OPTIMIZING HTTP-BASED ADAPTIVE VIDEO STREAMING FOR WIRELESS ACCESS NETWORKS

Xiaoling Qiu[1], Haiping Liu[3], Deshi Li[2], Song Zhang[2]
Dipak Ghosal[1], Biswanath Mukherjee[1]

1 Computer Science Department, University of California, Davis, USA
2 School of Electronic Information ,Wuhan University, China
3 Department of Electrical and Computer Engineering, University of California, Davis, USA
{xqiu,hpliu}@ucdavis.edu, dsli@whu.edu.cn, zhangsongwhu@163.com, {ghosal,mukherje}@cs.ucdavis.edu

Abstract

In this paper, we propose an optimization algorithm called Intelligent Bitrate Switching-based Adaptive Video Streaming (ISAVS) to address the key issue of bitrate switching in HTTP-based adaptive video streaming. Our proposed solution can provide the best possible video quality to users with minimum replay interruption, which is reported as one of the key video quality issues of on-line video streaming over wireless access networks. While the optimization algorithm is based on IIS Smooth Streaming architecture [1], it can be easily applied by other proposed approaches, such as Adobe Flash Dynamic Streaming, and Apple HTTP Adaptive Bitrate Streaming. Simulations show that our optimization algorithm selects the best video bitrate according to channel condition and the amount of video stream buffered in the client to provide viewers the best possible quality while reducing the number of interruptions.

Keywords: HTTP-based Video Streaming; Smooth Streaming; Adaptive Video Streaming; Optimization

1 Introduction

Clearly, on-line video streaming is becoming a critical part of today's Internet applications. People prefer receiving information not only by traditional plain text and still images, but High Definition (HD) video streaming. Some video-based social networks, such as YouTube [2] and Tudou [3], have become popular websites for people to share information. By the end of 2008, video search on YouTube accounted for a quarter of all Google search queries in the United States [12], illustrating the popularity of video in the Internet. Video advertisement has also become an important method of online advertisement to convey large and personalized information in a short period of time. Some TV or movie websites, like Hulu [4] and Netflix [5] offer TV programs and movies on-demand, and it is quickly becoming a more economic way for watching TV. Moreover, with the unbeatable and stunning image quality of HD video, video over the Internet services are becoming popular. It is expected that total HD-related online video revenue in the United States will be over $2.2 billion per year by 2012 [11].

However, video streaming applications, especially HD videos, pose many challenges for network designers and providers because of the requirements of high bandwidth and small delay. These challenges are exacerbated in wireless access networks 1) due to the unstable and time-varying wireless channel transmission rate, 2) due to contention for the relatively lower wireless channel bandwidth, and 3) due to congestion in the back-haul network. To address these problems, video providers often downgraded the video quality to the least common denominator by serving at the quality corresponding to the customer with lowest condition. In this case, customers with high bandwidth connections suffer from the lower video quality. Furthermore, even if the video providers provide several video bitrate choices, due to the dynamic nature of the wireless channel, the choice of the video bitrate may not be the best choice during video replay. As a result, adaptive video streaming has emerged as a solution to provide the best available video quality for customers connected either by relatively stable wired connections with different bandwidth or by dynamically changing and unpredictable wireless channels. Adaptive video streaming means that video bitrate is switched on-the-fly to provide the best video quality to the user based on the available bandwidth in the network and resources in the end device. Recently, there have been a number of proposals from a number of leading industries including Microsoft's Smooth Streaming [1], Adobe's Flash Dynamic Streaming [6], and Apple's HTTP Adaptive Bitrate Streaming [7]. All these methods are HTTP-based and these have many advantages over past generation video streaming technologies.

978-1-4244-6769-3/10/$26.00 ©2010 IEEE

One of the key challenges for adaptive video streaming is how and when to adaptively change video bitrates according to network condition and client-side buffer so as to provide the best video quality for users [1]. All the existing methods use heuristic bitrate switching strategies without optimization. For example, IIS (Internet Information Services) Smooth Streaming [1] makes use of the real-time available network bandwidth information to choose the video bitrate. This, however, as shown later, may not guarantee the best video quality for users. In this paper, we propose an optimization algorithm called Intelligent Bitrate Switching-based Adaptive Video Streaming (ISAVS) to provide the best video quality. While our proposed algorithm is based on the IIS Smooth Streaming architecture, it can be easily adopted in other adaptive video streaming technologies, such as Flash Dynamic Streaming [6], and HTTP Adaptive Bitrate Streaming [7]. To the best of our knowledge, there is no other study providing similar solutions to optimize video bitrate selection for these HTTP-based adaptive video streaming algorithms. One of most related works in [20] discusses in determining the required size of the video receiving buffer over TCP.

Compared with IIS Smooth Streaming method proposed in [1], the major contributions of our algorithm are as follows:

• IIS Smooth Streaming utilizes the real-time information on the available network bandwidth as the decision variable to select the video quality level. Our mechanism uses both real-time and historical information. In addition to monitoring the available network bandwidth, we also monitor the length of the buffered video in the client replay buffer.

• When utilizing the real-time information, the IIS Smooth Streaming only focuses on the average effective transmission rate[1]. However, from our analysis, variance in the available network bandwidth has a significant impact on the received video quality. Without considering the variance, the IIS Smooth Streaming algorithm may either select the inferior video category due to the temporary low transmission rate, or select good video category but cause the replay to stall due to the temporary high transmission rate. In wireless networks, considering the variance in the transmission rate yields significant improvement towards optimizing the streaming video quality.

• The heuristic algorithm of IIS Smooth Streaming cannot provide the detailed analysis to determine the optimal choice of the video categories. In our mechanism, we provide an optimization algorithm to achieve the best video quality, by choosing the appropriate video categories and consequently reducing the number of video replay stalls.

• In order to support our proposed video streaming algorithm, we propose (1) practical measurement methods to determine the network condition and (2) an efficient procedure to solve the optimization problem for the best choice of the video quality level.

The rest of the paper is organized as follows. We introduce the architecture and strategy of HTTP-based video streaming in Section 2. In Section 3, we formulate the problem and provide a mathematical model to obtain the optimum video bitrate. Section 4 presents the simulation setup and performance evaluation. Finally, we draw conclusions in Section 5.

2 Video over-the-top: HTTP-based video streaming

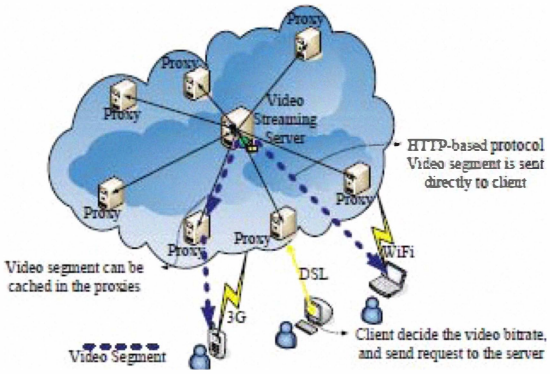

Figure 1. A generic HTTP-based video streaming architecture

Client-server based video delivery method has experienced three phases of development: 1) Real-Time Streaming Protocol [9] (RTSP)-based traditional streaming, 2) progressive download streaming and 3) HTTP-based adaptive streaming [1]. The IIS Smooth Streaming [1], which enables adaptive streaming of media to Silverlight using HTTP-based adaptive video streaming. Fig.1 shows the architecture of IIS Smooth Streaming. It is based on the client-server application model. In the video server, a media file is encoded to several files with different bitrates. A higher bitrate means lower video compression ratio and contains more video information and thus higher video quality. Therefore, each video bitrate represents a level of video quality. Each file is conceptually encoded as a series of fragments. A fragment is "an independently downloadable unit of media that comprises one or more smallest fundamental units, such as a frame"[8]. The encoding is naturally supported by ISO/IEC MP4 media file specification [10] since a MP4 file is composed as a series of short metadata-data pairs. A metadata-data pair can be treated as a fragment in the IIS Smooth Streaming architecture. A fragment is

usually a 2-second video (which is also the default value in IIS Smooth Streaming). Fragment is cut along the boundary of Group of Pictures (GOP) so that each fragment can be independently decoded without information of the past/future fragment. The client determines whether to download larger or smaller fragments, and send the request to the server based on the estimate of the network bandwidth, CPU capability, and screen resolution. Note that the client decides the level of video after a new segment is received. From Fig. 1, we can see that the video fragment can be further cached in an already setup Web proxy to relieve backbone network congestion and decrease delay. Note that the different levels or pieces of video segments must be synchronized by the server. In order to provide the quick start of viewing experience, the client always requests for the lowest level of video at the beginning and decides the higher video quality levels according to the available network bandwidth.

The key challenge of Smooth Streaming is the strategy of the client of how and when to request for the optimized video bitrate. Current Smooth Streaming uses a heuristic method: the client-side examines the real-time channel bandwidth, CPU capability and screen resolution after receiving a fragment, and then decides to request higher or lower bitrates for the next fragment from the server [1]. However, the heuristic approach has disadvantages. For example, when the bandwidth of network is not good, original Smooth Streaming may decide to request the lower bitrates from the server. It may be a correct decision when there are only few cached fragments in local buffer. However, if there are a large number of cached fragments in local buffer, the client can still request for higher bitrates to give better video experience for users since the sufficient buffered video can compensate the low network bandwidth for a period of time. On the contrary, the client should keep requesting for lower bitrates even when the network bandwidth is temporary high since there is few number of buffered fragments. Or the video may potentially stop replay due to consuming all buffered fragments and net-work bandwidth becoming low. Interrupted video can greatly degrade the video viewing experience. (A research survey of more than 2,300 online consumers shows that interrupted replay is the biggest reason for users' frustration for on line video experience [11].) Therefore, we can conclude that the cached fragments in local buffer should be taken into account when making the bitrate switching decision. Note that in this paper, we use the original strategy based on local CPU capability and screen resolution. Specifically, our optimization is based on the subset of all available video bitrates, which is gained by examining local CPU capability and screen resolution. We mainly concentrate on the more challenge part of how to improve video replay quality based on the accurate network bandwidth estimation.

3 Optimization algorithm

3.1 Problem formulations

Table 1 Key notation in the formulation

Notation	Definition
m	Number of video quality levels
B_i	Benefit for video quality level i
I	Freeze time /video stop replay time
P_0	Unit penalty for freeze time
T_c	Fragment inter-arrival time
W_b	Number of fragments in the buffer when a new fragment is received
T_R	Reciprocal of video replay rate

Our optimization algorithm is based on the following assumptions.

• Video to be streamed is compressed into m quality levels, where $m \in N$. A typical value is $m \le 10$.

• Suppose the "benefit" of the i th quality level is B_i, where each video quality level i has benefit B_i and fragment size denoted by C_i, $i = 1, 2, ..., m$. We assume that larger fragment size have higher quality and hence higher benefit. The method to determine the value of B_i is discussed in Section III-D.

• When the available network bandwidth is smaller than the video replay rate and the client buffer becomes empty, the video replay will STALL resulting in a video freeze. Let the random variable I denote the length of time when the video replay stalls, referred to as STALL/Freeze Time.

• We assume the unit "Penalty" for video STALL/Freeze Time as P_0, which is a negative value.

Based on the above assumptions and notations, we can write the following optimization function

$$\begin{aligned} &maximize_{i=1,2,\ldots,m}\{Benefit + Penalty\} \\ &= maximize_{i=1,2,\ldots,m}\{B_i + P_0 E[I]_i\} \end{aligned} \quad (1)$$

where i is the decision variable of quality level and $E[I]_i$ is the average freeze time. In Eq.1, there are m video quality levels available for the next video fragment to choose from. The optimization algorithm will choose quality level i that can maximize the quality benefit and has the least STALL penalty after the previous fragment is received.

In order to derive $E[I]_i$, we make the following assumptions:

• Let the random variable T_c denote the fragment inter-arrival time and T_{ci} denotes the inter-arrival time of the ith chunk. As shown in Fig. 2, T_{c1} is the inter-arrival time of the first fragment. The probability density function of T_c is denoted by $f_{T_c}(t)$. We will discuss how to derive the $f_{T_c}(t)$ in subsection III-B.

• Let the random variable W_b denote the number of fragments in the buffer at the application layer at the beginning of the inspection time, which is the time when a new fragment is received in the client.

• We assume that the video streaming replay rate is constant and denoted by $1/T_R$ fragments per seconds. Typical value of T_R is 2 seconds.

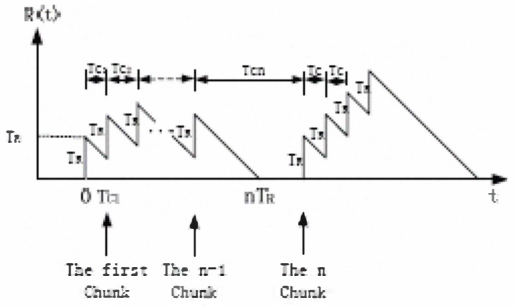

Figure 2. An example of R(t): the amount of time to finish replaying fragments in the buffer.

We could analyze the following two cases to derive the optimization algorithm:

1) If $T_c < W_b T_R$, which means a new fragment arrives before the buffer is empty. Then, $Freeze\ Time = 0$ and Benefit is B_i, where $i \in [0, m]$.

2) If $T_c \geq W_b T_R$, which means a new fragment arrives after the video buffer is empty. In this case,

$$Freeze\ Time = T_c - W_b T_R, \quad (2)$$

$$Benefit = B_i, \quad (3)$$

$$Penalty = P_0 (T_c - W_b T_R). \quad (4)$$

We can also derive the average freeze time

$$E[I]_i = \int_{W_b T_R}^{\infty} (t - W_b T_R)[f_{T_c}(t)]_i dt. \quad (5)$$

where $[f_{Tc}(t)]_i$ is the probability density function of inter-fragment time for the i th quality level, $i = 1, 2, 3, ..., m$. Furthermore, the average penalty is given by

$$\begin{aligned} E[P]_i &= E[P_0 I]_i = P_0 E[I]_i \\ &= P_0 \int_{W_b T_R}^{\infty} (t - W_b T_R)[f_{T_c}(t)]_i dt. \end{aligned} \quad (6)$$

By combining Eq. 1 and Eq. 6, the optimization function is given by

$$\max_{i=1,2,...,m} \{B_i + P_0 \int_{W_b T_R}^{\infty} (t - W_b T_R)[f_{T_c}(t)]_i dt\}. \quad (7)$$

3.2 Estimation of the probability density function of the inter-arrival time of fragments

The probability density function of the inter-arrival time of fragments $f_{T_c}(t)$ for a given quality level i is one of the key inputs of the optimization algorithm. This is obtained from measurements of the inter-arrival time of fragments. While for long videos, we can collect enough samples, for some short videos, like a 20-second video, which has only 10 fragments with 2-second replay time, there may not be enough samples to get an accurate estimate of $f_{T_c}(t)$. The small number of samples of T_c makes the estimation of $f_{T_c}(t)$ inaccurate. Therefore, we estimate the $f_{T_c}(t)$ based on the TCP segments in a fragment rather than the fragment itself. The payload size of a TCP segment is usually 1460 bytes, which equals to the standard Maximum Transmission Unit (MTU) 1500 bytes with the TCP header 20 bytes and IP header of 20 bytes. At the application layer, a 2-second video fragment is typically comprised of 30 to 300 TCP segments, which gives enough number of samples to estimate the $f_{T_c}(t)$.

Suppose a video fragment contains S TCP segments, each of which has a transmission time t_i. When the first TCP segment arrives, we can estimate the $T_c = S * t_1$ based on this specific sample, so $P[T_c = S * t_1] = 1/S$. After collecting all S segments, we have $\forall i, P[T_c = S * t_i] = 1/S$. In order to integrate the data for the estimation of $f_{T_c}(t)$, the following methods can be adopted.

1) To save the computation complexity, we discretize time into z intervals, $\forall 1 \leq i \leq z, [L_i, R_i)$, with the constraints

$$i = 1, \quad L_i = 0 \quad (8)$$

$$\forall 1 \leq i \leq z, \quad R_i = L_{i+1} \quad (9)$$

$$i = z, \quad R_i = +\infty \quad (10)$$

However, in the practical system, the transmission time T_c has the upper bound R_T, so

$$i = z, \quad R_i = R_T \quad (11)$$

We can construct $f_{Tc}(t)$ as a histogram, and the probability of $P[L_i \leq T_c < R_i] = s_i / S$, where s_i is the number of samples in the interval i. So in this

case, we can use the probability mass function, $P[L_i \le T_c < R_i] \approx \int_{L_i}^{R_i} f_{T_c}(t)\,dt$ for the optimization Eq. 7, other than the continuous probability density function, $f_{T_c}(t)$.

2) If more computation resource is available, we can smooth the histogram with a *Kernel-based* density estimation, which assigns different weights to sample points to derive a continuous function. The *Kernel-based* density estimation has been implemented in R (function *density ()*) [19]. With the *Kernel-based* density estimation, we can derive the continuous probability density function.

Using either of the above two solutions, we can obtain the estimation of $f_{T_c}(t)$ based on the samples in one fragment, which is denoted as $f_{T_c}^n(t)$ and n indicates the sequence number of this fragment. In order to avoid some temporal aberration of the network condition to severely influence the density estimation of T_c, we use the exponentially weighted moving average mechanism to incorporate some historical information in the derivation of $f_{T_c}(t)$:

$$f_{T_c}(t) = \alpha * f_{T_{c(n-1)}}(t) + (1 - \alpha) * f_{T_c}^n(t), \quad (12)$$

where α is designed to control the accuracy of the estimate of the probability density function. For a stable wireless channel, we set α large to maintain the gradual update of $f_{T_{cn}}(t)$ by reducing the weight of the temporary information. For time-varying channels with frequent quality changes, we set α small since the previous network conditions have less correlation to the current or future network condition. Reference value of α can be achieved by a linear regression technique.

The information collected for $f_{T_c}(t)$ is from one specific video quality level, which the end device has received. However, in order to solve the optimization model in Eq. 1, we need the information for all categories, $[f_{T_c}(t)]_i, i = 1,2,...,m$. Therefore, we add one additional functionality to support the estimation. Every time the server finishes sending the n th fragment of video quality level j to the client, it also notifies the sizes of the $(n+1)$th fragment in all video categories $C_i^{n+1}, i = 1,2,...,m$. Since we know the probability density function $f_{T_c}(t)$, and the transmission rate $X = C_j^n / T_c$. Therefore, the probability density function of X is given by

$$f_X(x) = \left|\frac{dT_c}{dX}|_{X=x}\right| f_{T_c}(C_j^n / x) = C_j^n * x^{-2} * f_{T_c}(C_j^n / x) \quad (13)$$

With $f_X(x)$, we can derive the corresponding $f_{T_c}(t)_i$ for fragment $n+1$ as

$$f_{T_c}(t)_i = \left|\frac{dX}{dT_c}|_{T_c=t}\right| f_X(C_i^{n+1} / t) = C_i^{n+1} * t^{-2} * f_X(C_i^{n+1} / t) \quad (14)$$

In order to simplify the calculation, we maintain and update the probability density function $f_X(x)$ during the whole transmission, other than $f_{T_{cn}}(t)$, since otherwise multiple copies of $f_{T_c}(t)_i, i = 1,2,...,m$ need to be stored and updated.

3.3 Complexity analysis

In order to support real-time video transmissions, the computation complexity of solving Eq. 7 should be acceptable. Suppose we utilize the first method for $f_{T_c}(t)$, and take $P[L_i \le T_c < R_i]$ to save the computation complexity, then the major part of Eq. 7 approximately becomes,

$$\begin{aligned} & B_i + P_0 \int_{W_b T_R}^{\infty} (t - W_b T_R)[f_{T_c}(t)]_i dt \\ &= B_i + P_0 \sum_{i=j}^{z} P[L_i \le T_c < R_i] \frac{1}{2}\left[R_i^2 - L_i^2\right] \\ &\quad - P_0 \sum_{i=j}^{z} P[L_i \le T_c < R_i] W_b T_R \end{aligned} \quad (15)$$

where $L_j \ge W_b T_R$. In order to compare results of all categories, we need to calculate m times of Eq. 15. Therefore the complexity to solve the optimization problem is $O(m*z)$, which polynomial in the number of quality levels is m.

3.4 Benefit and penalty

There are two categories of metrics for evaluating the quality of video: one is objective video quality metric and the other is subjective video quality metric. We can use of either of the video quality metric in our algorithm.

We can use the popular subjective metric Mean Opinion Score (MOS) in our algorithm. The MOS is defined as a single number in the range 1 to 5, where 1 is lowest perceived video quality, and 5 is the highest perceived video quality. We can define the benefit as MOS, that is $B_i = MOS$. To be more specific, if the video has five quality level, then we can let $B_1 = 1$, $B_2 = 2$, $B_3 = 3$, $B_4 = 4$,

and $B_5 = 5$. However, in IIS Smooth Streaming, there are m levels of video quality, where m may be larger than 5. Therefore, we can use the concept of MOS and let $B_6 = 6$, $B_7 = 7$ .and so on. The MOS evaluation methodology is described in [18].

However, subjective metric has the disadvantage that it is complicated since the values are usually gained by assessment by human subjects. Therefore, objective metric such as Peak Signal-to-Noise Ratio (PSNR) [16] which is one of the most popular objective metrics for video quality evaluation can be used. PSNR can be defined through Mean-Squared Error (MSE), which is defined as

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left\|I(i,j)-K(i,j)\right\|^2, \quad (16)$$

where I and K are two frames with size of $m*n$ pixels, and one of the frames is considered a noisy approximation of the other. Therefore,

$$PSNR = 10\times\log_{10}\left(\frac{MAX_I^2}{MSE}\right), \quad (17)$$

where MAX_I is the maximum pixel value of the image. In this paper, we use PSNR as an example to evaluate our mechanism. We set B_i as the average PSNR value of every quality level of the video. Although we use PSNR as the example in this paper, we could easily use other objective metrics in our optimization algorithm.

Our optimization algorithm has the benefit that the video viewers can get subjective best experiences by defining their unit "Penalty" for STALL. For example, different user can have different subjective measure of the video stalls. If user cannot bear any video stall, they can set the stall penalty P_0 with a smaller value (keep in mind that P_0 is negative). Then the optimization will try to calculate the optimal quality level without stalls. On the other hand, if the user prefers high-standard video more than unchopped video when network bandwidth is limited, the user can set the stall penalty with a larger value. In this case, our optimization algorithm will obtain the video with highest possible viewing experience. We can see that client developer can set the default values for stall penalty based on some experiments of most users. The client can change the value if they are not satisfied with the setting or depending on the video they would like to watch.

4 Simulations

4.1 Simulation setup

We evaluated the performance of our ISAVS optimization algorithm and compared it with IIS Smooth Streaming heuristic bitrate switching strategy. We use the Evalvid [14] architecture and QualNet network simulator [15] for performance evaluation. The most popular raw video sequence called "Foreman" [13] is used as the test video stream. "Foreman" is a 16 second video containing 400 frames. The YUV format raw video is coded and compressed by MP4 with GOP, pattern of IBBPBBPBBPBB. In our performance validation, we encoded the video with five different quality levels, starting from 1to 5, with different average video bitrates, specifically 240 kbps, 340 kbps, 440 kbps, 540 kbps, and 640 kbps, respectively. Fig. 3 shows the five different video bitrates of "foreman". As seen from the figure, the video bitrate is time-varying with some patterns; a detailed analysis can be found in [17] .We use the most popular metric PSNR as the "benefit" in our optimization algorithm shown in Table II. The average PSNR can be obtained from the definition or the model shown in [17] for real-time video application.

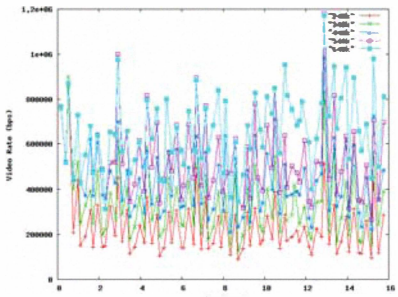

Figure 3 Different video bitrates and the corresponding quality (measure in PSNR) for the video "Foreman"

Table 2 Benefits in terms psnr of different video quality levels

Levels	Average Bitrates	Average PSNR (Benefit)
1	240 kbps	35.12 dB
2	340 kbps	36.75 dB
3	440 kbps	38.04 dB
4	540 kbps	39.13 dB
5	640 kbps	40.32 dB

4.2 Results and analysis

We set up an IEEE 802.11 wireless network using QualNet 4.5 for performance evaluation. Besides the target video streaming application, we set up some background CBR traffic. The end-to-end available bandwidth of the video streaming between the client and the server is shown in Fig 4. As seen from Fig 4, the available bandwidth vary over time since there are ten random CBR applications as background traffic in the WLAN, which compete for the network bandwidth with the video streaming application. Simulation results are

shown in Fig.6 for this case referred to as Case One.

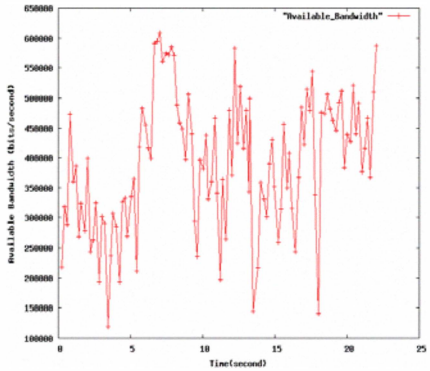

Figure 4. A trace of the available network bandwidth for HTTP-based streaming (Case one)

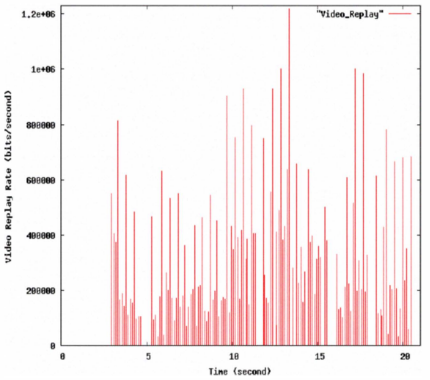

Figure 5. Video replay bitrate for the IIS Smooth Streaming strategy for Case one

The video replay bitrate of "Foreman" decided by IIS Smooth Streaming strategy is shown in Fig 5. As seen from Fig. 5, there are several STALLs when the video is replayed, which are at the beginning, and around 5s, 12s, 16s and 18s. The STALL at the beginning is unavoidable as it is the time for the first video fragment to be sent over the network. IIS Smooth Streaming uses the lowest quality level for the first fragment by default to avoid waiting too long at the start. However, the STALLs in the middle during video replay cause bad video viewing experiences and they can be reduced.

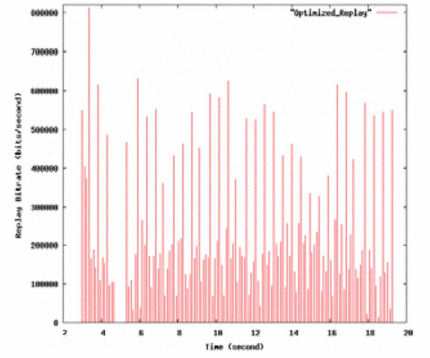

Figure 6. Video replay bitrate for ISAVS for Case one.

The video replay rate using our optimization algorithm under the same network condition is shown in Fig 6.We can see that our optimization algorithm can guarantee the best possible viewing experience according to available network bandwidth. As seen from Fig 6, our proposed algorithm has only 2 STALLs. One STALL is in the beginning and the other is around5second. These two stalls are unavoidable. The first STALL is caused by transmission time for the first fragment and is the same as in the case of IIS Smooth Streaming. The second STALL is because the available network bandwidth is not sufficient for video transmission. The available bandwidth cannot support even the lowest quality level of the video.

The detail comparison of IIS Smooth Streaming and ISAVS is shown in Table III. We set the unit penalty as −40dB/second. IIS Smooth Streaming stalls 4 times during video replay. The total stall time of IIS Smooth Streaming is 1.77s excluding the first unavoidable stall, which counts for 11 percents for a 16 second video. While ISAVS stalls only once due to the insufficient bandwidth with total stall time 0.575s, accounting for 4 percents of the whole video. Our optimization algorithm decreases the "STALL Percentage" by 63%, which is a significant improvement. As seen from the row "Total Benefit plus Penalty", which is calculated as $\sum_{i=1}^{m} B_i + P_0 * Total\ Stall\ Time$, our optimization algorithm ISAVS achieves higher PSNR and provide the best possible viewing experience for users based on the effective network bandwidth while IIS Smooth Streaming strategy cannot optimize the video and network resource.

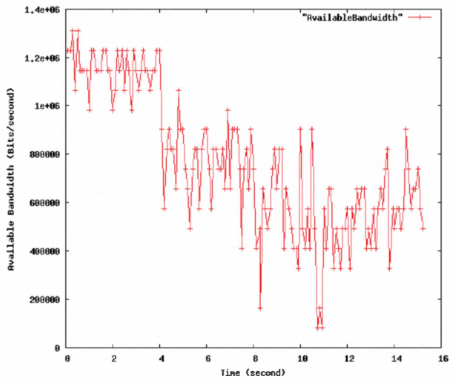

Figure 7. Available network bandwidth for HTTP-based streaming (Case two).

We consider simulation with the same setting except that the WLAN has more available bandwidth shown in Fig 7. We find out that both ISAVS and IIS Smooth Streaming do not have any stalls since the available bandwidth is sufficient to support the video transmission. Both methods choose almost the same quality level for all the

fragments except the last fragment: Smooth Streaming chooses the last fragment of the video with quality levels 3, while our optimization algorithm chooses quality levels 5. Smooth Streaming chooses level 3 because the available bandwidth in the previous measurement interval was not good. On the contrary, our optimization algorithm chooses better video quality for users according to available network bandwidth and the amount of video streams buffered.

Table 3 Case one: comparison of simulation results

Metrics	IIS	ISAVS
Total Stall Time[2]	1.77s	0.575s
Number of Stalls[2]	4	1
Percentage of Time Stall[12]	11%	4%
Total Benefit plus Penalty(PSNR)	109.9 dB	145.93 dB

5 Conclusions

In this paper, we proposed an optimization solution to solve the key challenge in HTTP-based adaptive video streaming. Our optimization model can be easily adopted in IIS Smooth Streaming, Flash Dynamic Streaming, or HTTP Adaptive Bitrate Streaming with either objective or subjective video quality metrics. From our detail simulations and analysis, we show the advantage of our optimization model to fully utilize network resource and buffered video contents and provide high video quality with minimal replay interruptions.

Acknowledgements

This research was funded in part by NSF grant CSR-0917315, China National High Technology Plan Grant 2007AA01Z225 and NSF China 60972044.

References

[1] Alex Zambelli, IIS Smooth Streaming Technical Overview, March 2009.
[2] http://www.youtube.com
[3] http://www.tudou.com
[4] http://www.hulu.com
[5] http://www.netflix.com/
[6] David Hassoun, Dynamic Streaming in Flash Media Server 3.5 - Part 1: Overview of the new capabilities, January, 2009.
[7] Apple IETF draft proposal, HTTP Live Streaming Draft-pantos-http-livestreaming-01, http://tools.ietf.org.
[8] Microsoft Std., IIS Smooth Streaming Transport Protocol, Sept., 2009.
[9] Schulzrinne, H., et al, Real Time Streaming Protocol (RTSP), RCF 2326.
[10] ISO Standard, Information Technology – Coding of Audio-visual Objects– Part 12: ISO Base Media File Format, http://www.iso.org.
[11] Jupiter Research white paper, The Importance of Delivering a Great Online Video Experience, July 11, 2007
[12] Erick Schonfeld, ComScore: YouTube Now 25 Percent of All Google Searches, http://techcrunch.com.
[13] http://www.tkn.tu−berlin.de/research/evalvid/qcif.html.
[14] A. Lie, and J. Klaue, Evalvid-RA: Trace Driven Simulation of Rate Adaptive MPEG-4 VBR Video, Multimedia Systems, Nov. 2007.
[15] www.scalable-networks.com.
[16] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, Analysis of video transmission over lossy channels, In IEEE Journal on Selected Areas in Communications, June 2000.
[17] X. Qiu, H. Liu, et al. Adaptive Video Compression Rate Optimization in Wireless Access Networks, In Proc. of IEEE LCN, Switzerland, 2009.
[18] ITU-R BT.500-12 Document Information, Methodology for the Subjective Assessment of the Quality of Television Pictures, Sep., 2009.
[19] R Development Core Team, The R Manuals, http://www.r-project.org
[20] Taehyun Kim, Mostafa Ammar, Receiver Buffer Requirements for Video Streaming over TCP, in proceedings of Visual Communications and Image Processing Conference, 2006

[1] Excludes the starting stall.