A CONTROL-THEORETIC APPROACH TO RATE ADAPTION FOR DASH OVER MULTIPLE CONTENT DISTRIBUTION SERVERS

Anonymous ICME submission

ABSTRACT

Recently, Dynamic Adaptive Streaming over HTTP (DASH) has been widely deployed on the Internet. However, it is still a challenge to play back video smoothly with high quality in time-varying Internet. It reuses popular web servers rather than relying on expensive streaming servers. In this paper, we investigate the problem of rate adaption over a new architecture for DASH, i.e. DASH from multiple content distribution servers. We take the advantage of multiple servers to offer even better DASH experience. In our scheme, users request multiple fragments, which is denoted as a block, in parallel from different servers to obtain larger bandwidth. We dynamically determine the number of fragments in a block depending on the bandwidth ratio of multiple servers. The fragment requests are scheduled to multiple servers according to their playback deadline priority. We propose to adapt the video rate when a block is completely downloaded, rather than a fragment, therefore, the requested video rates for multiple server are synchronized. Combining with the buffered video time information and predicted bandwidth of multiple servers, we propose a control-theoretic approach for rate adaption to provide a continuous video playback with high quality level. In the approach, we modify the traditional buffer model that two thresholds, an overflow threshold and an underflow threshold, are added to filter short-term rate variations and smooth playback is provided. We model and linearize the rate adaption system. By a Proportional-Derivative (*PD*) controller, we are able to adapt video rates with high responsiveness and stability. We carefully design the parameters for the *PD* controller, the phase margin and gain margin are also analyzed.

Index Terms— One, two, three, four, five

1. INTRODUCTION

dash is hot: In recent years, *Dynamic Adaptive Streaming over Http* (DASH) has been widely adopted for providing uninterrupted video streaming service to users with dynamic network conditions and heterogeneous devices[1, 2, 3]. Contrary to the past RTP/UDP, the use of HTTP over TCP is easy to configure and in particular, it greatly simplifies the traversal of firewalls and network address translators (NAT). Besides, it is cheap to be deployed since it employs standard HTTP servers and it also can be easily deployed within Content Delivery Networks (CDN). In DASH, a video content is encoded into multiple versions at different rates. Each encoded video is further divided into small video fragments, each of which normally contains seconds or tens of seconds worth of video. Using Http protocol, a client downloads video fragments sequentially by sending http "GET" requests to a server. Upon network condition changes, a client dynamically switch video version for the segments to be downloaded. Dynamic Http streaming is preferred by more and more content providers, including Microsoft smooth streaming[4], Apple HTTP live steaming[5], Adobe HTTP dynamic streaming[6], Netflix[7, 8], etc.

rate adaption:related work: Dynamic rate adaption is one of the most hot research topics since it is able to automatically throttle the video quality to match the available bandwidth, so that the user receives the video at the maximum possible quality. The existing rate adaption techniques can be classified into two main categories:

- Bandwidth based rate adaption. In bandwidth based rate adaption technique, it switches up or down the rate with estimated network bandwidth. Liu[9] proposed a smoothed HTTP/TCP throughput based rate adaption method for DASH. Most of the commercial vendors also adopt bandwidth based rate adaption. However, the inherent time-varying bandwidth would lead to short-term rate oscillation and deteriorate user experience of streaming services. This is demonstrated by experimental evaluation[10]. In [10], authors compared rate adaption of three popular DASH clients: Netflix client, Microsoft Smooth Streaming[4], and Adobe OSMF[11]. They concluded that none of them is good enough. They are either too aggressive or too conservative. Some clients even just jump between the highest video rate and the lowest video rate. Besides, bandwidth based rate adaption may cause playback freeze when bandwidth is over overestimated.

- Buffer based rate adaption. In buffer based rate adaptation technique, the rate selection decision is made to provide a continuous video playback by keeping the buffered video time staying around a predefined threshold. Though this hind of method can provide continuous video playback, it may cause dramatic video rate oscillation and lead to inferior user quality-of-

experience[12]. There are very few work about buffer based rate adaption technique. A feedback control mechanism is proposed in[13], but it adjusts the video rate by controlling the sending buffer size on the server side.

multiple servers: On the other hand, multiple content distribution servers have been adopted by most multimedia providers, such as Netflix and Hulu, which are the leading providers for online movies and TV shows. Adhikari[14, 15] study the architecture and their performance of Netflix and Hulu. It shows that though multi-CDN servers are adopted in these systems, only one server is used for a session at a time. The authors also demonstrate that a larger bandwidth is obtained by using multiple servers simultaneously through experiments.

challenge: However, to the best of our knowledge, there has been no work about rate adaption for DASH over multiple content distribution servers. Rate adaption for DASH over multiple servers faces great challenges:

- The heterogeneous bandwidth of multiple servers would cause playback video rate changed frequently. In all existing rate adaption algorithm, the rate is adjusted when a fragment is completely downloaded. While for multiple servers, multiple fragments are requested in parallel from different servers to obtain larger bandwidth. Because of the various bandwidth of multiple servers, the time needed to download a fragment is different. Therefore, if the rate is adjusted whenever a fragment is completely downloaded, the requested video rate for multiple servers may be different. This is because when a fragment is completely downloaded from a server, the requested is updated. But the new rate requested from this server may be different from the other since they have not completely downloaded the fragments with old rate and their requested rate can not be updated. This phenomenon leads the playback rate changed frequently and greatly deteriorate user quality-of-experience

- It is challenging to adapt video rate smoothly while providing a continuous video playback with high quality level. The existing bandwidth based and buffer based rate adaption algorithms either cause playback freeze or video rate fluctuation. These methods are also designed for single server and they can not be directly employed for multiple servers. Therefore, how to adapt video rate for DASH form multiple servers while providing smooth, continuous, and high quality video playback faces great challenge.

our work: In this paper, we investigate the problem of rate adaption for DASH from multiple content distribution servers. We take the advantage of multiple servers to offer even better DASH experience. Multiple fragments, which is denoted as a block, are requested in parallel from different servers to obtain larger bandwidth. The video rate is adapted when a block is completely downloaded rather than a fragment. All fragments in a certain block have the same video rate, therefore the requested video rate are smoothed. Combining with the buffered video time information and predicted bandwidth of multiple servers, a novel control-theoretic approach for rate adaption is proposed to provide a continuous video playback with high quality level. Our contributions are four-fold:

- We are the first to investigate the problem of rate adaption for DASH from multiple content distribution servers. A novel control-theoretic approach for rate adaption is proposed and a smooth, continuous, and high quality video playback is provided.

- We proposed to adapt video rate in block granularity rather than fragment granularity. To the download time, we dynamically determine the number of fragments in each block and schedule the requests of these fragments to multiple servers according to their playback deadline priority. Therefore, all fragments in a certain block have the same video rate and multiple servers complete the downloading requests nearly at the same time. The effect of heterogeneous bandwidth of multiple servers on video rate adaption is removed.

- A novel control-theoretic approach is proposed to select the best video rate for each block based on the buffered video time information and predicted bandwidth of multiple servers. In the approach, we modify the traditional buffer model that two thresholds, an overflow threshold and an underflow threshold, are added. The video rate is adapted only when the buffered video time is higher than the overflow threshold or lower than the underflow threshold, so a smooth video rate is provided. We model and linearize the rate adaption system, and a Proportional-Derivative (*PD*) controller is presented. By carefully designing the parameters for the *PD* controller, we are able to adapt video rates with high responsiveness and stability.

2. FRAGMENT REQUEST SCHEDULING

Since video rate adaption in fragment granularity would cause requested rate are updated asynchronously, we adapt the video rate in block granularity that when a block is completely downloaded, the video rate is adjusted. Then we face two questions: "*how to determine the length, i.e. the number of fragments in each block?*" and "*how to schedule the requests of these fragments to multiple servers?*"

2.1. Dynamic block length

We dynamically determined the block length according to the bandwidth ratio of multiple servers. Whenever a block is completely downloaded, the block length is updated as follows: for all available servers, the number of fragments requested from the server with lowest bandwidth is set to one. For the other servers, the number of fragments requested from them are set equal to the ratio of their bandwidth and the lowest bandwidth. Floor operation is employed to transform the ratios into integer. Since the ratio of fragments requested from multiple servers is approximate the same with their bandwidth ratio, all servers completely download the fragments in a certain block approximately at the same time. The effect of heterogeneous bandwidth of multiple servers is removed. At last, the block length is the total number of fragments requested from all servers.

On the other hand, if the block length too large, video rate may be adapted too sluggish. Therefore, if the length is bigger than a predefined value, the server with lowest bandwidth is discard and no fragment is requested from it. For the remaining servers, the number of fragments requested from them is updated using the same method described in the above paragraph. This process is iterated until the block length is no bigger than the predefined threshold.

We suppose there are S servers with bandwidth c_i for i-th server. The bandwidth is estimated by existing method and we do not focus on the bandwidth estimation problem in this paper. We sort the bandwidth in descending order so that $c_i \geq c_j, \forall i < j$. We denotes the length of k-th block as N_k with n_i fragments are requested from i-th server. The maximal length of the block is denoted as N, then the algorithm for determining the length of block k is presented in Algorithm 1.

Algorithm 1 Block Length Updating Algorithm

initialize $c_{\min} = c_s, S_{\max} = S$;
while 1 **do**
  $n_{S_{\max}} = 1$;
  for $i = 1$ to $S_{\max} - 1$ **do**
    $n_i = \left\lfloor \frac{c_i}{c_{\min}} \right\rfloor$;
  end for
  $N_k = \sum_{i=1}^{S_{\max}} n_i$
  if $N_k > N$ **then**
    $S_{\max} = S_{\max} - 1, c_{\min} = c_{S_{\max}}$;
  else
    return $N_k, S_{\max}$;
  end if
end while

2.2. Playback deadline based request scheduling

In the above subsection, we have determined the block length and which servers would be used. In this subsection, we schedule the fragment requests to these servers. Let $\mathbf{X}$ be the binary indicator matrix with size $N_k \times S_{\max}$, its entries $x_{ij} = 1$ if i-th fragment in block k is requested form server j, otherwise $x_{ij} = 0$. Since each fragment have the same size in a certain block(the same bitrate and playback duration), its download time has an inverse relationship with servers' bandwidth. In order to provide a continuous video playback, fragment i has higher playback deadline priority than fragment j when $i < j$. Therefore, we schedule the fragment requests to multiple server with the objective that fragment i is completely downloaded no later than fragment j for any $i < j$. The algorithm is presented in Algorithm 2:

Algorithm 2 Fragment Request Scheduling Algorithm

initialize $\mathbf{X} = \mathbf{0}$;
for $i = 1$ to N_k **do**
  $j* = \underset{1 \leq j \leq S\max}{\operatorname{argmin}} \left(\frac{1}{c_j} + \sum_{t=1}^{i-1} x_{tj} \frac{1}{c_j} \right)$;
  $x_{ij^*} = 1$;
end for
return $\mathbf{X}$;

3. BUFFERED VIDEO TIME MODEL

To sustain continuous playback, a video streaming client normally maintains a video buffer to absorb temporary mismatch between video download rate and video playback rate. In conventional single-version video streaming, video buffer is measured by the size of buffered video, which can be easily mapped into buffered video playback time when divided by the average video playback rate. In DASH, different video versions have different video playback rates. Since a video buffer contains segments from different versions, there is no longer direct mapping between buffered video size and buffered video time. To deal with multiple video versions, we use buffered video time to directly measure the length of video playback buffer.

Buffered video time process, denoted as $q(t)$, can be modeled as a queue with constant service rate of 1, i.e. in each unit of time, a piece of video with unit playback time is played and dequeued from the buffer. The enqueue process is driven by the video download rate and the downloaded video version. Specifically, for a video content, there are L different versions, with different playback rates $V_1 < V_2 < \cdots < V_L$. The set of these different versions is denoted as $\mathcal{V}$. All versions of the video are partitioned into fragments, each of which has the same playback time of Δ. We adapt the video rate on when a block is completely downloaded rather than a fragment. Without loss of generality, a client starts to download block

k at time instant $block\_t_k^s$ and the block is completely downloaded $block\_t_k^{(e)}$. The start download time and end download time for fragment $n(1 \leq n \leq N_k)$ in block k is denoted as $frag\_t_{n,k}^s$ and $frag\_t_{n,k}^s$, the video rate for block k is denoted as $v(k)$. According to Algorithm 2, we have

$$frag\_t_{n,k}^e - frag\_t_{n,k}^s = \frac{\Delta v\left(k\right)}{\sum_{j=1}^{S_{\max}} x_{nj} c_j} \tag{1}$$

$$frag\_t_{n,k}^e - block\_t_k^s = \frac{\Delta v\left(k\right)}{\sum_{j=1}^{S_{\max}} x_{nj} c_j} \sum_{j=1}^{S_{\max}} \left(x_{nj} \sum_{i=1}^{n} x_{ij} \right) \tag{2}$$

Equ.1 denotes the time consumed to download fragment n, Equ.2 denotes the time consumed to download all fragments before fragment n, including fragment n. When $n = N_k$, Equ.2 also denotes the time consumed to download block k. Then we model the buffered video time as

$$q\left(frag\_t_{n,k}^e\right) = q\left(block\_t_k^s\right) + \Delta n - \Delta v\left(k\right)\alpha\left(n\right). \tag{3}$$

where $\alpha(n) = \frac{\sum_{j=1}^{S_{\max}} \left(x_{nj} \sum_{i=1}^{n} x_{ij} \right)}{\sum_{j=1}^{S_{\max}} x_{nj} c_j}$ is independent from video rate $r(k)$. For any given n, $\alpha(n)$ can be easily obtained since $\mathbf{X}$ has been derived in Algorithm 2. Then the change rate of buffered video time, i.e. the first derivative of $q(t)(t \in [block\_t_k^s, block\_t_k^e])$ is modeled as

$$q'\left(t\right) = \frac{q\left(block\_t_k^e\right) - q\left(block\_t_k^s\right)}{block\_t_k^e - block\_t_k^s} = \frac{n}{v\left(k\right)\alpha\left(n\right)} - 1\,|_{n=N_k} \tag{4}$$

4. CONTROL-THEORETIC APPROACH TO RATE ADAPTION

In order to provide smooth, continuous, and high quality video playback, a suitable video rate needs to be selected. We propose to add two buffered video time thresholds, $q_{\min}$ and $q_{\max}$, as the operating points to filter the effect of buffer oscillations on rate decision. The video rate is adapted when the buffered video time is higher than the overflow threshold or lower than the underflow threshold, therefore a high quality level is provided with no playback freeze. Otherwise, the rate keep unchanged and a smooth video rate is provided.

4.1. Rate Control Model and Linearization

The video rate is adapted only when the buffered video time is higher than the overflow threshold or lower than the underflow threshold. When $q\left(block\_t_k^s\right) < q_{\min}$, $q_{\min}$ is used as the operating point to selected an appropriate video version to drag current buffered video time to $q_{\min}$. To ensure no buffer underflow happens for any fragments, we must have $q\left(frag\_t_{n,k}^e\right) \geq q_{\min}$ for all $n = 1, 2, ..., N_k$, therefore

$$v\left(k\right) \leq \frac{n}{\alpha\left(n\right)} + \frac{1}{\Delta\alpha\left(n\right)}\left(q\left(block\_t_k^s\right) - q_{\min}\right) \tag{5}$$

Similarly, when $q\left(block\_t_k^{(s)}\right) > q_{\max}$, we have

$$v\left(k\right) \geq \frac{n}{\alpha\left(n\right)} + \frac{1}{\Delta\alpha\left(n\right)}\left(q\left(block\_t_k^s\right) - q_{\max}\right) \tag{6}$$

Equ.5 and Equ.6 give the bounds of video rate to prevent buffer either underflow or overflow. In order to avoid too low or too high rate being selected, which may cause video rate changed frequently, we use the boundary values of Equ.5 and Equ.6 to control the video rate:

$$\begin{cases} v\left(k\right) = \frac{n}{\alpha(n)} + \frac{1}{\Delta\alpha(n)}\left(q\left(block\_t_k^s\right) - q_0\right) \\ q'\left(t\right) = \frac{n}{v(k)\alpha(n)} - 1 \\ q_0 = \begin{cases} q_{\min}, \text{if } q\left(t\right) < q_{\min} \\ q_{\min}, \text{if } q\left(t\right) < q_{\min} \end{cases} \end{cases} \tag{7}$$

The block diagram of the model described in (7) is shown in Fig.1, where $Q\left(\cdot\right)$ and $R\left(\cdot\right)$ are quantization and reciprocal operations respectively. It is is a nonlinear system, which is not suitable for analysis and implementation.

We next linearize the model described in (7) around the operating point q_0 satisfying $q'\left(t\right) = 0$. Let the corresponding video rate be v_0, then we have

$$q'\left(t\right) = 0 \Rightarrow v_0 = \frac{N_k}{\alpha\left(N_k\right)} \tag{8}$$

The following linearized state and output equations are obtained for the feedback system:

$$\begin{cases} \delta\left(v\left(k\right)\right) = \frac{1}{\Delta\alpha(n)}\delta\left(q\left(t\right)\right) \\ \delta\left(q'\left(t\right)\right) = -\frac{1}{v_0}\delta\left(v\left(k\right)\right) \end{cases} \tag{9}$$

where $\delta\left(v\left(k\right)\right) = v\left(k\right) - v_0$ and $\delta\left(q'\left(t\right)\right) = q'\left(t\right) - q_0'$. Applying the Laplace transform to the differential equations, we derive the linearized system block as shown in Fig.2. With some manipulation of the block diagram, we obtain the transfer function of the plant as

$$H\left(s\right) = \frac{\alpha\left(N_k\right)}{\Delta N_k \alpha\left(n\right) s + \alpha\left(N_k\right)} \tag{10}$$

Remark 1 (Stability): The pole of the linearized system is $s_p = -\frac{\alpha(N_k)}{\Delta N_k \alpha(n)}$ and it is located in the left phase plane. This indicates that the equilibrium state of the dynamics of the system is stable.

$$\delta\left(v\left(k\right)\right)=\frac{1}{\Delta\alpha\left(n\right)}K_p\left(q\left(block\_t_k^s\right)-q_0\right)+\frac{1}{\Delta\alpha\left(n\right)}K_d\frac{q\left(frag\_t_{n,k-1}^e\right)-q\left(block\_t_{k-1}^s\right)}{frag\_t_{n,k-1}^e-block\_t_{k-1}^s} \tag{11}$$

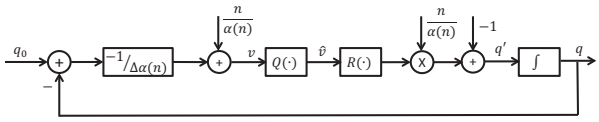

Fig. 1. Block diagram of the buffer based rate adaptation model.

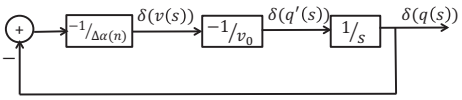

Fig. 2. Block diagram of the linearized model.

4.2. PD Controller for Rate Adaption

In this section, we design an effective controller for the linearized feedback system in Eq.(9). We redraw the block diagram in Fig.3. The $G_c(s)$ block represents the rate control component to be designed. From the control prospective, $G_c(s)$ is the controller to be designed, while the combination of the other blocks is the plant to be controlled.

We choose a proportional derivative (*PD*) controller since the proportional cycle can accelerate response time and improve system accuracy, and the derivative element can improve system dynamic performance. The classic *PD* controller can be designed with the following transfer function:

$$G_c\left(s\right)=K_p+K_ds \tag{12}$$

This controller is implemented at the client side. It uses the difference between current buffered video time and operation point q_0, and the variation tread of buffered video time to compute the adjustment of video rate. In the time domain, the rate adjustment is written in Equ.(11) for block k.

Generally, $\delta\left(v\left(k\right)\right)$ in Equ.(11) has different values for different $n=1,2,...,N_k$. Since the video rate is adapted in block granularity, therefore, when the buffered video time is higher than the overflow threshold, the biggest adjustment(generally with positive value) should be added to increase the video rate. On the other hand, when the buffered

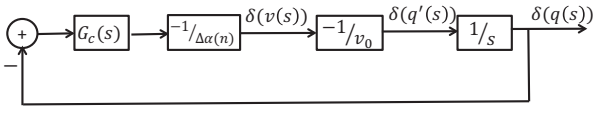

Fig. 3. Block diagram of the feedback control system for rate adjustment.

video time is smaller than the underflow the smallest adjustment(generally with negative value) should be used to decrease the video rate. Then the video rate for block k is shown as:

$$\tilde{v}\left(k\right)=\begin{cases}v_0+\min\left(\delta\left(v\left(k\right)\right)\right) & \text{if } q\left(block\_t_k^s\right)<q_{\min}\\ v_0+\max\left(\delta\left(v\left(k\right)\right)\right) & \text{if } q\left(block\_t_k^s\right)>q_{\max}\\ v\left(k-1\right) & \text{else}\end{cases} \tag{13}$$

Taking into account the finite discrete video rates, the actual requested video rate for segment k is $v\left(k\right)=Q\left(\tilde{v}\left(k\right)\right)$, where $Q\left(\cdot\right)$ is the quantization function.

At last, the rate adaption scheme for DASH over multiple servers is summarized as: whenever a block is completely downloaded, *i*)update the block length for next block by Algorithm 1; *ii*)request the fragments belonging to the new block from multiple servers by Algorithm 2; *iii*)set all fragments in the block with the same video rate according to Equ.(13).

4.3. Parameter Analysis

In this subsection, we analyze the parameters K_p and K_d to stabilize the close loop system described in Fig.3. Moreover, the gain margin (GM), phase margin (PM), and settling time of the control system are also analyzed. The overall system transfer function is

$$H_c\left(s\right)=\frac{K_ds+K_p}{\left(\Delta\alpha\left(n\right)v_0+K_d\right)s+K_p} \tag{14}$$

Now, we have the following propositions for the *PD* controller.

Proposition 1. *The feedback system in Fig.3 is stabilized by the PD controller, if the parameters satisfy the following conditions:*

$$\begin{cases}w_c\geq\frac{1}{\Delta_s}\sqrt{\frac{\Delta N_k+K_d}{\Delta N_k-K_d}}\ln\frac{20\Delta N_k}{\Delta N_k+K_d}\\ K_p=\sqrt{\left(\Delta N_k\right)^2-K_d^2}w_c\\ 0<K_d<\Delta N_k\end{cases} \tag{15}$$

Proof. According to proposition 1, $K_p>0$ and $K_d+\Delta n>0$. Therefore, the only pole of Eq.(14) is $s_p=-\frac{K_p}{\Delta n+K_d}<0$, which is located in the left phase plane. The system in Fig.3 is proved to be stable. □

Proposition 2. *For a PD controller satisfying Proposition 1, the* 5% *step response settling time* (T_s) *of the feedback control system satisfy:* $T_s\leq\Delta_s$.

Proof. The step response of the the system in Fig.3 is

$$U_c(s) = \frac{K_d s + K_p}{(\Delta\alpha(n) v_0 + K_d) s + K_p} \frac{1}{s}$$

Applying inverse Laplace transform, we have

$$u_c(t) = u(t) \left(1 - \frac{\Delta\alpha(n) v_0}{\Delta\alpha(n) v_0 + K_d} e^{-\frac{K_p}{\Delta\alpha(n) v_0 + K_d} t}\right)$$

From the constraints in proposition 1, the 5% step response settling time (T_s) of the feedback control system is

$$T_s = \frac{\Delta\alpha(n) v_0 + K_d}{\sqrt{(\Delta N_k)^2 - K_d^2 w_c}} \ln \frac{20\Delta\alpha(n) v_0}{\Delta\alpha(n) v_0 + K_d}$$
$$\leq \frac{\Delta\alpha(n) v_0 + K_d}{\Delta N_k + K_d} \frac{\ln \frac{20\Delta\alpha(n) v_0}{\Delta\alpha(n) v_0 + K_d}}{\ln \frac{20\Delta N_k}{\Delta N_k + K_d}} \Delta_s \leq \Delta_s$$

The last inequality is derived by the fact that $\alpha(n) \leq \alpha(N_k)$ for any $n = 1, 2, ..., N_k$, this is because fragment N_k would be completely downloaded at last according to Algorithm 2. □

5. REFERENCES

[1] A. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming Protocols," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 54–63, Mar. 2011.

[2] R. Kuschnig, I. Kofler, and Hellwagner H., "Evaluation of HTTP-based request-response streams for internet video streaming," *in Proc. ACM MMSys11*, pp. 245–256, Feb. 2011.

[3] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: standards and design principles," *in Proc. ACM MMSys11*, pp. 133–144, Feb. 2011.

[4] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft Corporation*, 2009.

[5] R. Pantos and W. May., "HTTP Live Streaming," *IETF Draft,*, jun. 2010.

[6] D. Hassoun, "Dynamic streaming in flash media server 3.5," *http://www.adobe.com/devnet/ashmediaserver/.*

[7] V. K. Adhikari, Yang Guo, Fang Hao, M. Varvello, V. Hilt, M. Steiner, and Zhi-Li Zhang, "Unreeling netflix: Understanding and improving multi-CDN movie delivery," *in Proceedings of IEEE INFOCOM*, pp. 1620–1628, Mar. 2012.

[8] J. F. Kurose and K. Ross, "Computer Networking: A Top-Down Approach Featuring the Internet," *6th ed. Addison-Wesley*, 2012.

[9] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj, "Rate Adaptation for Adaptive HTTP Streaming," *in Proc. ACM MMSys11*, pp. 169–174, Feb. 2011.

[10] S. Akhshabi, A. C. Begen, and Dovrolis C., "An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP," *in Proc. ACM MMSys11*, pp. 169–174, Feb. 2011.

[11] Adobe, "Open Source Media Framework," *http://www.osmf.org/.*

[12] E. C. R. Mok, X. Luo, and R. Chang, "Qdash: A Qoe-aware DASH system," *in ACM Multimedia Systems*, Feb. 2012.

[13] L. De Cicco, S. Mascolo, and Palmisano V., "Feedback Control for Adaptive Live Video Streaming," *in Proc. ACM MMSys11*, pp. 145–156, Feb. 2011.

[14] V.K. Adhikari, Yang Guo, Fang Hao, M. Varvello, V. Hilt, M. Steiner, and Zhi-Li Zhang, "Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery," *in INFOCOM*, pp. 1620–1628, Mar. 2012.

[15] V.K. Adhikari, Yang Guo, Fang Hao, V. Hilt, and Zhi-Li Zhang, "A Tale of Three CDNs: An Active Measurement Study of Hulu and Its CDNs," *in INFOCOM12 Mini-conference*, pp. 7–12, Mar. 2012.