Modeling and Analyzing the Influence of Chunk Size Variation on Bitrate Adaptation in DASH

Tong Zhang, Fengyuan Ren, Wenxue Cheng, Xiaohui Luo, Ran Shu, Xiaolan Liu
Tsinghua National Laboratory for Information Science and Technology, Beijing, China
Department of Computer Science and Technology, Tsinghua Univeresity, Beijing, China
{zhangt, renfy, chengwx, luoxiaohui, shuran}@csnet1.cs.tsinghua.edu.cn
xl-liu12@mails.tsinghua.edu.cn

***Abstract*—Recently, HTTP-based adaptive video streaming has been widely adopted in the Internet. Up to now, HTTP-based adaptive video streaming is standardized as Dynamic Adaptive Streaming over HTTP (DASH), where a client-side video player can dynamically pick the bitrate level according to the perceived network conditions. Actually, not only the available bandwidth is varying, but also the chunk sizes in the same bitrate level significantly fluctuate, which also influences the bitrate adaptation. However, existing bitrate adaptation algorithms do not accurately involve the chunk size variation, leading to performance losses. In this paper, we theoretically analyze the influence of chunk size variation on bitrate adaptation performance. Based on DASH system features, we build a general model describing the playback buffer evolution. Applying stochastic theories, we respectively analyze the influence of the chunk size variation on rebuffering probability and average bitrate level. Furthermore, based on theoretical insights, we provide several recommendations for algorithm designing and rate encoding, and also propose a simple bitrate adaptation algorithm. Extensive simulations verify our insights as well as the efficiency of the proposed recommendations and algorithm.**

***Index Terms*—Chunk Size, bitrate adaptation, rebuffering, average video rate, DASH**

I. Introduction

In recent years, HTTP-based adaptive video streaming has been widely employed in Internet video technologies, such as Microsoft Smooth Streaming [1, 2], Netflix [3], Apples HLS [4], Adobes HDS [5], and Akamai HD [6]. So far, HTTP-based adaptive video streaming is standardized as Dynamic Adaptive Streaming over HTTP (DASH) [7], where a client-side video player can dynamically pick the bitrate level according to the perceived available bandwidth. In DASH systems, each video is encoded into streams of multiple discrete bitrates and each stream is cut into multiple chunks (2-10 seconds of video time). Because chunks with the same ordinal in all streams are aligned in the video time, a player can switch to a different video rate at chunk boundaries.

The bitrate adaptation algorithm in the video player each time decides the bitrate level of the next chunk to be downloaded. If the chosen bitrate is too high, there will be rebuffering events (the video play pauses since the playback buffer is empty), whereas choosing a too low bitrate will under-utilize the available bandwidth and impair the video quality. All these cases degrade the viewing experience. Consequently, a good bitrate adaptation algorithm is vital to improve user experience [7]. Two principal goals are avoiding rebuffering events and maximizing the average bitrate level respectively [8, 9].

However, designing a suitable bitrate adaptation algorithm is challenging because of the time-varying property of the DASH system [10]. Not only the available bandwidth is varying, but also the chunk sizes in the same bitrate level significantly vary as a result of variable bitrate (VBR) encoding. The available bandwidth depends on the network environment when playing video, while all actual chunk sizes are already defined at encoding. That is, when a player chooses the next bitrate the future available bandwidth cannot be accurately acquired while the subsequent chunk sizes of all levels are known a priori. In this context, addressing the chunk size variation makes sense to improve the algorithm performance. However, as far as we know the varying chunk size has not been explicitly studied so far.

In this paper, we theoretically analyze the influence of chunk size variation on bitrate adaptation algorithm performance. Based on the features of HTTP-based adaptive video streaming, we build a general model describing the playback buffer evolution process. Applying stochastic theory, we respectively analyze the two most concerned metrics – rebuffering probability and average bitrate level as well as their relationships with chunk size variation in the widely-adopted rate-based algorithm. Furthermore, based on theoretical insights, we provide some recommendations for algorithm designing and bitrate encoding and also propose a lightweight adaptation algorithm. Extensive simulations verify our insights as well as the efficiency of proposed recommendations and algorithm.

In summary, our contributions are three-fold:

- Establishing a generally applicable model for HTTP-based adaptive video streaming, and theoretically analyze both the rebuffering probability and the average bitrate level as well as the effect of chunk size variation on the two metrics.
- Providing suggestions for designing a suitable bitrate adaptation algorithm as well as encoding videos based on the obtained insights.
- Designing a simple but effective bitrate adaptation algorithm.

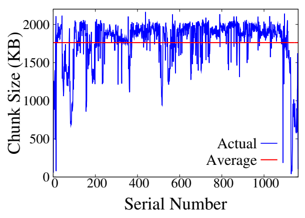
The remainder of the paper is organized as follows. In Section II, we briefly introduce the HTTP-based adaptive video streaming service and VBR encoding. We also count the chunk sizes of a real video to observe the statistical properties. We model the playback buffer evolution process and use stochastic theory to analyze both the rebuffering probability and average bitrate level in Section III. In the light of analysis results, we provide some suggestions for designing adaptation algorithms and encoding bitrates. In Section IV, we simply design a bitrate adaptation algorithm based on our model and analysis. Section V verifies both the correctness of our insights as well as the efficiency of our proposals. Finally, the paper is concluded in Section VI.

II. Background and Motivation

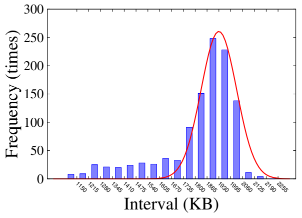
Today, video streaming is increasingly dominating the Internet traffic [11] and simultaneously, HTTP-based adaptive video streaming has gained very extensive adoption. Internet video services like Microsoft Smooth Streaming [1, 2], Netflix [3], Apples HLS [4], Adobes HDS [5], and Akamai HD [6] all runs over HTTP-based adaptive video streaming. Compared with early connection-oriented video transport protocols, HTTP-based streaming has several practical advantages. It allows traffics to traverse NATs and firewalls [12], enables the use of existing commodity CDN servers [13] and inspires better application resilience by making the server stateless [14], which explains its popularity.

HTTP-based adaptive video streaming is standardized as DASH where videos are chunked and encoded at multiple discrete bitrates. No matter what level a chunk belongs to, it always contains a fixed period of play time, 2-10 seconds in general. When a DASH player plays back a video, each time it sends an HTTP request to the CDN server and downloads one chunk. The downloaded chunks are stored in the client-side playback buffer waiting for playing, and a chunk cannot start playing until it is fully downloaded. Since chunks in all bitrate streams are aligned in the playback time, the player can select a different bitrate at each chunk boundary. The bitrate adaptation algorithm inside the player each time picks a bitrate level according to the current network conditions, thus the algorithm performance is critical to the viewers' experience [7]. Too high bitrate is likely to cause rebuffering events while too low bitrate will underutilize the available bandwidth and impair the video quality. Both cases reduce the user experience. Consequently, adaptation algorithms are forced into a tradeoff and the two principal goals are avoiding rebuffering events as well as maximizing the average bitrate level [8, 9].

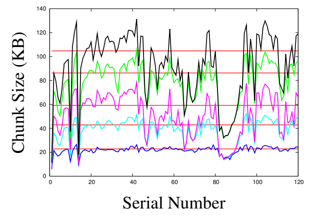
A good bitrate adaptation algorithm should adapt to the dynamics of the DASH system. The dynamics here not only refer to the varying available bandwidth, but also mean the fluctuant chunk sizes with the same bitrate. In practice, most video streaming services adopt variable bitrate (VBR) encoding [15]. Compared with constant bitrate (CBR) encoding, VBR allows more flexibility as well as bit efficiency [15]. Given a nominal bitrate, VBR encodes static scenes with fewer bits and dynamic scenes with more bits, which is different from CBR encoding. As a result, in the same bitrate level chunk sizes will not keep constant but fluctuate over time, and the nominal



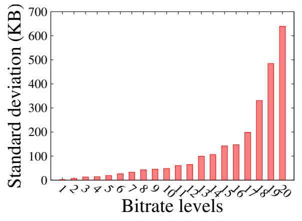
(a) Chunk sizes under VBR



(b) Distribution



(c) Variation trend



(d) Standard deviation

Fig. 1. Chunk size variation

bitrate represents the average video rate. Figure 1(a) depicts the 4-second chunk sizes of a real video (valkaama) encoded at 3.5 Mbps. The blue line describes the actual chunk sizes and the red line indicates the nominal(average) value. We can see that the chunk size drastically varies around the average. By counting all chunks in Figure 1(a), the standard deviation reaches up to 330KB on the condition that the average size is 1761KB, implying the frequent and violent chunk size fluctuation. Figure 1(b) exhibits the chunk size distribution of a randomly chosen level. We can see that, chunk sizes in the same bitrate level roughly obeys normal distribution, which forms the foundation of our later analysis.

Figure 1(c) focuses on 5 adjacent levels' chunk size in a specific small time horizon. By this way, we can explicitly observe the variation trend as well as find connections among different bitrates. We can clearly see that chunk sizes of various bitrates present consistent trend – they increase or decrease at the same time. The phenomenon is intuitive because besides bitrate levels, chunk size only depends on the activity of the scene, which is all the same among bitrate levels for any location in video. From Figure 1(c), another finding is the larger the nominal bitrate, the more violently chunk size varies. Figure 1(d) depicts the standard deviation of chunk size over all 20 bitrate levels, and we can observe that the deviation indeed goes up as the bitrate increases.

Intuitively, varying chunk sizes have an effect on the algorithm performance. Consider the case where there is only 1 chunk in the playback buffer and the next selected bitrate approximately equals the available bandwidth. If not considering chunk size variation (thinking all chunks for a given bitrate contains the same number of bytes), when the actual size is larger than the nominal value rebuffering will occur, and when the actual size is smaller a higher bitrate level could have been selected.

However, most existing adaptation algorithms only adapt themselves to the varying throughput while pay little attention to chunk size variation. Current solutions can be roughly divided into rate-based and buffer-based algorithms [13]. Rate-based algorithms such as [10, 16–19] firstly estimate the available bandwidth and pick the highest possible video rate based on that. They mainly focus on improving the throughput estimation instead of considering chunk size. To solve the throughput estimation inaccuracy, buffer-based algorithms like [8, 15] use playback buffer occupancy as the feedback signal rather than throughput estimation, and devote to control buffer occupancy at an expected value. Although in [8, 15] VBR encoding is referred to, the paper does not clarify the features or influence of chunk size but heuristically use them. Therefore, there is still spaces for improving the algorithm performance. Because all chunk sizes are determined at encoding and irrelevant to the external environment, they can be known in advance. We can take advantage over the case and take actual chunk sizes into consideration in designing. In this paper, we specially study the influence of chunk size variation on algorithm performance to provide some insights for algorithm designing, taking a common rate-based algorithm for example.

III. Modeling and Analysis

In this section, we firstly model the playback buffer evolution in DASH and formalize the bitrate adaptation into a function. On this basis, we theoretically analyze two principal metrics – rebuffering probability and average bitrate level, mainly discussing their relationships with chunk size variation.

A. Problem Formalization

To model the bitrate adaptation in DASH, we focus on the buffer occupancy since it involves a lot of information and closely correlated to our concerned two metrics. Like many previous models in [8, 10, 15, 19], in our model the playback buffer occupancy is measured in play time. That is because what the viewers care about is playback. Furthermore, chunks from the same video contain a fixed period of play time in DASH systems whatever bitrate level they belong to, which simplifies the play time calculation. On the other hand, our model is under the classic rate-based algorithm since its wide adoption in commercial vendors. Under this simple algorithm, each time a player picks a bitrate for the next chunk, it first estimates the available bandwidth and chooses the highest possible bitrate below the estimated value. [16]

Let B_{max} denote playback buffer size in seconds. We extract the buffer occupancy at each chunk download completion, and let the backlog at the completion of the i^{th} chunk download be B_i. t_i being the download time of chunk i and τ_i denoting its play time, the processes of chunk downloading and playing are both depicted in Figure 2. The above timeline illustrates downloading while the below one interprets playing, and the time scale of two lines are aligned. The line segments of one color/lightness represents the downloading of a chunk and the playing of the one right prior to it. Rebuffering events occur if and only if the previous chunk completes playing before the next one completes downloading, which

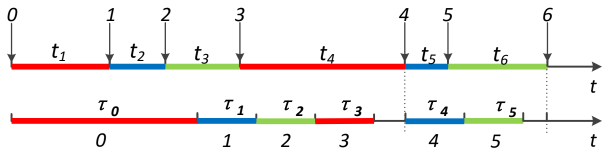


Fig. 2. Buffer occupancy evolution

corresponds to the segments with no color in Figure 2. Let $\mathbf{R} = \{R_1, R_2, \cdots, R_m\}$ represent the set of all optional bitrates. There are totally m levels for the player to choose and $R_1 < R_2 < \cdots < R_m$. We define the instantaneous size of chunk i as S_i and the average bandwidth during the i^{th} download as C_i, then the download time should be $\frac{S_i}{C_i}$. We also suppose the initial buffer occupancy before video starts to be t_s and each chunk contains Vs of play time, then we have $\tau_0 = t_s$ and $\tau_1 = \tau_2 = \cdots = V$. To simplify the model and subsequent analysis, we make some reasonable assumptions. Because in this paper we mainly address the chunk size variation, we let bandwidth be a constant value C, that is, $C_1 = C_2 = \cdots = C$, which is also assumed in [15]. Based on the constant throughput, we can reasonably continue to assume that the throughput estimation is accurate, because most of the prediction methods are based on historical bandwidth. Accordingly, the selected bitrate R should always be the highest possible level below C, i.e., $R = \max\limits_{1\leq j\leq m} \{R_j, R_j \leq C\}$. Furthermore, we suppose chunk sizes with bitrate j ($j = 1, 2, \cdots, m$) are i.i.d. and they obey normal distribution $N(C\mu_j, C^2\sigma_j^2)$, as demonstrated in Figure 1(b). In this case, if R_j is selected the chunk download time $t_i = \frac{S_i}{C}$ obeys another normal distribution $N(\mu_j, \sigma_j^2)$. Define $\mathbf{\Sigma}$ to be the set $\{\sigma_1, \sigma_2, \cdots, \sigma_m\}$. With above definitions and assumptions, we express the buffer occupancy evolution with the following equation.

$$B_{i+1} = \begin{cases} \left(B_i - \frac{S_{i+1}}{C}\right)_+ + V & B'_{i+1} \leq B_{max} \\ \left(B_i - \frac{S_{i+1}}{C}\right)_+ & B'_{i+1} \geq B_{max} \end{cases} \tag{1}$$

Each downloaded chunk adds Vs of play time to the buffer, but meanwhile a download time of video segment is played back, unless during this time the buffer goes empty. $B'_{i+1} = \left(B_i - \frac{S_{i+1}}{C}\right)_+ + V$, meaning the buffer occupancy to reach with unlimited buffer capacity. If B'_{i+1} is less than or equal to buffer size B_{max}, the actual B_{i+1} should equal B'_{i+1}, while if not, the newly downloaded chunk could not be added into the buffer.

In this paper, we focus on two main performance metrics – rebuffering probability P and average bitrate $E[R]$, then the bitrate adaptation could be generally described using the following equation.

$$(P, E[R]) = f\left(C, V, t_s, \mathbf{R}, \mathbf{\Sigma}, B_{max}\right) \tag{2}$$

where C, $\mathbf{R}$, $\mathbf{\Sigma}$ have been defined above, and P and $E[R]$ denote the two metrics we concern – rebuffering probability and average bitrate level. Equation 2 means that the performance metrics P and $E[R]$ are correlated with various bitrate adaptation parameters, and the adaptation algorithm itself is reflected in the abstract function f. In VBR encoding, the nominal bitrate represents the average video rate [15]. Thus if the selected bitrate is always R_j, we have $C\mu_j = R_j V$, meaning the average chunk size equals average bitrate multiplied by the play time per chunk. Hence, the bitrate level set $\mathbf{R}$ not only indicates different encoding schemes, but also refers to the average download time $\{\mu_1, \mu_2, \cdots, \mu_m\}$. To explore the influence of chunk size variation, we mainly study two broadly defined quantities – $\frac{\partial P}{\partial \sigma_j}$ and $\frac{\partial E[R]}{\partial \sigma_j}$, where j means the bitrate R_j is selected in the rate adaptation.

B. Influence of Chunk Size on Rebuffering

We first address the rebuffering issue and study the rebuffering probability P. Based on stochastic theories, we obtain the following conclusion.

Theorem 1. *If bitrate R_j is selected, the probability of rebuffering events during the video playback satisfies*

$$1 - \Phi\left(\frac{t_s - \mu_j}{\sigma_j}\right) \le P \le e^{\frac{(8-2\mu_j)(V-t_s)}{\sigma_j^2}}$$

Proof. The lower bound is straightforward. Suppose a video totally contains N chunks, and denote $A_i, i = 1, \cdots, N$ as the event of rebuffering occurring during the i^{th} chunk downloading, then

$$P = P(A_1 \bigcup A_2 \bigcup \cdots \bigcup A_N) \ge P(A_1)$$

That is, the probability of rebuffering during the whole video must be larger than or equal to that during the first chunk downloading. And because there exists t_ss of play time in the buffer before downloading and the download time of the first chunk is $\frac{S_1}{C}$, A_1 happens if and only if $\frac{S_1}{C} > t_s$, that is, $P(A_1) = P(\frac{S_1}{C} > t_s)$. Considering the normal distribution hypothesis, we have $P(A_1) = 1 - \Phi\left(\frac{t_s - \mu_j}{\sigma_j}\right)$, hence

$$P \ge 1 - \Phi\left(\frac{t_s - \mu_j}{\sigma_j}\right)$$

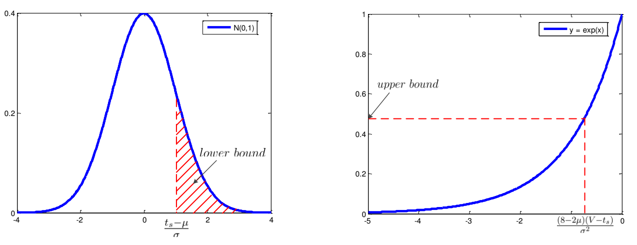
When it comes to the upper bound, firstly define a specific stochastic process $z(k) = e^{\theta(\sum_{i=1}^{k} \frac{S_i}{C} - \sum_{i=0}^{k-1} \tau_i)}$, where θ is the positive solution of equation $E[e^{\theta(\frac{S}{C} - V)}] = 1$. We first prove that $z(k)$ is a martingale with respect to the filtration

$$\mathcal{F}_k \triangleq \sigma\left\{\frac{S_i}{C}, \tau_i \middle| i \le k\right\}$$

which represents all possible values of $(\frac{S_i}{C}, \tau_i)$ before the $(k+1)^{th}$ downloading. Because we assume $\frac{S_i}{C}$ and $\tau_i, i = 1, 2, \cdots, N$ are two independent series, we have

$$\begin{aligned} E\left[z(k)|\mathcal{F}_{k-1}\right] &= E\left[e^{\theta \sum_{i=1}^{k}\left(\frac{S_i}{C} - \tau_{i-1}\right)} \middle| \mathcal{F}_{k-1}\right] \\ &= E\left[e^{\theta\left(\frac{S}{C} - V\right)}\right] e^{\theta \sum_{i=1}^{k-1}\left(\frac{S_i}{C} - \tau_{i-1}\right)} = z(k-1) \end{aligned}$$

under the condition on θ from the theorem. The second equality holds because $\frac{S_k}{C} - \tau_{k-1}$ is independent of $\mathcal{F}_{k-1}$



(a) Lower bound (b) Upper bound

Fig. 3. Lower and upper bound

and the identical relation that $E[f(X)|X] = f(X)$. Define the stopping time

$$K \triangleq \inf\left\{k > 0 \middle| \sum_{i=1}^{k}\left(\frac{S_i}{C} - \tau_{i-1}\right) > 0\right\}$$

meaning the time point where the first rebuffering event occurs, which is also the first location where $z(k) > 1$. In this case, the value of K is only relevant to the series $(\frac{S_i}{C} - \tau_{i-1})$, $i = 1, 2, \cdots, K$, satisfying the definition of stopping time. Note that with the assumption that the video contains N chunks in total, we have $P = P(K \le N)$, indicating the first rebuffering event occurs before the video ends. According to the *Optional Sampling Theorem* for martingales with $k \ge 1$,

$$\begin{aligned} E\left[z\left(K\right)\right] = E\left[z\left(1\right)\right] &= E\left[e^{\theta\left(\frac{S}{C} - t_s\right)}\right] \\ &= E\left[e^{\theta\left(\frac{S}{C} - V + V - t_s\right)}\right] = E\left[e^{\theta(V - t_s)}\right] \\ &\ge E\left[z(K)\right] P\left(K \le N\right) \ge P\left(K \le N\right) \end{aligned}$$

Thus we have

$$P\left(K \le N\right) \le e^{\theta(V - t_s)}$$

Since we assume $\frac{S}{C} \sim N(\mu, \sigma^2)$, substitute $V = 4$ into computing, we can obtain

$$\begin{aligned} E\left[e^{\theta\left(\frac{S}{C} - V\right)}\right] &= \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty} e^{\theta(x-V)} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \\ &= e^{-\frac{(8-2\mu)\sigma^2\theta - \sigma^4\theta^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty} e^{-\frac{\left(x - \left(\mu + \sigma^2\theta\right)\right)^2}{2\sigma^2}} dx \\ &= e^{-\frac{(8-2\mu)\sigma^2\theta - \sigma^4\theta^2}{2\sigma^2}} = 1 \end{aligned}$$

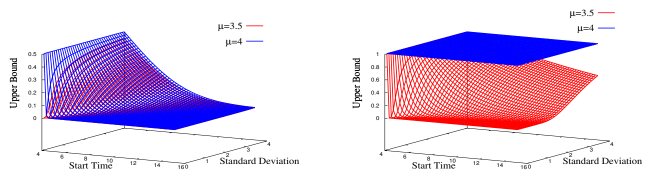
$$\theta = \frac{8 - 2\mu}{\sigma^2}$$

Considering $P = P(K \le N)$, we have

$$P \le e^{\frac{(8-2\mu)(V-t_s)}{\sigma^2}}$$

□

Figure 3 depicts both upper and lower bounds of rebuffering probability P. The area of the shaded part in Figure 3(a) represents the lower bound whose value is directly relevant to $\frac{t_s - \mu}{\sigma}$; While the location that the dotted line marks on the vertical axis in Figure 3(b) is the upper bound, depending on $\frac{(8-2\mu)(V-t_s)}{\sigma^2}$. We can see that both two bounds are correlated with three factors – μ, t_s, and σ and show the same variation trend when these three variables change. Intuitively, when t_s



(a) Lower bound (b) Upper bound

Fig. 4. Bounds v.s. different factors

increases, both upper and lower bounds goes down, while when μ or σ go up, both bounds decline. Figure 4 shows the variation trend of the two bounds with both t_s and σ changing, taking $V = 4$ for example. The two surfaces in each subgraph are under different μ. We can see that both bounds sharply vary as t_s or σ change. From Figure 4, we can come to the following insights as well as corresponding recommendations.

Insight 1. *Under any circumstances, chunk size variation (σ) will greatly increase the rebuffering probability.*

Recommendation 1. *When designing a rate selection algorithm, we need to consider the chunk size fluctuation.*

Insight 2. *A little start time (2-3 chunks) makes great contributions to decreasing rebuffering events.*

Recommendation 2. *The algorithm should guarantee that there are a handful of remaining chunks left in the buffer after each chunk playing.*

Insight 3. *Under rate-based algorithm, simply increasing available bandwidth does not necessarily reduce the rebuffering events.*

Figure 4 indicates that what really matters is the average chunk download time μ. When C goes up, the rate-based algorithm may pick a higher bitrate level R_{i+1}, then even if $C\mu = VR_{i+1}$ grows, $\mu = \frac{VR_{i+1}}{C}$ may also decline. Moreover, as stated above, the larger the bitrate level, the larger σ, which further aggravates rebuffering.

Recommendation 3. *In consideration of avoiding rebuffering, we should not blindly increase link capacity, but cautiously select the level not too close to the available bandwidth.*

C. Influence of Chunk Size on Video Quality

In this subsection, we study the effect that chunk size variation has on video quality. Here we choose to compare two expectations $E[R]_c$ and $E[R]_0$ – expected chosen bitrate level with and without considering actual chunk size respectively.

Without loss of generality, we assume that available bandwidth obeys the uniform distribution between C_{min} and C_{max}, i.e., $C \sim U(C_{min}, C_{max})$. We let $R_1 = C_{min}$ and fiction a variable $R_{m+1} = C_{max}$, then $R_2, \cdots, R_m$ are all in the range (C_{min}, C_{max}). If the adaptation algorithm does not consider chunk size variation, the selected level can be computed using the following equation.

$$E[R]_0 = \frac{1}{C_{max} - C_{min}} \sum_{i=1}^{m} (R_{i+1} - R_i) R_i \tag{3}$$

c When C is between R_i and R_{i+1} whose probability is $\frac{R_{i+1}-R_i}{C_{max}-C_{min}}$, the client would choose R_i. Adding up all possible R values multiplied by its probability produces the expectation. On the other hand, if the algorithm takes chunk size variation into account, the choice should be according to the actual chunk sizes. Let the instantaneous size of chunk k in level i be $S_i(k)$, we choose level R_i if and only if $i = \max_{1\le j\le m} \{j, S_j(k) \le CV\}$

Then we calculate the expectation $E[R]_c$. Here we newly define another set of standard deviations $\Sigma' = \{\sigma'_1, \sigma'_2, \cdots, \sigma'_m\}$, with σ'_i denoting the standard deviation of $\frac{S_i}{V}$. Note that σ'_i means the standard deviation of level i's instantaneous bitrate, which is different from the above defined σ_i denoting that of download time. In this context, we could rewrite the distribution of S_i to be $S_i \sim N(R_iV, V^2\sigma_i'^2)$, $i = 1, 2, \cdots, m$. Given $C = c$, all possible selected bitrate levels as well as their possibilities are listed as follows.

- For R_1, it is picked if and only if $S_2 > cV$, whose probability is $1 - \Phi(\frac{c-R_2}{\sigma'_2})$
- For $R_i = R_2, \cdots, R_{m-1}$, it is picked if and only if $S_i \le cV \wedge S_{i+1} > cV$, whose probability is $\Phi(\frac{c-R_i}{\sigma'_i}) - \Phi(\frac{c-R_{i+1}}{\sigma'_{i+1}})$
- For R_m, it is picked if and only if $S_m \le cV$, whose probability is $\Phi(\frac{c-R_m}{\sigma'_m})$

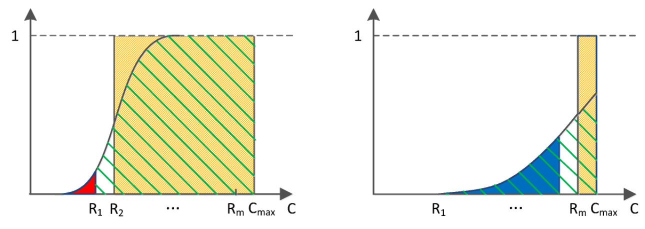
After obtaining probabilities, we can calculate the conditional expectation $E[R|C = c]$ with the following equation.

$$\begin{aligned}E[R|C=c] =& R_1\left(1-\Phi\left(\frac{c-R_2}{\sigma'_2}\right)\right) + R_m\Phi\left(\frac{c-R_m}{\sigma'_m}\right)\\ &+\sum_{i=2}^{m-1} R_i\left(\Phi\left(\frac{c-R_i}{\sigma'_i}\right)-\Phi\left(\frac{c-R_{i+1}}{\sigma'_{i+1}}\right)\right)\\ =& R_1+\sum_{i=2}^{m}\Phi\left(\frac{c-R_i}{\sigma'_i}\right)(R_i-R_{i-1})\end{aligned}$$

To compute $E[R]_c$, we only need to take an expectation of $E[R|C = c]$ over the randomness of C values. Considering $C \sim U(R_1, R_m)$, we have

$$\begin{aligned}E[R]_c &= \frac{\int_{C_{min}}^{C_{max}}\left[R_1+\sum_{i=2}^{m}\Phi\left(\frac{c-R_i}{\sigma'_i}\right)(R_i-R_{i-1})\right]dc}{C_{max}-C_{min}}\\ &= R_1+\frac{\sum_{i=2}^{m+1}(R_i-R_{i-1})\int_{C_{min}}^{C_{max}}\Phi\left(\frac{c-R_i}{\sigma'_i}\right)dc}{C_{max}-C_{min}}\end{aligned} \tag{4}$$

Next, we should compare $E[R]_c$ and $E[R]_0$ to observe the effect that considering chunk size variation has on the average bitrate. Note that when $\sigma'_1 = \sigma'_2 = \cdots = \sigma'_m = 0$, considering



(a) R_2 (b) R_m

Fig. 5. Improved average bitrate

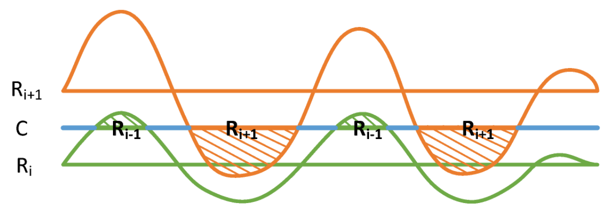


Fig. 6. Schematic diagram of bitrate

chunk size variation makes no difference on bitrate level selection. In this case, $E[R]_c$ degenerates to $E[R]_0$.

$$\begin{aligned} E[R]_c &= R_1 + \frac{1}{C_{max} - C_{min}} \sum_{i=2}^{m+1} (R_i - R_{i-1}) (R_{m+1} - R_i) \\ &= R_1 + \frac{1}{C_{max} - C_{min}} \sum_{i=2}^{m+1} (R_i - R_{i-1}) (R_{i-1} - R_1) \\ &= \frac{1}{C_{max} - C_{min}} \sum_{i=1}^{m} (R_{i+1} - R_i) R_i = E[R]_0 \end{aligned}$$

When chunk sizes really fluctuate, that is, $\sigma'_i > 0, i = 1, 2, \cdots, m$, we next demonstrate that $E[R]_c >= E[R]_0$. We first assume $R_2, R_3, \cdots, R_m$ are evenly spaced between C_{min} and C_{max}, and then discuss the influence of bitrate level spacing.

As above stated, when $\sigma'_1 = \sigma'_2 = \cdots = \sigma'_m = 0$, $E[R]_c = E[R]_0$. And as these σ' grow, Equation 4 implies that the two expectations mainly differ on both the first and the last terms $\int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_2}{\sigma'_2})dc$ and $\int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_m}{\sigma'_m})dc$. Figure 5 respectively compares these two terms. In subgraph 5(a), the s-shaped curve depicts the function $\int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_2}{\sigma'_2})dc$ when $\sigma'_2 > 0$ and the green shaded area represents its integral from C_{min} to C_{max}. When $\sigma'_2 = 0$, the function degrades to the step function $u(t - R_2)$, hence the yellow area denotes its integral from C_{min} to C_{max}. In this way, the small red area is right the difference between the two integrals. On this part, $E[R]_0$ is a little larger. However, on the other integral term, Subgraph 5(b) shows that $E[R]_c$ is much larger than $E[R]_0$. Similar to Subgraph 5(a), the yellow area denotes the integral of function $u(t - R_m)$ from C_{min} to C_{max}; While the green shaded area describes $\int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_m}{\sigma'_m})dc$ when $\sigma'_m > 0$, thus the blue area describes the value by which $E[R]_c$ is larger than $E[R]_0$ on this part. Then leveraging the property that $\sigma'_m > \sigma'_2$ stated in Section II, the blue area must be larger than the red one. Besides these two integral terms, some other ones that are close to the edge of integration domain are also likely to produce this kind of difference. Nevertheless, due to the fact that the larger R_i, the larger σ'_i, $E[R]_c$ is definitely larger than $E[R]_0$.

It calls for attention that this simple way of considering actual chunk size is even too conservative. Since there is a playback buffer storing downloaded but not played chunks, it can make up for the difference between playback rate and available bandwidth. Because each time the instantaneous rate of the selected chunk is below or equal to the available bandwidth, chunks will continuously accumulate in the buffer. When the buffer backlog is large enough, a higher level with instantaneous bitrate larger than available bandwidth could be safely chosen and the average bitrate could further increase.

In essence, involving chunk size variation into bitrate adaptation chooses lower level when chunk size is larger than CV, while is likely to choose higher ones in remaining cases. Therefore, the average bitrate does not decline under the condition of no rebuffering.

Then we discuss the effect of bitrate level spacing on $E[R]_c$. To this end, on the condition that $R_1 = C_{min}$, we calculate the partial derivative $\frac{\partial E[R]_c}{\partial R_i}, i = 2, 3, \cdots, m$. Since besides R_1, R_i only appears in $\frac{R_i - R_{i-1}}{C_{max} - C_{min}} \int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_i}{\sigma'_i})dc$ as well as $\frac{R_{i+1} - R_i}{C_{max} - C_{min}} \int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_{i+1}}{\sigma'_{i+1}})dc$, we have

$$\begin{aligned} \frac{\partial E[R]_c}{\partial R_i} =& \frac{\int_{C_{min}}^{C_{max}} \left(\Phi\left(\frac{c-R_i}{\sigma'_i}\right) - \Phi\left(\frac{c-R_{i+1}}{\sigma'_{i+1}}\right) \right) dc}{C_{max} - C_{min}} \\ & - \frac{R_i - R_{i-1}}{C_{max} - C_{min}} \int_{C_{min}}^{C_{max}} \frac{1}{\sqrt{2\pi}\sigma'_i} e^{-\frac{(c-R_i)^2}{2\sigma'^2_i}} dc \end{aligned} \tag{5}$$

As for R_m, because $R_{m+1} = C_{max}$ and $\sigma'_{i+1} = 0$, we can get $\int_{C_{min}}^{C_{max}} \Phi(\frac{c-R_{m+1}}{\sigma'_{m+1}})dc = 0$, in which case Equation 5 still applies to R_m. For most R_i that are not too close to C_{min} or C_{max}, suppose R_{i-1} and R_{i+1} are given, we have $\frac{\partial E[R]_c}{\partial R_i} = \frac{R_{i+1} - 2R_i + R_{i-1}}{C_{max} - C_{min}}$. In this case, $\frac{\partial E[R]_c}{\partial R_i} = 0$ when $R_i = \frac{R_{i-1} + R_{i+1}}{2}$ and $E[R]_c$ is maximized at this point. Since the deduction applies to most R_i, a simple but effective method is right to evenly divide the range $[C_{min}, C_{max}]$ to the m levels.

Under evenly spaced bitrate levels, Figure 6 approximately illustrates the improved average bitrate selection. With horizontal axis indicating time and vertical axis implying bitrate, the two curves separately describe the actual chunk rate with level R_i and R_{i+1}. When $C = \frac{R_i + R_{i+1}}{2}$ as the blue line shows, the green part represents R_{i-1} is chosen, while the orange part denotes selecting R_{i+1}. In other times, we still choose R_i. Because $\sigma'_{i+1} > \sigma'_i$, the orange part evidently accounts for more time than the green one, and then on average the selected bitrate is larger than R_i. Even if $C < \frac{R_i + R_{i+1}}{2}$, there are still a C range in which the selected bitrate is larger than R_i on time average. Considering $C \sim U(C_{min}, C_{max})$, the time-averaged bitrate is enhanced with more than half of the probability.

Algorithm 1 An improved algorithm

```
Input: $\mathcal{S}$ = Set of chunk sizes of the next chunk of all $m$
    bitrate levels, $B$ = Current buffer level, $C$ = Next available
    bandwidth
Output: $L$ = Selected bitrate level
1: $target\_size \leftarrow C(B - 3V)$
2: for each $S_i \in \mathcal{S}$, $i = m \to 1$ do
3:     if $S_i \leq target\_size$ then
4:         break
5:     end if
6: end for
7: $L = i$
8: return $L$
```

Insight 4. *Taking chunk sizes variation into consideration improves the average video quality on the premise of no rebuffering events.*

Recommendation 4. *If the bitrate adaptation algorithm considers chunk size variation, under fixed number of bitrate levels, a perfect method is right to equally divide the range $[C_{min}, C_{max}]$ to all the bitrate levels.*

IV. A Simple Design

Based on above insights and analysis, we naturally propose an improved bitrate adaptation algorithm that is both chunk size- and start time- aware. Under the new algorithm, each time the scheduler selects the highest bitrate whose instantaneous chunk size will make at least 4 chunks left in the buffer after downloading. Given the current buffer backlog B, the chunk period V and the following available bandwidth C, the target chunk size should satisfy:

$$B - \frac{target\_size}{C} + V = 4V$$

and we can get $target\_size = C(B - 3V)$. In practice, we choose the highest possible bitrate whose chunk size is less than or equal to the target value. The detailed algorithm is shown in Algorithm 1.

The algorithm leverages the advantage of both chunk size awareness and start time awareness. According to our modeling and analysis, such a practice will simultaneously reduce rebuffering events and improve the average video quality. In the next section, we will conduct evaluation on this improved algorithm.

V. Evaluation

In this section, we present our simulation setup and results. For the insights as well as recommendations proposed in Section III, we conduct emulated validations respectively. The simulation results show both the correctness of the insights and the effectiveness of our recommendations and design.

A. Simulation Setup

Bandwidth trace Although we assume the constant available bandwidth in our modeling, in our simulations, we use the throughput trace measured in [20]. The trace records 1s of available bandwidth information when a DASH system is running over 3G cellular networks. This adoption makes the setup closer to the actual conditions and leads to more real simulation results.

TABLE I
Overview of DASH dataset

Name	Length	Genre
Big Buck Bunny	00:09:46	Animation
Elephants Dream	00:10:54	Animation
Red Bull Playstreets	01:37:28	Sport
The Swiss Account	00:57:34	Sport
Valkaama	01:33:05	Movie
Of Forest and Men	00:10:53	Movie

TABLE II
Overview of DASH dataset

Name	Rebuffering Count		Average Bitrate	
	Classic	Enhanced	Classic	Enhanced
Big Buck Bunny	2	2	225473.9	232096.8
Elephants Dream	15	2	226632.4	235646.7
Red Bull Playstreets	113	111	153720.4	154730.2
The Swiss Account	77	63	170361.3	180505.5
Valkaama	61	44	149153.0	168747.7
Of Forest and Men	3	3	227279.3	238259.7

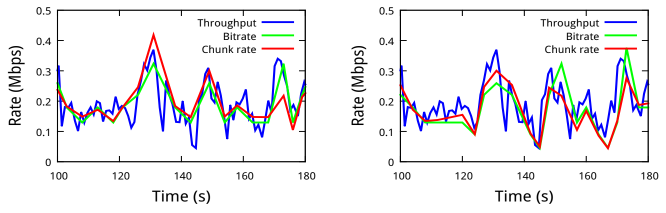
Video data set We adopt the DASH dataset provided in [21]. This dataset is publicly available and includes 6 video of 3 types and encoded at multiple bitrate levels using VBR encoding, and the video information is listed in table I. In our simulations, we choose the videos with chunks of 4 seconds.

Performance metrics Consistent with the two metrics we focus on in Section III, we evaluate the *number of rebuffering events* and *average bitrate level* in our actual simulations.

B. Influence of Chunk Size

Firstly, we evaluate the influence of chunk size variation. With the 6 videos encoded with VBR encoding, we respectively perform the classic rate-based algorithm (not concerning chunk sizes) and an enhanced one (considering chunk sizes) using the same bandwidth trace. The rebuffering counts and the average bitrate under the two algorithms are both shown in table II.

The results can be illustrated from two aspects. Firstly, if we only concern the two columns of rebuffering counts, we can see that under the classic algorithm the number of rebuffering events of any video is larger than or equal to that caused by too low bandwidth, which is listed in the “Enhanced” column. In other words, besides those inevitable rebuffering events caused by network breakdown, there are still quite a part resulting from the fluctuating chunk sizes, which should not take place if chunk sizes do not variate. In addition, it can be observed that taking into account actual chunk sizes indeed eliminates the rebuffering by chunk size variation, then Recommendation 1 is proven to be effective. On the other hand, when it comes to the video quality, the average bitrates for all 6 videos become higher under the enhanced algorithm of considering actual chunk sizes, which also corresponds to our insights. Note that the this improvement occurs on the condition of reduced rebuffering counts, thus is truly meaningful.



(a) Classic algorithm (b) Enhanced algorithm

Fig. 7. A typical illustration

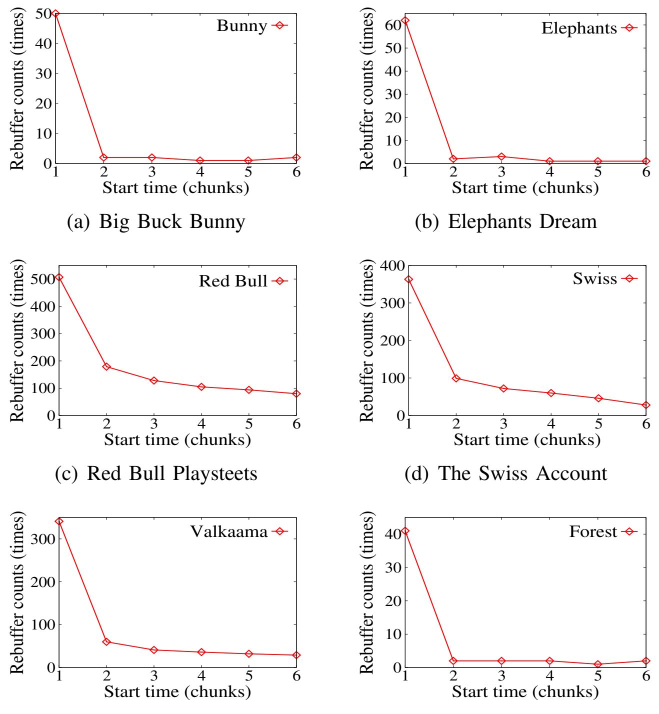
Then we choose a typical video "Elephants Dream" to detailedly demonstrate how considering chunk sizes reduces rebuffering events and improves the average bitrate. Figure 7 shows available bandwidth (blue line), selected bitrates (green line), as well as actual chunk rates (red line) of both two rate adaptation algorithms in the time interval $[100s, 180s]$. Compared with the classic algorithm, we can see when the actual chunk rate is larger than the nominal value, the enhanced algorithm chooses a lower bitrate to avoid rebuffering (the first peak on the graph); while if the actual chunk rate is smaller, a higher bitrate is selected to improve the video quality (the last peak on the graph). In this context, the actual chunk rate is always below but very close to the available bandwidth, indicating the good performance of involving chunk size into rate adaptation.

C. Importance of start time

In this part, we mainly address the effect of start time on the basis of involving actual chunk sizes. To this end, we conduct a series of simulations with different start time and observe the algorithm performance. Here the start time means how many downloaded chunks are already in the buffer before each chunk download. To achieve this, we adopt our newly designed algorithm and vary the number of chunks reserved after downloading. According to previous analysis that a little start time takes a great effect, we choose the start time from 1 chunk to 6 chunks. The results are illustrated in Figure 8 and Figure 9.

Figure 8 respectively shows the rebuffering counts under different start time for all 6 videos. It can be seen that for all videos, raising the start time makes a great contribution to eliminating rebuffering events, especially when it increases from 1 chunk to 2 chunks. The three short videos ("Big Buck Bunny", "Elephants Dream" and "Of Forest and Men") achieves near-optimal performance (2 rebuffering events) when the start time equals 2, while the three long videos ("Red Bull Playsteets", "The Swiss Account", and "Valkaama") see successive large declines in rebuffering times as the start time increases from 1 chunk to 6 chunks.

On the other hand, Figure 9 depicts the average bitrates on different start time conditions. We can see when it comes to video quality, different video presents different properties. Nevertheless, we can still observe the overall descending trend, which is intuitive because the more chunks are reserved in the buffer, the more conservative the algorithm will be. Moreover,



(a) Big Buck Bunny (b) Elephants Dream

(c) Red Bull Playsteets (d) The Swiss Account

(e) Valkaama (f) Of Forest and Men

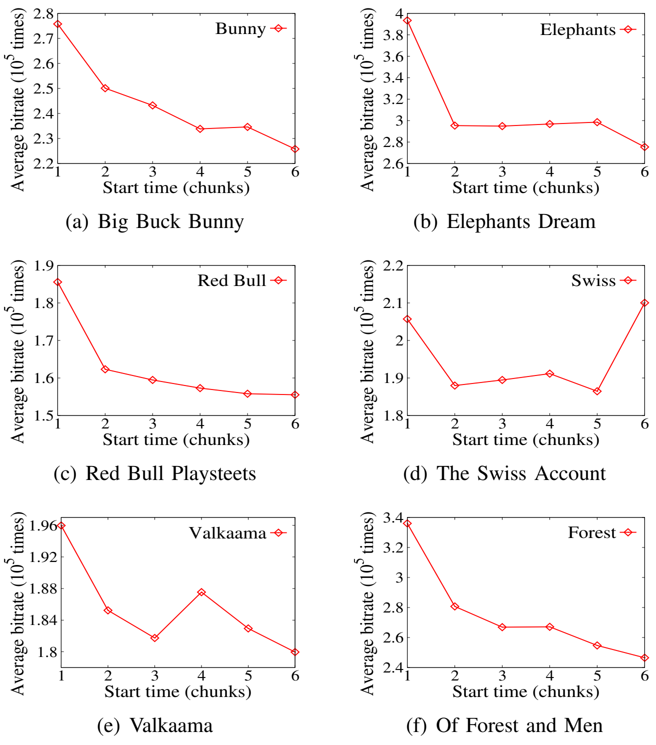
Fig. 8. A typical illustration

when the start time varies from 1 chunk to 2 chunks, the decrease in average bitrate is obvious, while as the start time further increase, the average bitrates decline less sharply or do not decline at all. In video "The Swiss Account" and "Valkaama", there are even cases where the average bitrate improves as the start time goes up. These exceptions are because more reserved chunks will promote larger subsequent bitrates, which contributes to the higher average value as well. In summary, when the start time is within certain limits, its increase will not greatly impact the video quality, exhibiting the good property of involving start time.

D. Improved algorithm

In our design, we set the start time to 4 chunks. From Figure 8 and Figure 9, we can see when the start time is equal to $4V$, the rebuffering times have been reduced to a low level, while the average bitrate sees little drops relative to the condition when start time equals 2 chunks. What's more, it is important to note that under our design, both the number of rebuffering events and the average bitrate are superior to that of the two basic algorithms shown in Table II.

On the other hand, since our designed algorithm also observes the buffer occupancy each time making a decision, we compare our design with a buffer-based algorithm [15] for fairness. The results are depicted in Table III. We can see that for three short videos both algorithms achieve very few rebuffering events, while for other three long videos our designed algorithm greatly reduces the rebuffering times relative to the buffer-based algorithm. Furthermore, except



(a) Big Buck Bunny (b) Elephants Dream (c) Red Bull Playsteets (d) The Swiss Account (e) Valkaama (f) Of Forest and Men

Fig. 9. A typical illustration

TABLE III
COMPARISON WITH BUFFER-BASED ALGORITHM

Name	Rebuffering Count		Average Bitrate	
	BBA	Improved	BBA	Improved
Big Buck Bunny	1	1	218197.4	231676.8
Elephants Dream	1	1	217936.5	268330.2
Red Bull Playstreets	176	108	158103.1	156417.5
The Swiss Account	90	62	173691.8	178272.6
Valkaama	45	38	157444.2	180553.2
Of Forest and Men	1	1	221487.8	257505.6

for the video “Red Bull Playstreets”, our designed algorithm improves the average bitrate of other 5 videos to different extents. Even for “Red Bull Playstreets”, our design cuts down 38.6% of rebuffering events at the cost of only 1.1% of average bitrate loss, still indicating the benign overall performance.

VI. Conclusions

In this paper, we theoretically analyze the influence of varying chunk size on bitrate adaptation algorithm performance. Firstly, we establish a model to describe the playback buffer evolution under bitrate adaptation in dash systems. Then based on the model and considering the user experience, we respectively address the rebuffering probability and the average bitrate level and quantitatively analyze what effect the chunk size variation has on these two metrics. Analysis results show that the varying chunk size indeed impacts the bitrate adaptation performance. Instead of simply increasing available bandwidth, involving the actual chunk size and a little start time into the adaptation algorithm is of great help to improving the two metrics. Furthermore, based on obtained insights, we provide several recommendations for algorithm designing and rate encoding and design a simple algorithm. Extensive simulations verify our analysis results and indicate that our designed algorithm has superior performance.

VII. Acknowledgments

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by 863 Plan under Grant No. 2015AA020101, and NSFC under Grant No. 61225011, and Suzhou-Tsinghua Special Project for Leading Innovation.

References

[1] *Smoothstreaming protocol.* http://go.microsoft.com/?linkid=9682896.
[2] A. Zambelli, “Iis smooth streaming technical overview,” *Microsoft Corporation*, 2009.
[3] *Mail service costs Netflix 20 times more than streaming.* http://goo.gl/msuYK.
[4] R. Pantos and W. May, “Http live streaming draft-pantos-http-live-streaming-05,” *IETF*, 2011.
[5] *Adobe http dynamic streaming.* www.adobe.com/products/hds-dynamic-streaming.html.
[6] *Akamai hd adaptive streaming.* http://wwwns.akamai.com/hdnetwork/demo/index.html.
[7] I. Sodagar, “The mpeg-dash standard for multimedia streaming over the internet,” *IEEE MultiMedia*, 2011.
[8] T.-Y. Huang, R. Johari, and N. McKeown, “Downton abbey without the hiccups: Buffer-based rate adaptation for http video streaming,” in *ACM SIGCOMM*, 2013.
[9] 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,” in *ACM SIGCOMM*, 2011.
[10] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, “Confused, timid, and unstable: picking a video streaming rate is hard,” in *ACM IMC*, 2012.
[11] *Sandvine: Global Internet Phenomena Report 2013 H2.* http://tinyurl.com/nt5k5qw.
[12] L. Popa, A. Ghodsi, and I. Stoica, “Http as the narrow waist of the future internet,” in *ACM SIGCOMM*, 2010.
[13] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over http,” in *ACM SIGCOMM*, 2015.
[14] H. H. Liu, Y. Wang, Y. R. Yang, H. Wang, and C. Tian, “Optimizing cost and performance for content multihoming,” in *ACM SIGCOMM*, 2012.
[15] 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*, 2015.
[16] C. Liu, I. Bouazizi, and M. Gabbouj, “Rate adaptation for adaptive http streaming,” in *ACM MMSys*, 2011.
[17] J. Jiang, V. Sekar, and H. Zhang, “Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive,” in *ACM CoNEXT*, 2012.
[18] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, “Elastic: a client-side controller for dynamic adaptive streaming over http (dash),” in *Packet Video Workshop (PV)*, 2013.
[19] G. Tian and Y. Liu, “Towards agile and smooth video adaptation in dynamic http streaming,” in *ACM CoNEXT*, 2012.
[20] *HSDPA dataset.* http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs.
[21] S. Lederer, C. Müller, and C. Timmerer, “Dynamic adaptive streaming over http dataset,” in *MMSys*, 2012.