment is transmitted for a time longer than the estimated time, the bitrates of the subsequent segments are first dropped to the lowest bitrate and then increased slowly. This TCP-like bitrate adaptation scheme acts well in avoiding video freezes, but at the cost of the low average quality level and high standard deviation of the quality level.

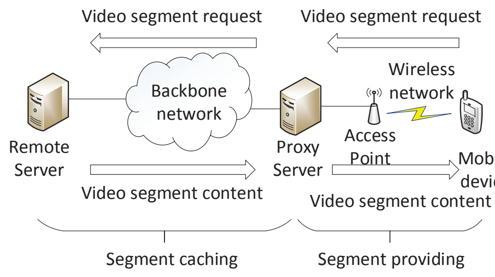

Fig. 2. Structure of our proposed adaptive bitrate streaming.

Aforementioned adaptation schemes focus on improving traditional QoS instead of the overall user-perceived quality which can only be assessed by taking into account all important QoS metrics in a QoE measure. However, existing bitrate adaptation schemes fail to guarantee a high level for all factors of QoE at the same time, especially in wireless networks with unstable network channels. For example, consider the three metrics incorporated in the QoE model proposed in [7], i.e., *the average quality level*, *the standard deviation of the quality level*, and *the duration and frequency of video freezes*. A high average quality level requires that the selected bitrate should keep close to the available bandwidth, while it also implies increased standard deviation of quality level when the channel data rate changes significantly. The probability and duration of video freezes will also be high since the transmitted segments tend to be large and delayed. Likewise, a low probability and duration of video freezes require a low bitrate for fast delivery of segments. As a consequence, the average quality level will be low. Similarly, a low standard deviation of quality level means that the bitrate should be switched in a smooth way, which also increases the probability and duration of video freezes in return, as analyzed above.

III. Proposed Caching-based Adaptive Bitrate Streaming Scheme

A. System Model

In order to improve the overall QoE of video streaming, we target a cache-enabled system depicted in Fig. 2. While video content is stored in remote servers, proxy servers are deployed at APs. The proxy is a transparent temporary caching component, which fetches video segments from the remote server to the edge of the backbone and serves the client with corresponding segments upon specific requests. The proxy strategically determines the bitrates of the requested segments based on estimation of the traffic condition of the backbone network as well as the channel status of the wireless network. If a requested segment with the determined bitrate is buffered in the memory of the caching proxy, it will directly serve the client with its local copy; otherwise, it needs to fetch the segment from the remote server before forwarding it to the client. The AP is able to capture the variation and make prediction of wireless channels. The caching proxy uses the prediction to decide what bitrates and segments to be cached so as to improve the cache hit ratio [8], i.e., the probability that the prefetched bitrate of a segment fits the actual wireless channel condition in the real time when the segment is requested.

B. Bitrate Adaptation

Our cache-based bitrate adaptation scheme consists of two parts: a segment prefetching algorithm and a segment providing algorithm. The segment prefetching algorithm determines the bitrate(s) of the segment(s) to be cached. The segment providing algorithm decides which bitrate of the requested segment to be selected to respond corresponding request, either from the caching proxy or from the remote server.

The segment prefetching algorithm is summarized in Algorithm 1. For the first segment, the proxy caches only its lowest bitrate. To prefetch the k^{th} segment(s), the caching proxy compares the current bandwidth of the backbone network, W and the data rate of the wireless channel, R. The wireless channel can be modeled as a finite-state Markov channel (FSMC) [9], for which a distinct data rate is achievable for each state. If $W \geq 2R$, the proxy estimates two most possible states of the wireless channel in the near future and obtains two smoothed throughput values λ_1 and λ_2 according to Eq. (1). We note that the weights δ_1 and δ_2 for calculating the smoothed throughput depend on the variation of channel conditions. In the case with $W \geq 2R$, the two largest bitrates are selected that are smaller than $m_1\lambda_1$ and $m_1\lambda_2$, separately, for prefetching the corresponding segments. Here, m_1 is a parameter to allow a margin between the network capacity and the segment bitrate. On the other hand, if $W < 2R$, only one bitrate is to be prefetched for each segment. The selection of margin values here also depends on the network condition. The margin value is given by m_2 and m_3 respectively for cases with $W \geq R$ and these with $W < R$, where $m_2 > m_3$.

The segment providing algorithm is listed in Algorithm 2. For the first segment, the proxy serves the requested segment with the lowest bitrate. In the subsequent segment delivery, the proxy chooses a smaller value of the backbone network bandwidth W and the data rate of the wireless channel R, as the current network capacity C. A measured throughput λ will be obtained by smoothing C with the previous measured throughput λ_0. If the k^{th} segment(s) have already been prefetched, the segment with the largest bitrate smaller than λ will be provided to the client. On the contrary, if the k^{th} segment is not prefetched, or the smallest bitrate of the prefetched segments is larger than λ, the proxy will skip its local cache and request the remote server for the segment of the largest bitrate that is smaller than λ.

Algorithm 1 The segment prefetching algorithm.

Input: W: backbone network bandwidth; R: wireless channel data rate; λ_0: throughput measured in the transmission of the last segment; $\mathbb{L} = \{r_0, r_1, \ldots, r_{N-1}\}$: set of N available bitrates for video segments

Output: $\mathbb{B}_k$: set of bitrate(s) of prefetched k^{th} segment(s)

```
1:  if $k = 1$ then
2:      $\mathbb{B}_k \leftarrow \{r_0\}$    // Select the lowest bitrate
3:      $\lambda_0 \leftarrow 0$
4:      return $\mathbb{B}_k$
5:  end if
6:  if $W \geq 2R$ then
        // Find data rates of two most possible channel states
7:      $\{R_1, R_2\} \leftarrow FSMC(R, 2)$
8:      if $R_1 \geq \lambda_0$ then
9:          $\lambda_1 \leftarrow \delta_1 R_1 + (1-\delta_1)\lambda_0$  // Smoothed throughput
10:     else
11:         $\lambda_1 \leftarrow \delta_2 R_1 + (1-\delta_2)\lambda_0$  // Smoothed throughput
12:     end if
        // Choose the largest affordable bitrate
13:     $b_1 = \max\{r_i | r_i \in \mathbb{L}, r_i \leq m_1\lambda_1\}$
14:     if $R_2 \geq \lambda_0$ then
15:         $\lambda_2 \leftarrow \delta_1 R_2 + (1-\delta_1)\lambda_0$
16:     else
17:         $\lambda_2 \leftarrow \delta_2 R_2 + (1-\delta_2)\lambda_0$
18:     end if
19:     $b_2 = \max\{r_i | r_i \in \mathbb{L}, r_i \leq m_1\lambda_2\}$
20:     $\mathbb{B}_k \leftarrow \{b_1, b_2\}$
21:     return $\mathbb{B}_k$
22: else
        // Find the data rate of the most possible channel state
23:     $\{R_1\} \leftarrow FSMC(R, 1)$
24:     if $R_1 \geq \lambda_0$ then
25:         $\lambda_1 \leftarrow \delta_1 R_1 + (1-\delta_1)\lambda_0$
26:     else
27:         $\lambda_1 \leftarrow \delta_2 R_1 + (1-\delta_2)\lambda_0$
28:     end if
29:     if $W \geq R$ then
30:         $b_1 = \max\{r_i | r_i \in \mathbb{L}, r_i \leq m_2\lambda_1\}$
31:     else
32:         $b_1 = \max\{r_i | r_i \in \mathbb{L}, r_i \leq m_3\lambda_1\}$
33:     end if
34:     $\mathbb{B}_k \leftarrow \{b_1\}$
35:     return $\mathbb{B}_k$
36: end if
```

IV. Performance Evaluation

In this section, we carry out a series of experiments to evaluate the performance of the proposed cache-based adaptation scheme, termed as CBA henceforth. Four existing bitrate adaptation schemes [1]–[3], [5] are also simulated to align with our scheme under different network conditions.

A. Experimental Setup

We simulate a system shown in Fig. 3. A and B are two Windows machines in one local area network. An Apache server runs on machine A, working as the remote server. Two Java programs on machine B simulate the caching proxy and the client, respectively. A 4-state FSMC model is implemented

Algorithm 2 The segment providing algorithm.

Input: W: backbone network bandwidth; R: wireless channel data rate; λ_0: throughput measured in the transmission of the last segment; $\mathbb{L} = \{r_0, r_1, \ldots, r_{N-1}\}$: set of N available bitrates for video segments; $\mathbb{B}_k$: set of bitrate(s) of prefetched k^{th} segment(s)

Output: S_k: k^{th} segment

```
1:  if $k = 1$ then
2:      $\lambda_0 \leftarrow 0$
3:      $S_k \leftarrow s_0$   // Return segment of lowest bitrate
4:      return $S_k$
5:  end if
6:  $C \leftarrow \min\{W, R\}$
7:  if $\lambda_0 = 0$ then
8:      $\lambda \leftarrow C$
9:  else
10:     if $C \geq \lambda_0$ then
11:         $\lambda \leftarrow \delta_1 C + (1 - \delta_1)\lambda_0$
12:     else
13:         $\lambda \leftarrow \delta_2 C + (1 - \delta_2)\lambda_0$
14:     end if
15: end if
16: $\lambda_0 \leftarrow \lambda$;
17: if $\mathbb{B}_k \neq \phi$ and $\min\{\mathbb{B}_k\} \leq \lambda$ then
        // $s_j$ is the segment of rate $b_j$
18:     $S_k = \max\{s_j | b_j \in \mathbb{B}_k, b_j \leq \lambda, 1 \leq j \leq 2\}$
19:     return $S_k$
20: else
21:     $b_{new} = \max\{r_i | r_i \in \mathbb{L}, r_i \leq \lambda\}$
22:     Get segment $S_k$ of bitrate $b_{new}$ from remote server
23:     return $S_k$
24: end if
```

on machine B to simulate the wireless channel between the proxy and the client. Corresponding to states 1 to 4, the data rates supported by the wireless channel are 500 kbps, 1 Mbps, 2 Mbps, and 4 Mbps, respectively. We use this model to generate trace files that characterize variations of the channel conditions. During the experiment, the Java programs read the trace files to simulate the current wireless channel state and predict the next state according to the model. A bandwidth shaping tool "NetLimiter" [10] running on machine B helps to regulate the bandwidth from A to B, which simulates the network dynamics between the remote server and the AP.

In the experiment, we use a video trace named "Red Bull Playstreets" from [11], which is about 97-minute long. The video is then partitioned into 2622 segments of 2s long. The dataset provides 17 quality levels with bitrates 100, 150, 200, 250, 300, 400, 500, 700, 900, 1200, 1500, 2000, 2500, 3000, 4000, 5000, and 6000 in kbps.

B. QoE Model

To compare the performance of the bitrate adaptation schemes, we use the QoE model in [7] to map these objective performance factors to mean opinion score (MOS), which is a five-point scale representing different levels of user experience, i.e., *Excellent* (5), *Good* (4), *Fair* (3), *Poor* (2), and *Bad* (1). Specifically, user QoE is evaluated by

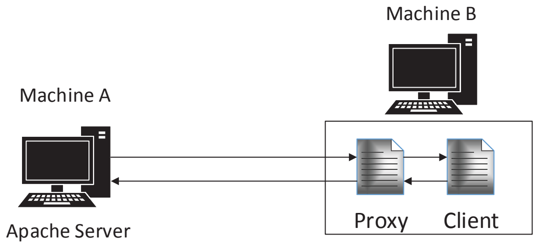

Fig. 3. The framework of the simulation system.

$$eMOS = \max(5.67 \cdot \mu - 6.72 \cdot \sigma - 4.95 \cdot \phi + 0.17, 0) \quad (2)$$

$$\mu = \frac{\sum_{k=1}^{K} \frac{Q_k}{N}}{K} \quad (3)$$

$$\sigma = \sqrt{\frac{\sum_{k=1}^{K} (\frac{Q_k}{N} - \mu)^2}{K - 1}} \quad (4)$$

$$\phi = \frac{7 \cdot \max(\frac{\ln(F_{freq})}{6} + 1, 0) + (\frac{\min(F_{avg}, 15)}{15})}{8} \quad (5)$$

where N denotes the total number of quality levels and K denotes the total number of segments in the video. Moreover, Q_k is the quality level requested for the k^{th} segment, F_{freq} is the ratio of the number of video freezes to the number of segments, and F_{avg} is the average duration of all freezes.

C. Numerical Results

Fig. 4 shows the QoE values (i.e., MOS) of different bitrate adaptation schemes when the backbone bandwidth is 5 Mbps. As the highest data rate of the wireless channel is 4 Mbps, the bottleneck in this experiment exists in the wireless channel. The average data rate of the wireless channel ranges from 596.3 kbps to 2.605 Mbps in the experiment. With the increase of the average data rate, the variation of the channel data rate also decreases, demonstrating the improvement of the wireless channel condition. Fig. 4 shows that the QoE of each scheme increases from a low level with the increase of the average channel data rate. As seen, our proposed CBA method always performs the best compared to the four reference schemes.

Figs. 5-7 further show the three metrics in the QoE model, i.e., *average quality level*, *standard deviation of quality level*, and *freeze level*, respectively. Here, we normalize the results to have a clearer interpretation of the difference. It can be seen that ***CBA*** is able to provide a high average quality level while maintaining relatively low levels of quality variation and video freezes at the same time. On the contrary, ***SA*** achieves the most stable video quality, whereas its performance is the worst when the channel condition is bad, due to its slow reaction towards bandwidth changes. ***IA*** performs better than SA, BBA, and CA when the channel condition is good. This can be attributed to IA's fast reaction towards bandwidth changes and the low freeze level as a consequence. ***BBA*** outperforms SA, IA, and CA, especially when the channel condition is bad.

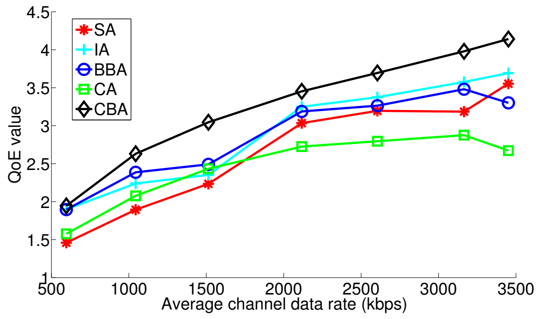

Fig. 4. QoE in terms of MOS values.

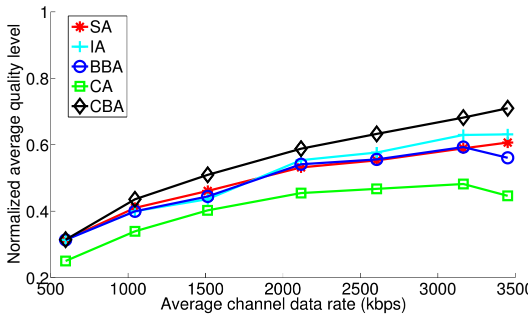

Fig. 5. Normalized average quality level

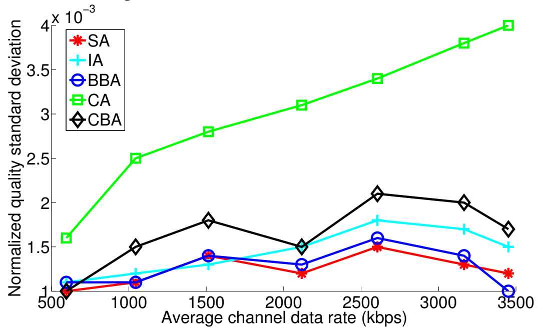

Fig. 6. Normalized quality standard derivation.

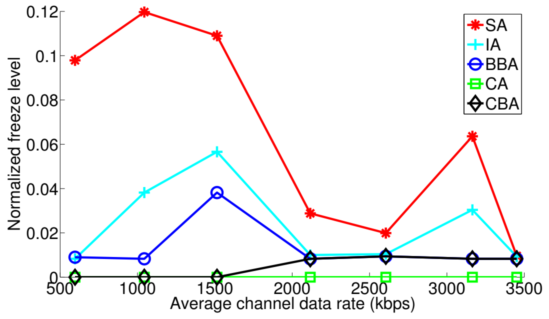

Fig. 7. Normalized freeze level.

This is because, in bad channel conditions, this buffer-based scheme BBA does better than the other three schemes above in mitigating video freezes while maintaining a relatively high average video quality at the same time. ***CA*** achieves the lowest freeze level, but also the lowest average quality level and the highest quality variation, because of its conservative nature.

We can observe some fluctuations in Fig. 6 and Fig. 7. It is because the channel variation decreases along with the increase of the average channel data rate. When the average channel data rate increases, the client tends to choose segments of better quality, which increases the probability of freezes and quality variation. On the contrary, with the decrease of the channel variation, the probability of freezes and quality variation both decrease consequently. Under the contradictory influence of these two factors, the quality variation and freeze level may fluctuate largely if the adaptation schemes cannot properly balance the opposite effects.

V. Conclusions

In this paper, we propose a novel bitrate adaptation method with proxy caching for video streaming over mobile networks, which can significantly improve the overall QoE of end users. Our proposed method can adapt to the network condition by exploiting prediction of the channel bandwidth and proxy-based content prefetching. The accurate prediction of the channel condition increases the probability of prefetching segments of proper bitrates, and decreases the freeze level as a result. The prefetching of segments by the caching proxy from the remote server also enables delivery of segments of bitrates higher than those achievable with the existing bitrate adaptation schemes. Experiment results demonstrate that our proposed method outperforms the reference schemes in terms of the overall QoE in various network conditions. This is attributed to the advantages of our method in providing a higher quality level while maintaining a low freeze level.

References

[1] S. Akhshabi, A. C. Begen, and C. Dovrolis, “An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP,” in *Proceedings of 2nd Annual ACM Conference on Multimedia Systems*, 2011, pp. 157–168.

[2] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham, “Adaptive streaming of audiovisual content using MPEG DASH,” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 1, pp. 78–85, 2012.

[3] K. Miller, E. Quacchio, G. Gennari, and A. Wolisz, “Adaptation algorithm for adaptive streaming over HTTP,” in *Proceedings of 19th International Packet Video Workshop (PV)*, 2012, pp. 173–178.

[4] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, “Using the buffer to avoid rebuffers: Evidence from a large video streaming service,” 2014. [Online]. Available: http://arxiv.org/abs/1401.2209

[5] C. Liu, I. Bouazizi, and M. Gabbouj, “Rate adaptation for adaptive HTTP streaming,” in *Proceedings of 2nd Annual ACM Conference on Multimedia Systems*, 2011, pp. 169–174.

[6] 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 SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[7] M. Claeys, S. Latré, J. Famaey, and F. De Turck, “Design and evaluation of a self-learning HTTP adaptive video streaming client,” *IEEE Communications Letters*, vol. 18, no. 4, pp. 716–719, 2014.

[8] J. He, H. Zhang, B. Zhao, and S. Rangarajan, “A collaborative framework for in-network video caching in mobile networks,” in *Proc. IEEE SECON*, 2013, pp. 406–414.

[9] P. Sadeghi, R. A. Kennedy, P. B. Rapajic, and R. Shams, “Finite-state Markov modeling of fading channels - A survey of principles and applications,” *IEEE Signal Processing Magazine*, vol. 25, no. 5, pp. 57–80, 2008.

[10] “What is NetLimiter?” Locktime Software, 2014. [Online]. Available: http://www.netlimiter.com/

[11] C. Müller *et al.*, “ITEC-Dyanmic Adaptive Streaming over HTTP,” The VINT Project, 2011. [Online]. Available: http://www-itec.uni-klu.ac.at/dash/