AutoTune: Game-based Adaptive Bitrate Streaming in P2P-Assisted Cloud-Based VoD Systems

Yuhua Lin and Haiying Shen
Department of Electrical and Computer Engineering
Clemson University, Clemson, South Carolina 29634
{*yuhual, shenh*}*@clemson.edu*

***Abstract*—Hybrid peer-to-peer assisted cloud-based video-on-demand (VoD) systems augment cloud-based VoD systems with P2P networks to improve scalability and save bandwidth costs in the cloud. In these systems, the VoD service provider (e.g., NetFlix) relies on the cloud to deliver videos to users and pays for the cloud bandwidth consumption. The users can download videos from both the cloud and peers in the P2P network. It is important for VoD service provider to i) minimize the cloud bandwidth consumption, and ii) guarantee users' satisfaction (i.e., quality-of-experience). Though previous adaptive bitrate streaming (ABR) methods improve video playback smoothness, they cannot achieve these two goals simultaneously. To tackle this challenge, we propose AutoTune, a game-based adaptive bitrate streaming method. In AutoTune, we formulate the bitrate adaptation problem in ABR as a noncooperative Stackelberg game, where VoD service provider and the users are players. The VoD service provider acts as a leader and it decides the VoD service price for users with the objective of minimizing cloud bandwidth consumption while ensuring users' participation. In response to the VoD service price, the users select video bitrates that lead to maximum utility (defined as a function of its satisfaction minus associated VoD service fee). Finally, the Stackelberg equilibrium is reached in which the cloud bandwidth consumption is minimized while users are satisfied with selected video bitrates. Experimental results from the PeerSim simulator and the PlanetLab real-world testbed show that compared to existing methods, AutoTune can provide high user satisfaction and save cloud bandwidth consumption.**

***Keywords*-Video streaming; Cloud computing; Bitrate adaptation**

I. Introduction

The cloud has proved to be an effective infrastructure to host stable and robust video streaming services [1]–[3]. More and more video-on-demand (VoD) service providers (e.g., Netflix) are deploying their web applications on the cloud. Meanwhile, peer-to-peer (P2P) technology has been applied to various multimedia streaming applications for P2P video sharing in order to reduce the bandwidth cost of the server. As a result, hybrid P2P-assisted cloud-based video-on-demand (hybrid VoD in short) systems that combine P2P and cloud computing have been proposed [4] for multimedia streaming services, which take advantage of both systems. LiveSky [4] is a popular live streaming system that adopts the hybrid VoD approach to serve ten million users. The potential benefits of such a hybrid approach are analyzed and validated by previous studies [5]–[8]. Figure 1 demonstrates an overview of the hybrid VoD architecture. It has three components: servers owned by VoD service providers, cloud (including cloud storage and its content delivery network) and VoD users (in this paper, users, video players, clients, peers and nodes are interchangeably used based on the context). The VoD service provider stores all original video files on its servers, and uploads new videos to the cloud when they are released to users. The cloud is responsible for storing video files and streaming the videos to end users across the wide areas, using edges servers that are distributed globally. End users can download videos from both the cloud and peers in the P2P network. The cloud acts as main contributor that streams videos to users. When a user is far from the nearest cloud edge server or the network connection between the user and the cloud is not in a good condition, peers are needed to assist the streaming of videos. The users need to pay the VoD service provider for using the video streaming service (i.e., watching videos). The VoD service provider needs to pay for users' bandwidth consumption (i.e., its bandwidth usage) on the cloud during a period of time [9], [10]. To maximize the VoD service provider's profit and attract a large number of users, it is important for the VoD service provider to 1) minimize cloud bandwidth consumption; and 2) guarantee users' satisfaction (i.e., quality-of-experience) in video watching.

Various adaptive bitrate streaming (ABR) methods have been proposed for video streaming systems to improve video playback smoothness. One approach is to adjust user's video bitrate by examining bandwidth or buffer conditions of the server [2], [11]–[13]. When the server has low bandwidth and its sending buffer has a small number of video chunks, it reduces video bitrate in order to deliver video chunks to the users on time. Otherwise, it increases video bitrate in order to provide high quality videos to users. However, this

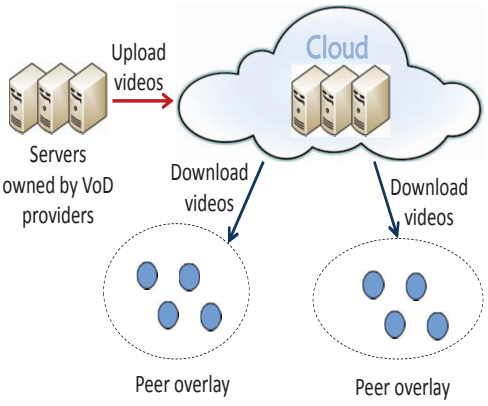


Figure 1: Overview of a P2P-assisted cloud-based VoD architecture.

server-side adaptation approach fails to guarantee user satisfaction, as it adapts a user's video bitrate based on the server's bandwidth capacity. A user may get low video bitrate even if there are a large number of video chunks stored in its buffer and it has high bandwidth capacity. Another approach is to adjust video bitrate by estimating a user's bandwidth capacity based on the current level of its playback buffer [14]–[16]. It reduces the bitrate to be more conservative when user's buffer is at risk of underrunning, and increases bitrate to be more aggressive when user's buffer has stored a large number of video chunks. As each user aims to maximize its own video bitrate based on its buffer condition, it leads to a large size of video downloads from the cloud. Therefore, this client-side adaptation approach fails to minimize cloud bandwidth consumption if it is applied to hybrid VoD systems.

To overcome the drawbacks of existing ABR methods in achieving the aforementioned goals in hybrid VoD systems, we propose AutoTune, a game-based adaptive bitrate streaming method. In AutoTune, we formulate the bitrate adaptation problem in ABR as a noncooperative Stackelberg game, where the VoD service provider and users are players. The Stackelberg equilibrium is reached in which cloud bandwidth consumption is minimized while users are satisfied with the selected video bitrates.

The remainder of the paper is organized as follows. Section II gives an overview of related work. Section III introduces the goal of this work and provides an overview of AutoTune. Section IV introduces the detailed design of AutoTune. Section V presents the performance evaluation on both PlanetLab and PeerSim. Section VI concludes the paper with remarks on our future work.

II. Related Work

LiveSky [4] is the first real-world implementation of P2P-assisted cloud-based VoD systems. Early works analyzed the effectiveness of the hybrid VoD systems in reducing server bandwidth costs and increasing system scalability through simulations [6]–[8]. To improve the quality-of-experience for users under unstable network condition in video streaming systems, many methods have been proposed to adaptively tune video bitrates. Existing ABR methods can be classified to two groups: server-side adaptation and client-side adaptation.

In the server-side adaptation approach, video bitrate is determined by available bandwidth or the buffer condition of the server. Mansy *et al.* [11] modeled the adaptive streaming as a linear optimization problem and proposed a problem solution that allocates a bitrate to a client based on client's requested bitrate and upload bandwidth capacity of the servers, with the goal of maximizing the fraction of clients that receive their requested bitrates. Dynamic adaptive streaming over HTTP (DASH) [2], [12], [13] picks a high bitrate for users to improve video quality when TCP throughput is high, and switches to a low bitrate to avoid playback interruptions in order to provide smooth video streaming service to users. However, these methods overlook real time buffer conditions on the client side, and fail to maximize the users' satisfaction of the video quality.

In the client-side adaptation approach, video bitrate is adjusted by available bandwidth or the number of downloaded video chunks in the buffer of users. Adobe and Apple both have developed web-based adaptive video streaming service as a function of flash players [14], [15]. In these designs, the video client monitors its bandwidth and CPU conditions and adaptively switches video quality during playback. Joint-Family [16] manages multiple swarms for peers watching a video of different bitrates. When a peer's buffer has more than (or less than) a certain number of video chunks to play and the last bitrate change is more than several seconds ago, the peer increases (or decreases) the video bitrate. However, this client-side adaptation approach aims to maximize each user's video bitrate, so it leads to increased video size downloaded from the cloud if it is applied to hybrid VoD systems), thus cannot minimize cloud bandwidth consumption.

Stackelberg games [17] have been applied to video streaming system to solve the resource allocation problem [18]–[20]. When mobile users and desktop users both receive video streaming services from the cloud, they compete with each other for cloud bandwidth allocation. Nan *et al.* [18] formulated the bandwidth allocation problem as a Stackelberg game, they then proved the existence of a unique Nash Equilibrium in this game where both mobile users and desktop users

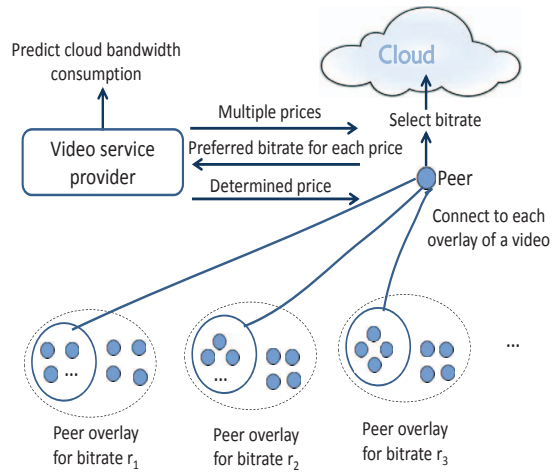


Figure 2: Overview of the game-based bitrate adaptation process.

meet their QoS requirements. Wu *et al.* [19] developed a Stackelberg game between a VoD service provider and the peers, where the VoD service provider uses a rewarding strategy to incentivize peers to contribute upload bandwidth resource. In order to maximize upload bandwidth from peers in VoD systems, Mostafavi *et al.* [20] formulated a Stackelberg game in which the VoD server is the leader and peers are followers. The leader decides the amount of payment for each peer, the peers then calculate how much upload bandwidth to contribute accordingly.

AutoTune is advantageous to existing ABR methods in that it can simultaneously minimize cloud bandwidth consumption and guarantee users' satisfaction (i.e., quality-of-experience) in video watching in hybrid VoD systems. To the best of our knowledge, this is the first work to apply a Stackelberg game to solve the ABR problem in hybrid VoD systems.

III. Background and Problem Statement

In a VoD system that provides videos with various bitrates, each video has a number of distinct files for different bitrates (i.e., one file at each bitrate). The hybrid VoD system maintains a P2P overlay for each bitrate of a video, where peers exchange video chunks with each other. A peer watching a video joins different overlays for different bitrates of the video concurrently and maintains active connections with its neighbors [16], [21]. As in current P2P applications, users in this hybrid VoD system are incentivized to contribute their upload bandwidth in order to download video chunks from the peers [8]. When a peer switches to a new bitrate, it sends out requests to the corresponding overlay as it downloads and uploads chunks. In the P2P-assisted cloud-based VoD system, users download video chunks from two sources, the cloud and peers. The VoD service provider needs to pay the cloud provider for bandwidth consumption for users' video downloading from the cloud (i.e., bandwidth usage of the VoD service provider), but does not need to pay for users' video downloading from peers. The cloud provider uses a usage-based pay-as-you-go charging model for the bandwidth usage of the VoD service provider [9], [10], in which monetary cost is calculated by the total amount of bytes transferred from the cloud to users during a period of time. To maximize its profit in a competitive market, a VoD service provider aims to reduce cloud bandwidth consumption while guaranteeing users' satisfaction in video watching.

Using an ABR method is one way to achieve this goal, but it faces two challenges. On one hand, each user desires to enjoy a good quality-of-experience and maximize the bitrate of the video that it is watching. Due to limited bandwidth capacities of peers, it is hard to find peers that can supply very high bitrates. Then, the user must download a large-size video file from the cloud and hence increases cloud bandwidth consumption. On the other hand, the VoD service provider could constrain cloud bandwidth consumption of users, which however may degrade users' satisfaction (i.e., quality-of-experience) in video watching.

To tackle the aforementioned challenge, in this paper, we propose a game-based video bitrate adaptation method called AutoTune. It leverages an observed fact that users are satisfied when video bitrate reaches a certain level and an additional bitrate increase will not further greatly increase users' satisfaction. Specifically, a user can enjoy good quality-of-experience at bitrate 720kbps, and it will not greatly increase the user's satisfaction by switching the bitrate to 1080kbps [22]. As the VoD service provider is only charged by the total amount of cloud bandwidth used during a period, to reduce cloud bandwidth payment cost, it can encourage users to download video chunks from peers by setting price for users' cloud bandwidth usage. Therefore, users can be motivated to choose bitrate that they are satisfied with rather than choosing an excessive high bitrate and motivated to use peer bandwidth that can support the bitrate that they are satisfied with.

In AutoTune, the VoD service provider periodically estimates the expected cloud bandwidth demand in the next time period (denoted by T_p). It then accordingly sets multiple unit prices for cloud bandwidth consumption when users download video chunks from the cloud with the objective of encouraging users to select a video bitrate that has high bandwidth supply from peers. When a user needs to adjust its bitrate based on its buffer condition, it identifies multiple possible bitrates (Section IV-A). A Stackelberg game between users and

VoD service provider is formulated as the following process: 1) based on each of the multiple unit prices, the user chooses a new bitrate that maximizes its utility, which is the user's satisfaction with the video quality minus the VoD service cost; 2) the VoD service provider chooses one unit price among multiple unit prices that maximizes its own revenue. Note that the provider will not simply choose the highest unit price because it will discourage users from watching a video at higher bitrates. Finally, each user chooses a new bitrate based on this determined unit price (Section IV-B). We will elaborate each step in Section IV.

IV. System Design

An overview of our game-based bitrate adaptation process is shown in Figure 2. A client first determines multiple possible new bitrates to watch based on its buffer condition (Section IV-A). Based on multiple prices for each bitrate from the VoD service provider, the client calculates the bitrate that maximizes its utility based on each price. The VoD service provider calculates optimal VoD service price that maximizes its revenue based on the responses from clients. The client then calculates the optimal video bitrate that maximizes its utility (Section IV-B). Important notations used in this paper are listed in Table I, in which we use "predefined" to indicate that the parameter is predefined by the VoD service provider.

Table I: Table of important notations.

r_i	video bitrate
R	the set of bitrates for each video
t_g	time gap since the last bitrate change
PR_k	the set of bitrates that user k can switch to
N_u^b	upper bound of # of chunks in buffer (predefined)
N_l^b	lower bound of # of chunks in buffer (predefined)
N^b	# of chunks stored in the buffer
$F(k)$	utility function of user k
U_s	user's satisfaction degree in watching a video
U_p	payment cost on cloud bandwidth consumption
s_i	satisfaction parameter for bitrate r_i (predefined)
α_i	scale factor corresponding to bitrate r_i (predefined)
p	unit price of the VoD service (predefined)
r_u	joint bandwidth contribution from peers
w_k	weight of payment cost set by user k
$L(p)$	utility function of the VoD service provider
$\bar{c}$	base cloud bandwidth usage (predefined)
$\bar{p}$	base unit price for the cloud bandwidth (predefined)
m	# of levels of VoD service prices (predefined)

A. Client Buffer Based Bitrate Adaptation

We use a set $R =< r_1, r_2..., r_B >\ (r_1 < r_2 < ... < r_B)$ to denote different bitrates for each video provided by the VoD service provider. We use a general set of bitrates for all videos in this paper for convenience of analysis, and our design can be easily extended to the scenario where different videos have various bitrates.

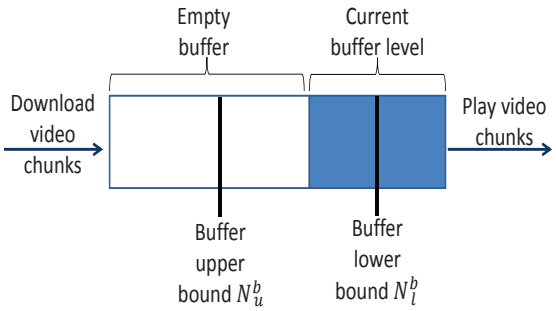


Figure 3: Demonstration of user buffer utilization.

A video encoded with a higher bitrate is larger in size and has a higher quality. The chunk size is finite (i.e., a number of seconds long regardless of video bitrate) and a chunk is stored in the player's buffer after it is downloaded. To avoid playback interruption, it is required to have at least one chunk available in the player's buffer. When downloading chunks, we use the Earliest-First chunk selection strategy, which is to download the chunks in order [23]. This strategy allows for a fast startup, potentially fewer and shorter playback interruptions, and smaller wastage of downloaded chunks when a user abandons viewing a video. The rule for adaptive bitrate streaming is that when a client's buffer has few chunks, the client should be more "conservative" and deliberately underestimate its bandwidth capacity so as to pick a lower video bitrate to quickly replenish its buffer, and vice versa.

Figure 3 shows buffer status when a user is playing a video. A video player downloads video chunks from the cloud and peers and stores them in its buffer (as shown on the left side). At the same time, the video player fetches chunks from the buffer and plays them (as shown on the right side). In order to guarantee smooth playback of a video, the video player needs to store a certain amount of video chunks (called reservoir) to tolerate sudden changes in network condition. As most browser-based video players do not have control over the underlying TCP connections, they cannot cancel the downloading process of an ongoing video chunk before the completion of downloading. While the video player is in the middle of downloading a chunk, if the download speed suddenly slows down due to inferior network connection, the reservoir might run empty before the video player can switch to a lower bitrate. In this case, video playback is interrupted. Thus, we need to maintain a buffer level (denoted by N^b seconds of chunks) that is greater than the reservoir, so that there are enough video chunks in the buffer to tolerate download bandwidth variation.

Assume that a video player is currently playing a video at bitrate r_i, it increases bitrate if the follow-

ing two conditions hold: 1) its buffer has more than N_u^b sequential chunks to playback, and 2) the last bitrate change was made more than t_l seconds ago. The rationale behind the first condition is that we need to have a high buffer level to prevent playback interruption in the process of bitrate adaptation. Also, as the action of bitrate adaptation will not affect the buffer status immediately, if multiple consecutive bitrate adaptations occur within a short time period, it may lead to excessively high or excessively low bitrates. The second condition prevents such an excessive reaction by preventing duplicated bitrate adaptations within a short time. We use t_g to denote the time gap since the last bitrate change, and use PR_k to denote the new set of possible bitrates that video player k can choose for bitrate increase or decrease.

$$PR_k = \{r_j : r_j \geq r_i \wedge \frac{r_j - r_i}{r_i} < \frac{N^b - N_u^b}{N_u^b}\}, \quad if \begin{cases} N^b > N_u^b \\ t_g > t_l \end{cases} \tag{1}$$

An increased bitrate means a larger file and longer download time. $\frac{r_j - r_i}{r_i} < \frac{N^b - N_u^b}{N_u^b}$ guarantees that the size of excessive buffered chunks can tolerate bitrate increase; that is, it is long enough to play for the time period when the chunks with increased bitrate is being downloaded. Then, when video bitrate adjusts up, the user will not suffer from playback interruption. Contrarily, video player k decreases bitrate if its buffer has less than N_l^b seconds of chunks, and the last downward bitrate change is more than t_l seconds ago.

$$PR_k = \{r_j : r_j \leq r_1 \wedge \frac{r_i - r_j}{r_i} < \frac{N_l^b - N^b}{N_l^b}\}, \quad if \begin{cases} N^b < N_l^b \\ t_g > t_l \end{cases} \tag{2}$$

Algorithm 1 The client buffer based bitrate adaptation method.

Input: user k's buffer level N^b; current bitrate r_i; time gap since last bitrate change t_g;
Output: new set of video bitrate PR_k;
if $t_g > t_l$ **then** //time gap since last bitrate change is larger than t_l
  if $N^b > N_u^b$ **then** //buffer level exceeds upper bound
    $PR_k = \{r_j : r_j \geq r_i \wedge \frac{r_j - r_i}{r_i} < \frac{N^b - N_u^b}{N_u^b}\}$
  end if
  if $N^b < N_l^b$ **then** //buffer level below lower bound
    $PR_k = \{r_j : r_j \leq r_1 \wedge \frac{r_i - r_j}{r_i} < \frac{N_l^b - N^b}{N_l^b}\}$
  end if
end if
return PR_k

Algorithm 1 shows the pseudocode of the client buffer based bitrate adaptation method. After video player k decides to switch its video bitrate based on its buffer condition and calculate the new set of bitrates PR_k, it picks a suitable bitrate from PR_k by using the price driven bitrate adaptation method introduced in Section IV-B. The newly selected bitrate should achieve a tradeoff between reducing cloud bandwidth consumption and guaranteeing user's quality-of-experience.

B. Price Driven Bitrate Adaptation

In hybrid VoD systems, content and bandwidth sharing among peers is desirable for the satisfaction of peers, in which peers exchange their downloaded videos with each other. To ensure a long-term streaming quality, peers contribute their bandwidths in a cooperative manner and form overlays. One peer joins an overlay only if it can receive high-bitrate video with smooth continuity, and otherwise it will keep out of the overlay. When upload bandwidth from a user's connected peers is not sufficient the support smooth playback of a video, the user needs to download video chunks from the cloud.

To reduce cloud bandwidth consumption, the VoD service provider can adjust the VoD service price to encourage users to download videos from peers and discourage them from choosing an excessively high bitrate. Thus, we can formulate the ABR problem as a noncooperative Stackelberg game [24] between the VoD service provider and users, where the VoD service provider is the leader and users are followers. In the Stackelberg game, the leader considers the response of the follower, and then takes an action that maximizes its utility. The follower observes the action made by the leader and picks an action in response to the leader's action to maximize its own utility.

In our established Stackelberg game, on one hand, the VoD service provider needs to set VoD service price for users with the objective of minimizing cloud bandwidth consumption while ensuring users' participation. The VoD service price should be reasonably high enough so as to encourage users to download videos from the peers and to choose a reasonably high bitrate to save cloud bandwidth consumption. Also, the price should be acceptable for users to continue using the VoD service. On the other hand, based on the VoD service price, users select the bitrate of video that can achieve a tradeoff between their satisfaction and the associated VoD service cost. Finally, we solve the Stackelberg equilibrium of the game, i.e., the game reaches a state that cloud bandwidth consumption is minimized while users are satisfied with selected video bitrates. Below, we first introduce the utility of a user, then introduce the utility of the VoD service provider, and finally present the solution.

1) Utility Function of a User: For each user k, we define a utility function to quantify the level of benefit that user k obtains from watching a video. It equals to the user's satisfaction degree minus the payment cost from watching the video:

$$F(k)(r_i, \alpha_i, s_i, p, r_u) = U_s(r_i, \alpha_i, s_i) - w_k U_p(r_i, r_u), \quad (3)$$

where $U_s(\cdot)$ is a function to represent a user's satisfaction degree in watching a video of a specific bitrate, and $U_p(\cdot)$ is the payment cost function on cloud bandwidth consumption. In Equation (3), r_i is the requested video bitrate from user k; s_i is the satisfaction parameter associated with a specific video bitrate r_i, which is a measurement of the satisfaction level a user obtains when watching a video at bitrate r_i; α_i is a scale factor corresponding to r_i; p is the unit price of the VoD service; and r_u is the joint bandwidth contribution from peers. For example, if a user is connected to three peers and each of them has download bandwidth of 100kbps, then r_u=300kbps. $w_k \in [0, 1]$ is the weight of payment cost set by user k. w_i=0 means that a user only aims to maximize its video quality and does not consider cloud VoD service price; while w_i=1 means that a user aims to save the bandwidth payment cost by sacrificing satisfaction.

Function $U_s(\cdot)$ is considered to be non-decreasing as each user desires high-quality videos and a video at a higher bitrate makes a user more satisfied. At the same time, the marginal satisfaction of a user is non-increasing because a user's level of satisfaction gradually gets saturated when the video bitrate increases [22]. For example, when the video bitrate increases from 200kbps to 500kbps, a user may get greatly satisfied because the video quality goes from blurred to clear. Since the user's satisfaction is almost saturated, (s)he will not get much satisfaction when the video bitrate increases from 500kbps to 800kbps. Considering these properties, we design $U_s(\cdot)$ as a concave function. Since the natural logarithmic functions are representative concave functions [17] that are commonly used to evaluate user satisfaction [18], [24], we define:

$$U_s(r_i, \alpha_i, s_i) = \alpha_i \ln(1 + s_i r_i). \quad (4)$$

s_i is a satisfaction parameter associated with r_i, in order to emphasize the influence of r_i in calculating U_s, we consider the product of s_i and r_i. A user's payment cost for watching a video equals the product of unit VoD service price (p) and user's cloud bandwidth consumption for watching the video, which is its total bandwidth consumption minus the download bandwidth consumption from peers (i.e., the upload bandwidth from connected peers). Then,

$$U_p = p(r_i - r_u). \quad (5)$$

The price per unit of cloud bandwidth set by the VoD service provider affects the utilities of the users; the utility of a user decreases with a higher price and vice versa. Combining Equation (4) and Equation (5) into Equation (3), we get:

$$F(k)(r_i, \alpha_i, s_i, p) = \alpha_i \ln(1+s_i r_i) - w_k p(r_i - r_u). \quad (6)$$

2) Utility Function of the VoD Service Provider: When a user downloads a video from the cloud, it is charged by the VoD service provider only based on download bitrate from the cloud. The objective of a VoD service providers is to maximize its revenue, which is calculated by:

$$L(p) = p \sum_n (r_i - r_u), \quad (7)$$

where n is the number of users watching videos during a unit time period. Given the cloud bandwidth demand from each user, the VoD service provider needs to set a price per unit of cloud bandwidth (p) so that its revenue is maximized.

3) Optimal Bitrate Selection: As discussed above, the video bitrate adaptation problem can be modeled as a Stackelberg leader-follower game. In this game, the VoD service provider determines a set of prices based on predicted cloud bandwidth usage in the next time period. For each price, each user k that needs to select the bitrate from its PR_k that maximizes its utility. Note that the users' utility function is a concave function that causes users to demand less at higher bitrates in order to reduce the payment cost. Based on the selected bitrates reported by users, the VoD service provider determines the price that maximizes its revenue. Finally, based on the determined price, each user k selects its bitrate. Below, we introduce the details of each step.

After each time period T_p, the VoD service provider estimates its cloud bandwidth usage for the next time period and then sets a set of prices for VoD service accordingly. The cloud bandwidth consumption for the next time period is predictable according to previous study [25]. Assuming that current time is t_i, cloud bandwidth consumption during time interval $[t_i, t_i+T_p]$ (denoted by $\tilde{c}_{t_i}$) can be estimated by using an Exponentially Weighted Moving Average (EWMA) model [26]. That is:

$$\tilde{c}_{t_i} = \beta \times c_{t_{i-1}} + (1 - \beta) \times \tilde{c}_{t_{i-1}}, \quad (8)$$

where $\tilde{c}_{t_{i-1}}$ denotes estimated cloud bandwidth consumption and $c_{t_{i-1}}$ denotes actual cloud bandwidth consumption in time interval $[t_i - T_p, t_i)$, and β $(0 < \beta < 1)$ is a constant used to control the degree of

weighting decrease.

The VoD service provider then determine multiple prices for estimated bandwidth usage in the next time period. We use $\bar{c}$ to denote base cloud bandwidth usage and $\bar{p}$ to denote base unit price for the base cloud bandwidth, i.e., the smallest unit price, set by the VoD service provider. Assume the VoD service provider sets m levels of VoD service prices for each level of cloud bandwidth usage. We define the m VoD service prices for estimated cloud bandwidth usage as:

$$p_j = \log(j) \cdot \bar{p} \cdot \lfloor \tilde{c}_i / \bar{c} \rfloor, \ j \in [1, m]. \tag{9}$$

We use $V =< p_1, p_2, ..., p_m >$ to denote m levels of unit prices for cloud bandwidth usage in the next time period. The VoD service provider notifies users of $V =< p_1, p_2, ..., p_m >$. The unit VoD service price grows with the total cloud bandwidth consumption of the VoD system. That is, when expected cloud bandwidth consumption is high, the VoD service provider sets a relatively higher unit price for users. In this case, users can either select a video bitrate that leads to less cloud bandwidth consumption or pay more money for the cloud bandwidth they used.

As we explained in Section IV-A, a video player k needs to choose a bitrate in PR_k when it wants to increase or decrease its bitrate based on its buffer condition. For each $p_j \in P$, the video player who acts as a follower chooses a new bitrate r_i that maximizes its utility $F(k)$, denoted by r_{ij}. That is, the selected r_i satisfies:

$$r_{ij} = \underset{r_i \in RP_k}{\operatorname{argmax}} \ F(k)(r_i, \alpha_i, s_i, p_j) \tag{10}$$

Finally, video player k creates a preferred bitrate vector $R_k =< r_{i1}, r_{i2}, ..., r_{im} >$ and sends R_k to the VoD service provider.

The VoD service provider acts as the leader and aims to set a price (denoted by p_l) that maximizes its utility $L(p)$ when it receives the preferred bitrate vectors from all clients. That is,

$$p_l = \underset{p_j \in P}{\operatorname{argmax}} \ L(p_j) = \underset{p_j \in P}{\operatorname{argmax}} \ p_j \sum_n r_{ij}. \tag{11}$$

The VoD service provider then notifies all clients of the newly set VoD service price p_l. Then, each client k picks its preferred bitrate corresponding to p_l in its R_k, i.e., r_{il}.

C. Chunk Request Management

As in typical P2P systems, each peer is allowed to connect to a limited number of neighbors, e.g., the number of neighbors is limited to 80 in BitTorrent. In AutoTune, a node maintains active connection to a maximum of N_p peers, a node shares and receives video chunks to and from other peers. We assume that the cloud has full control over the P2P network. Therefore, each user contacts the cloud when it first enters the system. The cloud will assign a number of connections to the new user and help it join the overlay of the video it is watching. Each user caches video chunks of multiple video bitrates it has downloaded in its local machine, so it can share these video chunks with its neighbors in multiple overlays. In our design, we assume that users are incentivized to be honest and selfless by underlying incentive mechanism in the P2P network, i.e., they will report the optimal bitrates truthfully and contribute bandwidth to the P2P network.

When a user is watching a video, it requests video chunks according to their time sequence, i.e., a user always uses the earliest-first chunk selection strategy and downloads the chunks in order. The user first broadcasts a chunk request to all of its neighbors, and if it cannot receive the chunk needed from its neighbors, it will download the missing chunk from the cloud.

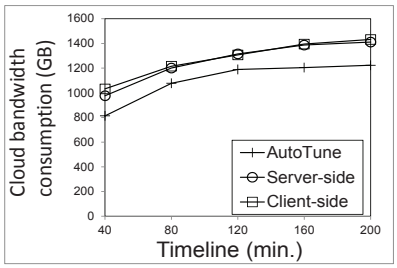
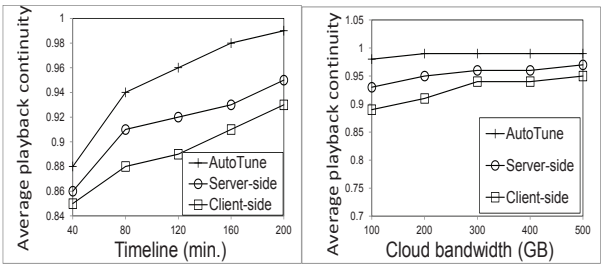


Figure 4: Cloud bandwidth consumption on PeerSim.

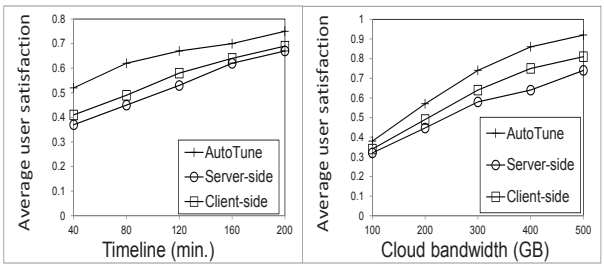
V. Performance Evaluation

We conducted experiments on the PeerSim [27] simulator and the PlanetLab [28] real-world testbed to evaluate the performance of *AutoTune* in comparison with other systems. The PeerSim simulation can test a large-scale network while PlanetLab can provide a real-world testing environment (e.g., real packet transmission delay). We measured the performance in cloud bandwidth consumption, user satisfaction and playback continuity. We compared *AutoTune* with the server-side adaptation method [11] (denoted by *Server-side*) and client-side adaptation [16] (denoted by *Client-side*) in the hybrid VoD system. These two methods does not consider the pricing policies. In both PeerSim simulation and PlanetLab experiment, we have implemented the client buffer based bitrate adaptation method to determine if a user needs to tune its video bitrate. We then implemented the price driven bitrate adaptation strategy to determine which bitrate a user would choose that maximizes its utility. In this strategy, each user calculates the optimal bitrates corresponding to different cloud bandwidth unit prices. The VoD service provider,



(a) Results at different time intervals.(b) Results at different cloud bandwidth consumption.

Figure 5: Average playback continuity on PeerSim.



(a) Results at different time intervals.(b) Results at different cloud bandwidth consumption.

Figure 6: Average user satisfaction on PeerSim.

acts as the leader, determines the optimal unit price that maximizes its utility. Each user then selects its desired bitrate according to the price set by the VoD service provider.

A. Experimental Settings

In the experiment, we used 1,000 videos which are ranked by their number of views in descending order. As views of videos tend to follow Zipf's distribution with the characteristic exponent $s = 1$ [29], a user watches video with rank i with probability $\frac{1/i}{\sum_{n=1}^{N} 1/n}$, where N is the number of videos. The length of each video was randomly selected from (10, 120] minutes and each user watched 2 videos in this experiment. As Netflix serves videos in bitrates between 100Kbps and 3600Kbps [2], in this experiment, we defined 11 bitrates for each video from 100Kbps to 3600Kbps with 350Kbps increment in each step. We set the default size of a video chunk to 1KB [30]. Each user is randomly assigned a weight of payment cost in [0,1], scale factor α_i is randomly selected in [0,1]. Other parameter settings included: t_g=30 second, N_u^b=10, N_l^b=3 and m=10. We used binary files to represent video files and let nodes receive and delete the binary files to simulate the process of watching a video.

We used the statistics in [31], [32] for the distribution of download bandwidth in the simulation. A node's upload bandwidth was set to 1/3 of its download bandwidth [33]. In order to simulate dynamics of users, users join the system following the Poison distribution with an average rate of 5 users per second [21]. Each user leaves the system after it finishes watching a video and joins the system after 10 minutes. As in [34]–[37], the capacities of nodes follow the Pareto distribution with a mean of 5 and a shape parameter of 1. In the PeerSim simulation, we deployed 10,000 nodes as VoD users, and deployed 10 nodes as cloud servers. In the PlanetLab experiments, we used 350 distributed nodes nationwide to simulate video service users. We used one cloud server, which is functioned by the node with IP 128.112.139.43 in Princeton University.

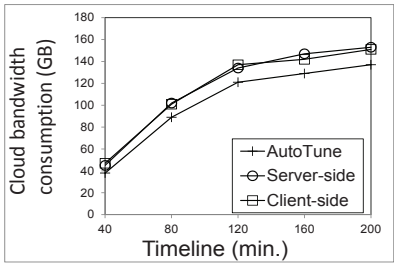
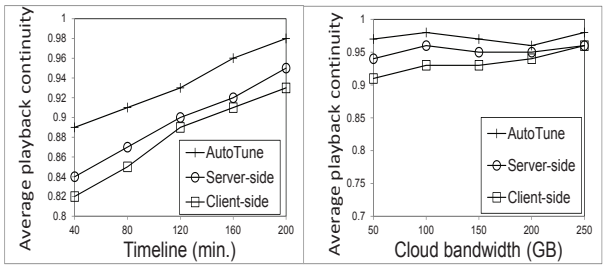


Figure 7: Cloud bandwidth consumption on PlanetLab.

B. Experimental Results on PeerSim

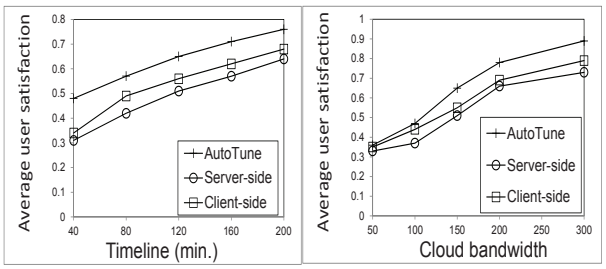
We first present the experimental results of cloud bandwidth consumption of the whole system over time in PeerSim shown in Figure 4. The cloud bandwidth consumption was calculated by the total amount of bytes transferred from the cloud to users. We see that *AutoTune* consumes the least volume of cloud bandwidth due to the reason that the VoD service provider encourages users to download video chunks from peers by setting price on cloud bandwidth consumption, and users minimize their cloud bandwidth consumption in order to increase the utility. *Server-side* and *Client-side* do not have control over cloud bandwidth consumption and they achieve comparable performance.

Video playback continuity is a crucial metric for user quality-of-experience. When there are not enough video chunks stored in a video player's buffer, it suffers from playback interruption until new chunks are downloaded. We divided the length of each video into a sequence of time slots by using a 5-minute window, and recorded whether there is a playback interruption during each time slot. We then measured playback continuity by dividing the number of time slots without playback interruptions by the total number of slots. Figure 5(a) and Figure 5(b) show average playback continuity among all users at different time intervals and at different cloud bandwidth consumption, respectively. We see that *Server-side* generates higher playback continuity



(a) Results at different time intervals.(b) Results at different cloud bandwidth consumption.

Figure 8: Average playback continuity on PlanetLab.



(a) Results at different time intervals.(b) Results at different cloud bandwidth consumption.

Figure 9: Average user satisfaction on PlanetLab.

than *Client-side* because it rejects some bitrate increase requests when cloud bandwidth capacity is not sufficient to support download bandwidth. *AutoTune* generates the highest playback continuity due to the reason that it achieves a tradeoff between minimizing cloud bandwidth consumption and guaranteeing users' satisfaction. Thus, users are able to find sufficient peer contribution when switching to a new bitrate. Note that the average playback continuity cannot reach 1 as the playback is interrupted when a user cannot download video chunks in time due to the reason that cloud bandwidth capacity is not sufficient to support all chunk requests.

We measure the a user's satisfaction when watching a video with bitrate r_i by $ln(1 + r_i)/ln(1 + 3600)$ (shown in Equation (4)), 3600 is the maximum bitrate set in the experiment. Figure 6(a) and Figure 6(b) show average user satisfaction among all users in PeerSim at different time intervals and at different cloud bandwidth consumption, respectively. We see that *AutoTune* generates the highest user satisfaction as it aims to maximize users' satisfaction and at the same time minimize cloud bandwidth consumption. Each user selects a new video bitrate that guarantees its satisfaction and has high peer bandwidth contribution. *Server-side* and *Client-side* produce smaller user satisfaction as they cannot maximize peer bandwidth contribution and failed to maximize video bitrate in return.

C. Experimental Results on PlanetLab

Figure 7 shows the cloud bandwidth consumption of the whole system overtime on PlanetLab. We see the results are consistent with those in Figure 4 due to the same reasons. Note that the relative performances between different methods are similar in simulation and PlanetLab due to intrinsic differences between these methos; the performances of a specific method in simulation and PlanetLab are different in absolute values and trends because the experimental platforms are different.

Figure 8(a) and Figure 8(b) show the average playback continuity among all users at different time intervals and at different cloud bandwidth consumption, respectively. We see that the relative performance in average playback continuity of three comparison methods follows: *Client-side*<*Server-side*< *AutoTune*, due to the same reasons as in Figure 5(a) and Figure 5(b).

Figure 9(a) and Figure 9(b) show average user satisfaction among all users at different time intervals and at different cloud bandwidth consumption, respectively. We see that *AutoTune* generates the highest user satisfaction than *Server-side* and *Client-side* due to the same reasons explained in Figure 6(a) and Figure 6(b).

VI. Conclusions

In this paper, we study a problem of how to achieve a tradeoff between minimizing cloud bandwidth consumption and guaranteeing users' satisfaction (i.e., quality-of-experience) in hybrid VoD systems. To solve this problem, we modeled it as a Stackelberg game and proposed a game-based adaptive bitrate streaming method called AutoTune. In AutoTune, the VoD service provide sets the VoD service price so that each client is encouraged to select the bitrate that it is satisfied with rather than selecting excessively high bitrates. Then, the client can find more peers to supply sufficient bandwidth to upload its requested video instead of relying on the cloud. As a result, the specified VoD service price by the VoD service provide and selected bitrates of the clients minimize cloud bandwidth consumption and guarantee users' satisfaction.

We conducted extensive experiments on the PeerSim simulation and the PlanetLab real-world testbed. The experimental results show that AutoTune outperforms previous adaptive bitrate streaming methods in terms of cloud bandwidth consumption, user satisfaction and playback continuity. In our future work, we will further study how to determine parameters such as weight w_k to meet different user requirements and how to encourage peers to contribute bandwidth through incentives of better cloud service.

ACKNOWLEDGEMENTS

This research was supported in part by U.S. NSF grants NSF-1404981, IIS-1354123, CNS-1254006, CNS-1249603, and Microsoft Research Faculty Fellowship 8300751.

REFERENCES

[1] S. Ibrahim, B. He, and H. Jin. Towards pay-as-you-consume cloud computing. In *Proc. of SCC*, 2011.

[2] V. K. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *Proc. of INFOCOM*, 2012.

[3] R. Torres, A. Finamore, J. Kim, M. Mellia, M. Munafo, and S. Rao. Dissecting video server selection strategies in the youtube cdn. In *Proc. of ICDCS*, 2011.

[4] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Livesky: Enhancing CND with P2P. *TOMCCAP*, 6(3):16, 2010.

[5] I. Trajkovska, J. Salvachua, and A. Mozo. A novel P2P and cloud computing hybrid architecture for multimedia streaming with QoS cost functions. In *Proc. of Multimedia*, 2010.

[6] C. Huang, A. Wang, J. Li, and K. Ross. Understanding hybrid CDN-P2P: why limelight needs its own Red Swoosh. In *Proc. of NOSSDAV*, 2008.

[7] V. Darlagiannis, A. Mauthe, and R. Steinmetz. Sampling cluster endurance for peer-to-peer based content distribution networks. *Multimedia systems*, 13(1):19–33, 2007.

[8] P. Rodriguez, S. Tan, and C. Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Computer Communication Review*, 36(1):75–78, 2006.

[9] AWS on-demand bandwidth pricing. http://aws.amazon.com/cloudfront/pricing/, [Accessed in May, 2015].

[10] D. Niu, C. Feng, and B. Li. Pricing cloud bandwidth reservations under demand uncertainty. In *ACM SIGMETRICS Performance Evaluation Review*, volume 40, pages 151–162, 2012.

[11] A. Mansy and M. Ammar. Analysis of adaptive streaming for hybrid CDN/P2P live video systems. In *Proc. of ICNP*, 2011.

[12] S. Akhshabi, A. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proc. of MMSys*, 2011.

[13] R. Mok, X. Luo, E. Chan, and R. Chang. Qdash: a qoe-aware dash system. In *Proc. of MMSys*, 2012.

[14] R. Pantos and W. May. Http live streaming. *IETF Draft, June*, 2010.

[15] D. Hassoun. Dynamic streaming in flash media server 3.5. http://www.adobe.com/devnet/flashmediaserver/, [Accessed in May, 2015].

[16] K. Hwang, V. Gopalakrishnan, R. Jana, S. Lee, V. Misra, K. Ramakrishnan, and D. Rubenstein. Joint-family: Enabling adaptive bitrate streaming in peer-to-peer video-on-demand. In *Proc. of ICNP*, 2013.

[17] K. Binmore. *Mathematical Analysis: a straightforward approach*. Cambridge University Press, 1982.

[18] G. Nan, Z. Mao, M. Yu, M. Li, H. Wang, and Y. Zhang. Stackelberg game for bandwidth allocation in cloud-based wireless live-streaming social networks. *IEEE Systems Journal*, 8(1):256–267, 2014.

[19] W. Wu, J. Lui, and R. Ma. Incentivizing upload capacity in P2P-VoD systems: a game theoretic analysis. In *Game Theory for Networks*, pages 337–352. 2012.

[20] S. Mostafavi and M. Dehghan. Game theoretic bandwidth procurement mechanisms in live P2P streaming systems. *Multimedia Tools and Applications*, pages 1–24, 2015.

[21] D. Wu, Y. Liu, and K. W. Ross. Modeling and Analysis of Multichannel P2P Live Video Systems. *TON*, 18(4):1248–1260, 2010.

[22] W. Song, D. Tjondronegoro, and M. Docherty. Saving bitrate vs. pleasing users: where is the break-even point in mobile video quality? In *Proc. of Multimedia*, 2011.

[23] B. Fan, D. Andersen, M. Kaminsky, and K. Papagiannaki. Balancing throughput, robustness, and in-order delivery in P2P VoD. In *Proc. of CoNEXT*, 2010.

[24] W. Tushar, W. Saad, H. Poor, and D. Smith. Economics of electric vehicle charging: A game theoretic approach. *TSG*, 3(4):1767–1778, 2012.

[25] D. Niu, H. Xu, B. Li, and S. Zhao. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In *Proc. of INFOCOM*, 2012.

[26] J. Lucas and M. Saccucci. Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics*, 32(1):1–29, 1990.

[27] The PeerSim simulator. http://peersim.sf.net, [Accessed in May, 2015].

[28] PlanetLab. http://www.planet-lab.org/, [Accessed in May, 2015].

[29] H. Shen, Y. Lin, and H. Chandler. An Interest-Based Per-Community P2P Hierarchical Structure for Short Video Sharing in the YouTube Social Network. In *Proc. of ICDCS*, 2014.

[30] K. Xu, M. Zhang, J. Liu, Z. Qin, and M. Ye. Proxy caching for peer-to-peer live streaming. *Computer Networks*, 54(7):1229–1241, 2010.

[31] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *Proc. of SIGCOMM*, 2007.

[32] Xu Cheng and Jiangchuan Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *Proc. of INFOCOM*, 2009.

[33] The difference between upload and download speed for broadband DSL. http://www.broadbandinfo.com/cable/speed-test, [Accessed in May, 2015].

[34] H. Shen and C. Xu. Locality-aware and churn-resilient load balancing algorithms in structured peer-to-peer networks. *TPDS*, 18(6):849–862, 2007.

[35] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: investigating unfairness. In *Proc. of SIGMETRICS/Performance*, 2001.

[36] X. Zhang, Y. Qu, and L. Xiao. Improving distributed workload performance by sharing both cpu and memory resources. In *Proc. of ICDCS*, 2000.

[37] R. Subrata and A. Y. Zomaya. Game-theoretic approach for load balancing in computational grids. *TPDS*, 19(1):66–76, 2008.