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

Yuhua Lin, *Member, IEEE* and Haiying Shen, *Senior Member, IEEE*

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 the 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. To enhance the performance of AutoTune, we further propose the reputation-based incentive scheme and the popularity-based cache management scheme. 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. Also, the proposed enhancement schemes are effective in improving the performance of AutoTune.

Index Terms—Video streaming; Cloud computing; Bitrate adaptation; Noncooperative Stackelberg game

1 Introduction

The cloud has proved to be an effective infrastructure to host stable and robust video streaming services [1]–[5]. 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 [6] for multimedia streaming services, which take advantage of both systems. LiveSky [6] 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 [7]–[10]. 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

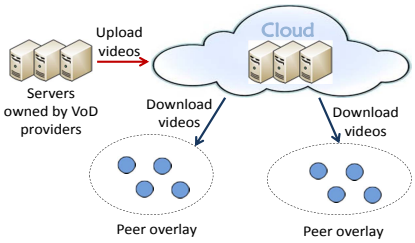


Fig. 1: Overview of a P2P-assisted cloud-based VoD architecture.

are distributed globally. End users can download videos from both the cloud and peers in the P2P network. The cloud acts as the 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 [11]–[13]. To maximize the VoD service provider's profit and attract more 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], [14]–[18]. When the server has low bandwidth and its sending buffer has a small number of

• *Yuhua Lin and Haiying Shen are with the Department of Electrical and Computer Engineering, Clemson University, Clemson, South Carolina 29634.*
E-mail: {yuhual, shenh}@clemson.edu

Manuscript received April 19, 2005; revised September 17, 2014.

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 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 [19]–[23]. 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. We then solve the Stackelberg equilibrium in which cloud bandwidth consumption is minimized while users are satisfied with the selected video bitrates. To enhance the performance of AutoTune, we further propose the reputation-based incentive scheme to motivate users to contribute upload bandwidth and serve video requests, and the popularity-based cache management scheme to increase the availability of video chunks in the P2P network. We conducted extensive experiments in the PeerSim simulator and the PlanetLab real-world testbed. Experimental results show the effectiveness of AutoTune in achieving a high user satisfaction and saving cloud bandwidth consumption, and the effectiveness of the proposed enhancement schemes in improving the performance of AutoTune.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work. Section 3 introduces the goal of this work and provides an overview of AutoTune. Section 4 introduces the detailed design of AutoTune and our proposed reputation-based incentive scheme and popularity-based cache management scheme. Section 5 presents the performance evaluation on both PlanetLab and PeerSim. Section 6 concludes the paper with remarks on our future work.

2 Related Work

LiveSky [6] 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 [8]–[10]. 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, the controller monitors available bandwidth on the server side and assign video bitrates to all users using the available bandwidth capacity. Mansy *et al.* [14] 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. Cicco *et al.* [17] proposed a quality adaptation controller for live streaming systems, which employs feedback control theory to adjust video bitrate based on the server's available bandwidth. However, these methods fail to consider 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 [19], [20]. In these designs, the video client monitors its bandwidth and CPU conditions and adaptively switches video quality during playback. Dynamic adaptive streaming over HTTP (DASH) [2], [15], [16] 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. Joint-Family [21] 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. Huang *et al.* [22] designed two separate phases of operation. That is, when a user's buffer is growing from empty, it selects a video rate based on bandwidth capacity estimation; when its buffer is occupied to a certain range, it picks a video rate based on its buffer condition. Tian *et al.* [23] proposed a method that uses the number of video chunks buffered in the client-side as a feedback signal, and smoothly increases video bitrate when the client's available network bandwidth increases. 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 [24] have been applied to video streaming system to solve the resource allocation problem [25]–[27]. 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.* [25] 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 meet their QoS requirements. Wu *et al.* [26] 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.* [27] 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.

To sum up, existing ABR methods either fail to maximize the users' satisfaction of the video quality or cannot minimize cloud bandwidth consumption. Compared to the existing ABR methods, our proposed AutoTune is advantageous 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.

Previous works study how to provide incentives to encourage users to offer their upload bandwidth to serve other peers in VoD systems [28]–[30]. Wu *et al.* [28] designed a mechanism using monetary rewards to motivate peers to store videos and serve video chunk requests from other peers. Li *et al.* [31] proposed a taxation-based mechanism to incentivize peers with high-bandwidth connections to contribute more upload bandwidth. Some works [29], [30] discourage free-riding behaviors by giving high priority to serve peers who have uploaded a large size of contents to the system. In this paper, we leverage existing reputation system techniques and design a practical reputation-based incentive scheme.

Cache management schemes are designed to improve the availability of files in the P2P system. Least recently used (LRU) [32] and First-In-First-Out (FIFO) [33] schemes replace outdated files in the cache without considering file popularities. Study in [34] shows that swapping out videos based on popularity can achieve a high hit rate. In this paper, we design a popularity-based cache management strategy that considers both video popularity and the amount of bandwidth demanded from peers.

3 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 [21], [35]. 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 [10]. 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 [11], [12], 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 [36]. 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 a price for users' cloud bandwidth usage. Therefore, users can be motivated to choose a 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 this hybrid VoD system, 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 4.1). 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 4.2). We will elaborate each step in Section 4.

4 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 4.1). 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

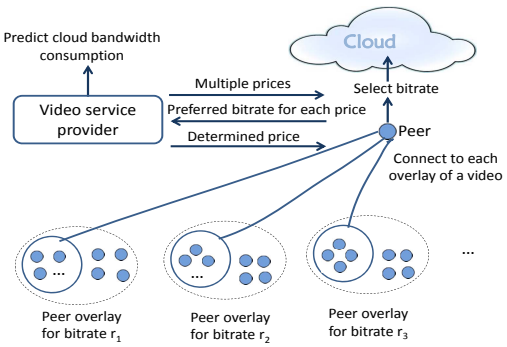


Fig. 2: Overview of the game-based bitrate adaptation process.

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 4.2). Important notations used in this paper are listed in Table 1, in which we use "predefined" to indicate that the parameter is predefined by the VoD service provider.

TABLE 1: 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
Z_u^b	upper bound of # of chunks stored in buffer (predefined)
Z_l^b	lower bound of # of chunks stored in buffer (predefined)
Z^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)
n_i	node i
q_{ik}	reputation score of n_k given by node n_i
S_{ik}^d	size of video chunks n_i downloads from n_k
v_i	a video
θ_i^r	probability that v_i is requested by other users
S_{ji}^u	total chunk size of video v_i uploaded by n_j
S_j^m	the maximum value of S_{ji}^u of all videos stored in n_j' cache
θ_{ji}^s	probability of selecting v_i to swap out from n_j' cache
q_i^c	compensation reputation for storing video v_i

4.1 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. 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 [37]. This strategy allows for a fast start-up, 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.

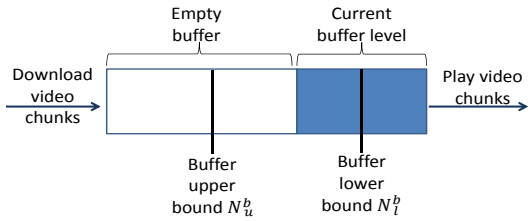


Fig. 3: Demonstration of user buffer utilization.

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 an 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 Z^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 following two conditions hold: 1) its buffer has more than Z_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.

$$PR_k = \{r_j : r_j \geq r_i \wedge \frac{r_j - r_i}{r_i} < \frac{Z^b - Z_u^b}{Z_u^b}\}, if \begin{cases} Z^b > Z_u^b \\ t_g > t_l \end{cases} \quad (1)$$

An increased bitrate means a larger file and longer download time. $\frac{r_j - r_i}{r_i} < \frac{Z^b - Z_u^b}{Z_u^b}$ 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 Z_l^b seconds of chunks, and the last downward bitrate change is more than t_l seconds ago.

$$PR_k = \{r_j : r_j \leq r_1 \wedge \frac{r_i - r_j}{r_i} < \frac{Z_l^b - Z^b}{Z_l^b}\}, if \begin{cases} Z^b < Z_l^b \\ t_g > t_l \end{cases} \quad (2)$$

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 4.2. The newly selected bitrate should achieve a tradeoff between reducing cloud bandwidth consumption and guaranteeing user's quality-of-experience.

Algorithm 1 The client buffer based bitrate adaptation method.

1: **Input:** user k's buffer level Z^b; current bitrate r_i; time gap since last bitrate change t_g;
2: **Output:** new set of video bitrate PR_k;
3: **if** $t_g > t_l$ **then** //time gap since last bitrate change is larger than t_l
4: **if** $Z^b > Z_u^b$ **then** //buffer level exceeds upper bound
5: $PR_k = \{r_j : r_j \geq r_i \wedge \frac{r_j - r_i}{r_i} < \frac{Z^b - Z_u^b}{Z_u^b}\}$
6: **end if**
7: **if** $Z^b < Z_l^b$ **then** //buffer level below lower bound
8: $PR_k = \{r_j : r_j \leq r_1 \wedge \frac{r_i - r_j}{r_i} < \frac{Z_l^b - Z^b}{Z_l^b}\}$
9: **end if**
10: **end if**
11: **return** PR_k

4.2 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 to 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 [38] between the VoD service provider and users. In our established Stackelberg game, on one hand, the VoD service provider needs to set a 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.

4.2.1 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 a 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 [36]. 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 [24] that are commonly used to evaluate user satisfaction [25], [38], 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:

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

4.2.2 Utility Function of the VoD Service Provider

A user aims to download video chunks from its peers as bandwidth resources from the P2P network is free. It downloads videos chunks from the cloud only when upload bandwidth from the user's connected peers is not sufficient to support smooth playback of a video. 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.

4.2.3 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 [39]. 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 [40]. 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 determines 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:

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

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 4.1, 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:

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

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,

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

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}. The process of selecting an optimal bitrate for a user is detailed in Algorithm 2.

Algorithm 2 Pseudocode for optimal bitrate selection.

```
Input: each user's new set of bitrates PR_k;
Output: new video bitrate r_i for each user;
VoD service provider sends VoD service unit prices V =< p_1, p_2, ..., p_M > to users
for each user do
    select R_k =< r_i1, r_i2, ..., r_im > according to Equation (10)
    send R_k to the VoD service provider
end for
VoD service provider determines the optimal unit price p_l according to Equation (11)
for each user do
    selects the optimal bitrate in response to p_l
end for
```

4.3 Reputation-based Incentive Scheme

In a typical P2P system, each peer is allowed to connect to a limited number of neighbors, e.g., the number of neighbors is limited to 80 in BitTorrent. Each user caches video chunks of multiple video bitrates it has downloaded in its local machine. Peers watching the same video or storing the same video in their local machines are organized into an overlay to share video chunks with each other. Similarly, in this

hybrid VoD system, a node maintains active connections to a maximum of N_p peers, and shares and receives video chunks to and from its connected peers (i.e., neighbors). Each user contacts the cloud when it first joins the system, the cloud then assigns it a number of peers in the overlays of the videos that it has stored, so each user can share video chunks with its neighbors in multiple overlays.

In this hybrid VoD system, it is important to incentivize users to be honest and selfless. That is, the users report the optimal bitrates truthfully and contribute bandwidth to the P2P network. Users are motivated to report optimal bitrates honestly due to two facts. When a user reports his/her optimal bitrate, if he/she reports a higher bitrate than his/her actual optimal bitrate, he/she needs to pay a higher price for the reported bitrate; if he/she reports a lower bitrate than his/her actual optimal bitrate, he/she will receive a poorer video quality than he/she desires. However, free-riding is a common phenomenon in the P2P systems that some selfish nodes download much more video chunks than the video chunks they uploaded. Selfish peers may deliberately limit their upload bandwidths or reject download requests from other peers, which leads to increased cloud bandwidth consumption. Therefore, a scheme is needed to encourage peer contribution to reduce the load of the cloud.

The reputation-based incentive scheme requires no monetary costs from the VoD service provider. There are a variety of ways to incentivize peers to contribute upload bandwidth. For example, the VoD service providers can give discounts on the payment cost of watching a video for peers based on the amount of video chunks uploaded by them to other peers, i.e., reducing the value of U_p in Equation (5). To this end, we propose a reputation-based incentive scheme. In this scheme, nodes evaluate the bandwidth contribution of their neighbors by assigning each neighbor a reputation score, which represents the capacity and willingness of a neighbor in providing requested video chunks. When a node receives a number of concurrent chunk requests, it serves the requests based on the highest-reputation-first manner. In this way, in order to get served by peers in the P2P network and reduce the payment cost of cloud bandwidth consumption, each user is motivated to contribute upload bandwidth and earn reputation scores.

4.3.1 Calculation of Reputation Scores

A node n_i maintains a reputation score towards every other node in the system, which is stored in a vector $\mathcal{Q}(i)$=$\{q_{i1}, q_{i2}, ..., q_{in-1}\}$. q_{ik} means the reputation score of n_k given by node n_i from its point of view. q_{ik} is calculated by the cloud, which jointly considers two factors: the size of video chunks n_i downloads from n_k and the total size of video chunks n_k contributes to the P2P network.

$$q_{ik} = \gamma S_{ik}^d + (1-\gamma) \sum_{j=1}^{n} S_{jk}. \quad (12)$$

In Equation (12), S_{ik}^d denotes the size of video chunks n_i downloads from n_k within a certain time period (e.g., a month); $\sum_{j=1}^{n} S_{jk}$ represents the accumulated size of video chunks n_k sends to other peers. $\gamma \in [0, 1]$ is a factor to control how much weight n_i places on its own judgement, i.e., $\gamma = 1$ means n_i evaluates n_k's reputation only based on its own experience and $\gamma = 0$ means n_i evaluates n_k's reputation based n_k's overall contribution. Each node records the cumulated size of video chunks it downloads from every other node and reports this information to the cloud. The cloud collects download information from all nodes and calculates reputation scores for each node, and then sends the reputation scores of all other nodes to each node in the system. As each node maintains a reputation score towards another node that is calculated according to its interaction experience, this method is effective in preventing collusion in which a collective of nodes falsely report high bandwidth contribution for each other.

4.3.2 Reputation Oriented Chunk Request Management

In this hybrid VoD system, 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. If it cannot receive a chunk needed from its neighbors, it will download the missing chunk from the cloud. Each peer has finite upload bandwidth and can serve a limited number of peers concurrently. We use service capacity $\mathcal{C}_s$ to denote the maximum number of users a peer can support simultaneously. Each peer sets the maximum number of peers it supports when it joins the VoD system.

In order to incentivize peers to contribute upload bandwidth and earn reputation scores, we use a reputation oriented chunk request management scheme. When a number of concurrent chunk requests are received by a peer, it serves the requests based on the highest-reputation-first manner. That is, the video chunks requester with high reputation score will get response earlier. If the number of video requests exceeds a peer's service capacity $\mathcal{C}_s$, the peer will serve requests with top $\mathcal{C}_s$ reputation scores and discard other chunk requests.

4.4 Popularity-Based Cache Management

4.4.1 Replacing Videos with Low Popularity and Contribution

In this hybrid VoD system, after a user finishes watching a video, the video is stored in the user's cache to serve video requests from other users. Since videos have different popularities, the probability that a video is requested by a peer can be predicted based on its popularity. The cloud records the number of views for each video during a certain time period (e.g., one week), it then calculates the probability that each video in this hybrid VoD system will be requested based on its popularity. Specifically, the cloud first ranks the videos based on the number of views during a time period; the video with rank 1 has a higher viewing frequency than the video with rank 2. It then calculates the probability that video v_i with rank k is requested by the users in the VoD system (denoted by θ_i^r), which equals the ratio of the number of views of v_i over the total number of views for all videos in this hybrid VoD system [41]. That is,

$$\theta_i^r = \frac{1}{k} / \sum_{d=1}^{n} \frac{1}{d} = 1/(k \cdot \sum_{d=1}^{n} \frac{1}{d}), \quad (13)$$

where k is the rank of v_i and n is the number of videos in the VoD system. The videos are ranked from 1 through n.

After the cloud calculates θ_i^r for each video, this information is sent to all users storing the video in their caches.

Assume user n_j has stored a list of videos in its local machine, V =$(v_1, v_2, ...v_m)$. When the user's cache is full, it swaps an existing video out from V with the newly downloaded video. To increase the availability of videos in the P2P network, these are two cases that we need to consider when selecting an outdated video to swap out of the cache. First, for videos with high popularities, they are likely to receive a large number of video requests, so we need to increase the availability of these videos in the P2P network. That is, we need to let a large number of users cache popular videos in their memories. Therefore, videos with high popularities should have high probabilities to stay in the cache. Second, if a user has shared a large number of video chunks of a specific video (i.e., high contribution) with other users, this video is likely to be requested by other users in the future. Therefore, the videos with high contribution should have high probabilities to stay in the cache so as to provide video chunks to others. In some P2P file sharing applications such as BitTorrent, each client records the total size of chunks of each video that the client has uploaded to the P2P network, which is an indicator of how much bandwidth is demanded from peer contribution for each video. We use S_{ji}^u to denote the total size of chunks of video v_i uploaded by user n_j to the P2P network during a specific period of time.

Considering these two factors in increasing a video's availability in the P2P network, we define the probability to select video v_i to swap out from user n_j's cache (denote by θ_{ji}^s) in Equation (14) as:

$$\theta_{ji}^s = 1 - \theta_i^r \times S_{ji}^u / S_j^m, \tag{14}$$

where S_j^m is the maximum value of S_{ji}^u of all videos stored in user n_j's cache. From Equation (14), we can infer that if v_i has a higher probability to be watched and larger size of video chunks of v_i has been uploaded to the P2P network, it has a small probability to be replaced in the cache. For an unpopular video that a user , it has a high probability to be swapped out of the cache. After we have determined the probabilities to be selected for all videos stored in the user cache, we select a video to swap out of n_j's cache based on these probabilities. The selected video will be replaced by the newly downloaded video.

Algorithm 3 describes the process of selecting a video to swap out of a user's cache. The algorithm first sorts all videos stored in a user's cache based on the videos' popularity (Line 3). For each video stored in the user cache, the algorithm then calculates the probability that it is requested by other users in the VoD system (Line 5). The algorithm then calculates the probability to select a video to swap out from the user cache based on its rank among all videos in the cache (Lines 6). The algorithm finally selects an outdated video based on the calculated probability (Lines 9-15). The computation complexity of Algorithm 3 is $O(m)$, where m is the number of videos in a user cache. This popularity-based cache management strategy increases video availability in the P2P network by considering video popularity when replacing outdated videos from a user's cache.

Algorithm 3 Pseudocode of selecting an existing video from a user cache.

```
1: Input: a list of videos V =(v_1, v_2, ...v_m); sum=0; acc=0;
2: Output: a video to be replaced by the newly download one;
3: sort the videos in V in descending order of popularities.
4: for each video v_i in V do
5:    calculate θ_i^r according to Equation (10)
6:    calculate θ_ji^s according to Equation (14)
7:    add θ_ji^s to sum  //calculate the sum of probabilities
8: end for
9: generate a random number ran within range [0,sum]
10: for each video v_i in V do
11:    add θ_ji^s to acc  //calculate accumulated probability
12:    if acc>ran  //accumulated probability is higher than
       the random number
13:       return v_i  //select the current video
14:    end if
15: end for
```

4.4.2 Compensating Users For Storing Videos with Reputation Scores

As users earn reputation scores by contributing upload bandwidth to peers, if a user stores an unpopular video in its caches, it has a low probability to earn reputation by sharing this unpopular video with other peers. Therefore, we need to compensate a user for storing an unpopular video by rewarding it with a certain reputation. Recall that the cloud records the number of views for each video. The cloud calculates the compensation reputation q_i^c for storing video v_i by:

$$q_i^c = \rho \times (1 - \theta_i^r), \tag{15}$$

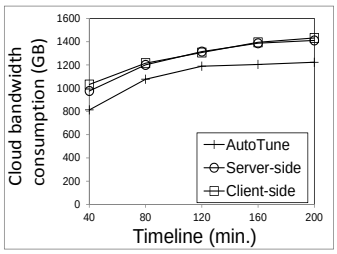
where θ_i^r is the probability that video v_i is requested by other users, and ρ denotes the factor of rewards given to the user for storing a video. We can see from Equation (15) that higher compensation reputation is endowed for storing a video that is unlikely to be requested by users, which guarantees that a user can get certain rewards if it cannot earn rewards by contributing upload bandwidth of an unpopular video to peers. Each time when user n_j enters the VoD system, the cloud calculates its total compensation reputation $q^c(j)$. $q^c(j)$ is calculated by summing up all compensation reputation of videos stored in n_j's cache. Suppose V =$(v_1, v_2, ...v_m)$ is a list of videos stored in n_j's cache, $q^c(j)$ is calculated by:

$$q^c(j) = \sum_{i=1}^{m} q_i^c. \tag{16}$$

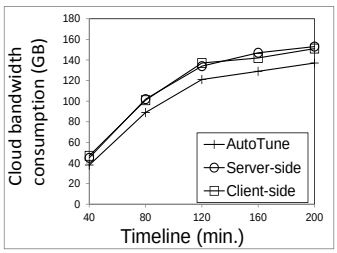
The cloud then adds $q^c(j)$ to q_{kj} calculated in Equation (12) to update the reputation score of n_j. It then sends n_j's updated reputation score to all other users.

5 Performance Evaluation

We conducted experiments in the PeerSim [42] simulator and the PlanetLab [43] 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, users' average bitrate, user satisfaction and playback continuity. We compared *AutoTune* with the server-side adaptation

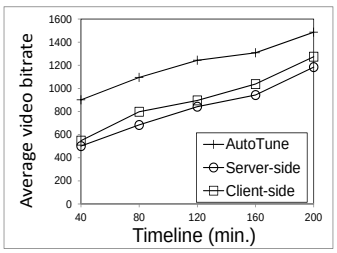


(a) Results in PeerSim.

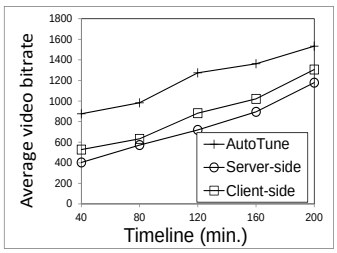


(b) Results in PlanetLab.

Fig. 4: Cloud bandwidth consumption at different time intervals.



(a) Results in PeerSim.



(b) Results in PlanetLab.

Fig. 5: Average video bitrate at different time intervals.

method [14] (denoted by *Server-side*) and client-side adaptation [21] (denoted by *Client-side*) in the hybrid VoD system. These two methods do 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, 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.

TABLE 2: Experiment default parameters.

Parameter	Default value (simulation/PlanetLab)
Number of nodes	10,000/350
Number of videos	1,000
Video length	(10, 120] minutes
Video bitrates	[100, 3600] Kbps
Number of cloud servers	10/1

5.1 Experimental Settings

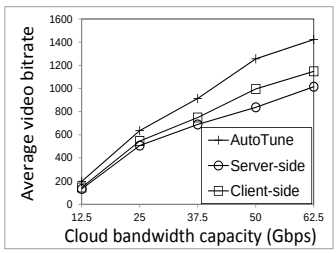
Table 2 shows key settings in our experiments. We used 1,000 videos which are ranked by their number of views in descending order. As views of videos tend to follow Zipf's distribution with the characteristic exponent $s = 1$ [44], a user watches video with rank i with probability $\frac{1/i}{\sum_{n=1}^{N} 1/n}$, where N is the number of videos. The length of each video was randomly selected from (10, 120] minutes. 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 [45]. 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, Z_u^b=10, Z_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 [46], [47] for the distribution of download bandwidth in the simulation. We varied the upload bandwidth capacities of all nodes in the system. In real applications such as BitTorrent, uses can specify the maximum upload bandwidth based on how much they would like to contribute to the P2P network. Specifically, 20% of nodes have an upload capacity of 256kbps, 25% of nodes have 384kbps, 35% of nodes have 512kbps, and 20% have 1024kbps on both PeerSim and PlanetLab [48]. In order to simulate dynamics of users, users join the system following the Poisson distribution with an average rate of 5 users per second [35]. Each user leaves the system after it finishes watching a video and joins the system after 10 minutes. As in [49]–[52], 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. In this experiment, each user watched 2 videos, and the intermediate time between watching 2 consecutive videos is 10 minute. We let each user cached up to 10 videos regardless of the size of videos.

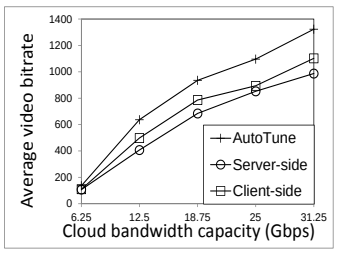
5.2 Overall Performance of *AutoTune*

Figure 4(a) and Figure 4(b) show the cloud bandwidth consumption of the whole system over time in PeerSim and PlanetLab, respectively. The cloud bandwidth consumption was calculated by the total amount of bytes transferred from the cloud to users by the time indicated in the X-axis. 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 a 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. Comparing Figure 4(a) and Figure 4(b), we see that the cloud bandwidth consumption in PeerSim is generally higher than that in PlanetLab because the cloud needs to support a larger number of users.

We then present experimental results on average video bitrate among all users. Figure 5(a) and Figure 5(b) show the average video bitrate at different time intervals in PeerSim and PlanetLab, respectively. Figure 6(a) and Figure 6(b) show the average video bitrate at different cloud bandwidth capacities in PeerSim and PlanetLab, respectively. In Figure 6(a), we manually limited cloud bandwidth capacity in [12.5, 62.5] Gbps; while in Figure 6(b) we manually limited cloud bandwidth capacity in [6.25, 31.25] Gbps. We then tested the performance of each method under different cloud bandwidth capacity restrictions. This experiment aims to test the performance of different methods under limited cloud bandwidth capacity though the VoD service provider

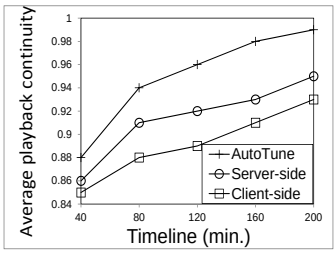


(a) Results in PeerSim.

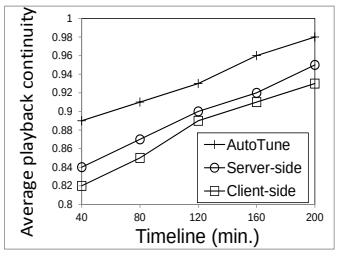


(b) Results in PlanetLab.

Fig. 6: Average video bitrate at different cloud bandwidth capacities.

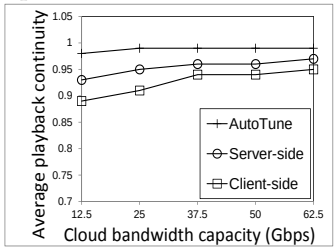


(a) Results in PeerSim.

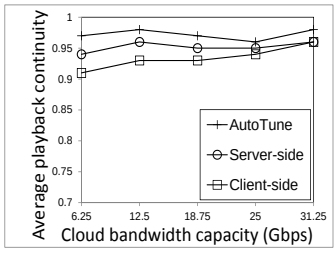


(b) Results in PlanetLab.

Fig. 7: Average playback continuity at different time intervals.

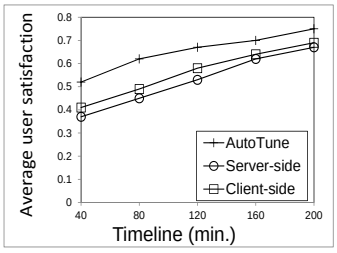


(a) Results in PeerSim.

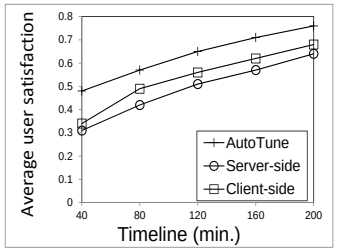


(b) Results in PlanetLab.

Fig. 8: Average playback continuity at different cloud bandwidth capacities.



(a) Results in PeerSim.



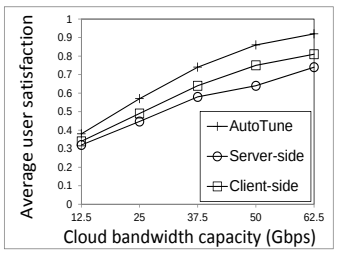
(b) Results in PlanetLab.

Fig. 9: Average user satisfaction at different time intervals.

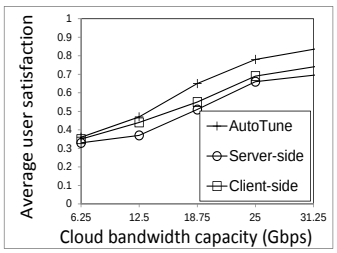
can consume as much cloud bandwidth as its users desire. We see that *Server-side* generates the least average video bitrate among all three comparison methods. This is because when cloud bandwidth capacity is not sufficient to support download bandwidth of all users when receiving a number of bitrate change requests, the cloud tries to maximize the number of users who can switch to their required bitrate. Thus, it cannot increase the average bitrate as some users are not able to increase their video bitrates. *Client-side* produces higher average video bitrate than *Server-side* as it lets users actively switch bitrate based on their buffer condition, so that users are more likely to receive high bitrate videos. *AutoTune* generates the highest average video bitrate as it encourages users to download video chunks from peers by adjusting the VoD service price on cloud bandwidth consumption. Therefore, given the same volume of cloud bandwidth capacity, users in *AutoTune* are more likely to get high video bitrate as they can receive high peer bandwidth contribution. The results also demonstrate the effectiveness of *AutoTune* in reducing cloud bandwidth consumption, as it can achieve the same level of average video bitrate using the least cloud bandwidth capacity. The average video bitrate in PlanetLab is lower than that in PeerSim. This is due to the reason that there are fewer users in PlanetLab, so the video availability in the P2P network is lower and users will select lower bitrates to maximize their utilities.

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 playing time 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 7(a) and Figure 7(b) show average playback continuity among all users at different time intervals in PeerSim and PlanetLab, respectively. Figure 8(a) and Figure 8(b) show average playback continuity among all users at different cloud bandwidth capacities in PeerSim and PlanetLab, respectively. We see that *Server-side* generates higher playback continuity 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. The average video playback continuity in PlanetLab is lower than that in PeerSim. This is due to the reason that the video availability in the P2P network is lower in PlanetLab and users will suffer from playback interruptions when the bandwidth capacity from the cloud and peers is not sufficient to support download bandwidth.

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 9(a) and Figure 9(b) show average user satisfaction among all users at different time intervals in PeerSim and PlanetLab, respectively. Figure 10(a) and Figure 10(b) show average user satisfaction among all users at different cloud bandwidth capacities in PeerSim and PlanetLab, 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

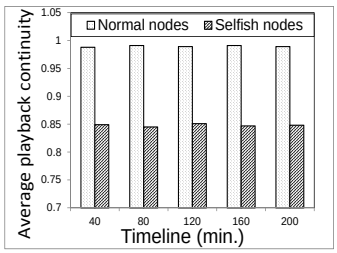


(a) Results in PeerSim.

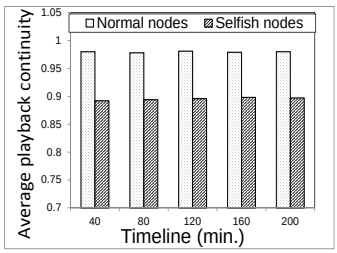


(b) Results in PlanetLab.

Fig. 10: Average user satisfaction at different cloud bandwidth capacities.

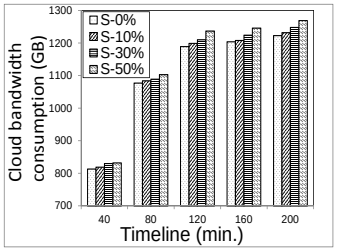


(a) Results in PeerSim.

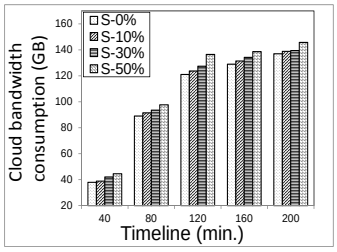


(b) Results in PlanetLab.

Fig. 11: Average playback continuity for different types of nodes.

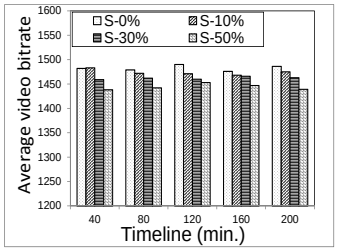


(a) Results in PeerSim.

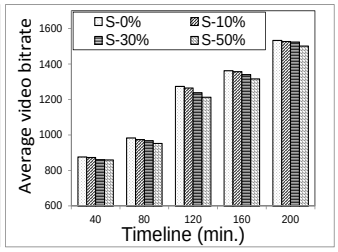


(b) Results in PlanetLab.

Fig. 12: Cloud bandwidth consumption.



(a) Results in PeerSim.



(b) Results in PlanetLab.

Fig. 13: Average video bitrate.

as they cannot maximize peer bandwidth contribution and failed to maximize video bitrate in return.

5.3 Performance of Reputation-Based Incentive Scheme

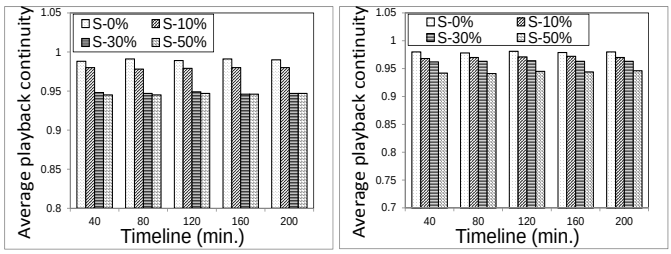
This experiment was divided into 10 sessions, and we used a warmup period of 9 sessions to accumulate reputation scores for all nodes. The cloud records the amount of upload bandwidth contribution of each node after each session, and then calculates the reputation score of each node and sends the reputation scores to all nodes at the beginning of the next session. In this experiment, we have implemented *AutoTune* with the proposed reputation-based incentive scheme. We define a selfish node as a node who throttles its upload bandwidth to 50% of its upload capacity. In the following figures, We use *S-x%* to mean that $x\%$ nodes in the P2P network are selfish, while other nodes are normal ones who do not throttle their upload bandwidth.

In our reputation-based incentive scheme, video chunks requesters with higher reputation scores will have higher priorities in receiving chunks from peers, which increases their video playback continuity. In this experiment, we set the percentage of selfish nodes in the system to 30%. Figure 11(a) and Figure 11(b) show the average playback continuity for two groups of nodes: normal nodes and selfish nodes in PeerSim and PlanetLab, respectively. We see that the normal nodes have higher playback continuity than selfish nodes due to the reason that normal nodes earn high reputation scores by contributing their upload bandwidths. Consequently, selfish nodes have lower priorities in receiving chunks from peers than normal nodes, and may suffer from playback interruption when chunks providers are serving multiple nodes. In order to increase the video playback continuity, nodes are motivated to contribute their upload bandwidth to earn reputation scores.

Figure 12(a) and Figure 12(b) show the cloud bandwidth consumption with different percentage of selfish nodes in PeerSim and PlanetLab, respectively. We see that the cloud bandwidth consumption increases drastically from 40 minute to 120 minute, and the rate of increase slows down after 120 minute. This is due to the reason that when time evolves, more users finish watching videos and more videos are stored in their caches. So users can download video chunks from each other and cloud bandwidth consumption is reduced. We also see that the cloud bandwidth consumption increases when the percentage of selfish nodes increases from 0% to 50 %; *S-0%* consumes the least volume of cloud bandwidth while *S-50%* consumes the largest volume of cloud bandwidth. This is due to the reason that when there are more selfish nodes in the VoD system, the bandwidth contribution from the P2P network gets smaller and users need to download more video chunks from the cloud.

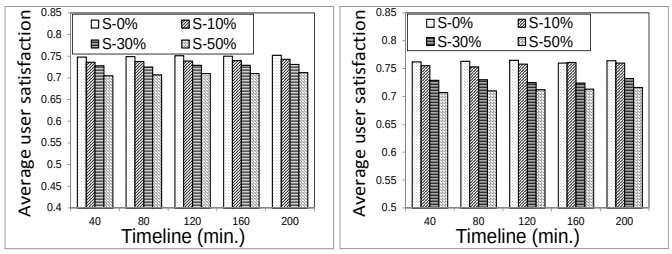
Figure 13(a) and Figure 13(b) plot the average video bitrate with different percentage of selfish nodes in PeerSim and PlanetLab, respectively. We see that when the percentage of selfish nodes increases from 0% to 50%, the average video bitrate decreases, which means that the quality of video is low. When the bandwidth contribution from the P2P network is smaller, users must resort to the cloud for their requested videos. However, uses have higher payment cost on cloud bandwidth consumption. In order to reduce the payment cost and maximize utility based on Equation (5), a user needs to select lower bitrates that lead to smaller cloud bandwidth consumption. The experimental results imply that nodes are incentivized to be cooperative in order to reduce their payment and increase their video quality.

Figure 14(a) and Figure 14(b) show the average playback continuity with different percentage of selfish nodes in PeerSim and PlanetLab, respectively. We see that the average playback continuity decreases when the percentage of selfish nodes increases from 0% to 50 %. This is due to the reason that when bandwidth contribution from the P2P network is reduced, more users resort to the cloud for video chunks. Because users need to download video chunks from



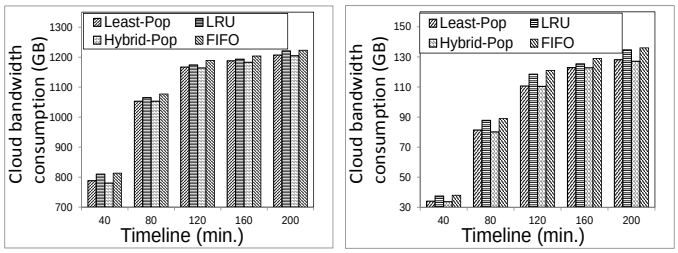
(a) Results in PeerSim. (b) Results in PlanetLab.

Fig. 14: Average playback continuity.



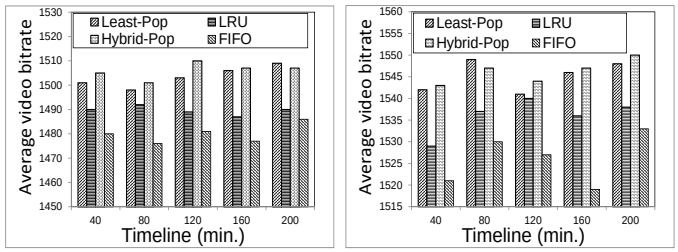
(a) Results in PeerSim. (b) Results in PlanetLab.

Fig. 15: Average user satisfaction.



(a) Results in PeerSim. (b) Results in PlanetLab.

Fig. 16: Cloud bandwidth consumption.



(a) Results in PeerSim. (b) Results in PlanetLab.

Fig. 17: Average video bitrate.

the cloud that is physically far from them, which leads to longer latency. As a result, the playback is more likely to be interrupted when users are watching videos.

Figure 15(a) and Figure 15(b) show the average user satisfaction with different percentage of selfish nodes in PeerSim and PlanetLab, respectively. We see that *S-0%* generates the highest user satisfaction while *S-50%* generates the lowest user satisfaction. This is due to the reason that a higher percentage of selfish nodes in the system lead to lower average video bitrate as shown in Figure 13(a) and Figure 13(b). As we can see from Equation (4), a user's satisfaction in watching a video has a positive correlation with the video bitrate, therefore, the average user satisfaction drops when the average video bitrate is reduced.

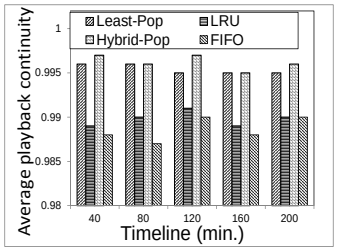
In our proposed reputation based incentive scheme, users with higher reputation scores are served by other peers with higher priorities, so users are encouraged to contribute upload bandwidth to earn reputation scores. Thus, the system with a small percentage of selfish nodes (e.g., S-0% and S-10%) represents the system with our proposed reputation based incentive scheme, while system with a large percentage of selfish nodes (e.g., S-30% and S-50%) represents the system without our proposed reputation based incentive scheme. The experimental results show that the proposed reputation-based incentive scheme is effective in reducing cloud bandwidth consumption and increasing users' average video bitrate, user satisfaction and playback continuity.

5.4 Performance of Popularity-Based Cache Management Strategy

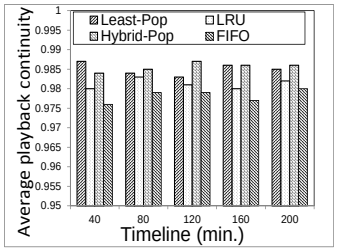
In this experiment, we have implemented *AutoTune* with the popularity-based cache management strategy. As it considers both video popularity and how much bandwidth is demanded from peer contribution when selecting an old video to swap out, we denoted the proposed strategy by *Hybrid-Pop*. We have also implemented two comparison methods. When a peer's cache is full, *FIFO* (First-in-First-out) selects the video that is first downloaded and replaces it with the new video; *LRU* selects the video that is least-recently visited and replaced it with the new video; *Least-Pop* selects the least popular video and replaced it with the new video. When *AutoTune* is not enhanced with any cache management strategy, we assume that it uses the *FIFO* method to replace videos in user caches. Thus, we can observe the effectiveness of the popularity-based cache management strategy by comparing *Hybrid-Pop* and *FIFO*.

Figure 16(a) and Figure 16(b) show the cloud bandwidth consumption at different time intervals in PeerSim and PlanetLab, respectively. We see that the relative performance of different methods follows: *FIFO*>*LRU*>*Least-Pop*>*Hybrid-Pop*. In *FIFO*, the first downloaded videos are replaced by the new videos without considering the video popularities. It cannot increase the availability of popular videos in the P2P network since it does not consider the video popularity when swapping out a video from the user cache. As a result, more users request video chunks from the cloud and it leads to the highest cloud bandwidth consumption. *LRU* replaces the least recently visited video. As we mention earlier, views of videos tend to follow Zipf's distribution. So users tend to watch popular videos, and the least recently visited video in a user's cache is likely to be a less popular video. Thus, *LRU* is able to outperform *FIFO* with smaller cloud bandwidth consumption. *Least-Pop* generates less cloud bandwidth consumption than *LRU* as it replaces the least popular video in the cache, which increases the availability of popular videos in the P2P network. *Hybrid-Pop* generates the least cloud bandwidth consumption. This is due to the reason that besides video popularity, it also considers the bandwidth demand from the P2P network of each video when replacing an old video from user cache. In *Hybrid-Pop*, popular videos or videos whose chunks are frequently downloaded by other users have a small probability to be swapped out of the cache, so users are more likely to download chunks they need from the P2P network.

Figure 17(a) and Figure 17(b) plot the average video bitrate at different time intervals in PeerSim and Planet-

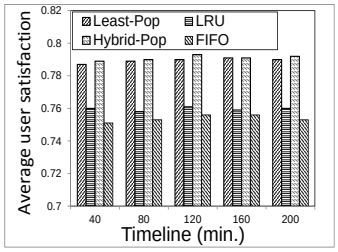


(a) Results in PeerSim.

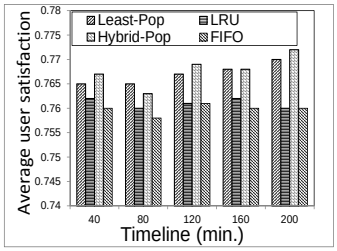


(b) Results in PlanetLab.

Fig. 18: Average playback continuity.



(a) Results in PeerSim.



(b) Results in PlanetLab.

Fig. 19: Average user satisfaction.

Lab, respectively. We see that the results follow *FIFO*< *LRU*<*Least-Pop*<*Hybrid-Pop*. This is because when the cloud bandwidth is increased, in order to maximize his/her utility, users have to select lower video bitrate to reduce payment cost on cloud bandwidth consumption.

Figure 18(a) and Figure 18(b) plot the average playback continuity at different time intervals in PeerSim and PlanetLab, respectively. We see that the relative performance of different methods follows: *FIFO*<*LRU*<*Least-Pop*<*Hybrid-Pop*. This is due to the same reason as explained in Figure 16(a) and Figure 16(b). *FIFO* and *LRU* replace an old video in the cache without considering the video popularities, which may reduce the availability of popular videos in the P2P network and increase the number of chunk requests to the cloud. Some users may suffer from playback interruption as they need to download video chunks from the cloud that is physically far from the user. On the other hand, *Least-Pop* increases the availability of popular videos in the P2P network so that users are more likely to download video chunks they need from nearby peers in time. *Hybrid-Pop* is effective in increasing the availability of videos in the P2P network, which enable users to download video chunks from nearby peers instead of the long-distance cloud. Therefore, users can download video chunks with short latency.

Figure 19(a) and Figure 19(b) plot the average user satisfaction at different time intervals in PeerSim and PlanetLab, respectively. We see that the results follow *FIFO*<*LRU*<*Least-Pop*<*Hybrid-Pop*. As shown in Figure 17(a) and Figure 17(b), the average video bitrate of different methods follows *FIFO*<*LRU*<*Least-Pop*<*Hybrid-Pop*. At the same time, the user satisfaction has a positive correlation with the video bitrate. Thus, the relative performance of different methods in the average user satisfaction mirrors that in Figure 17(a) and Figure 17(b).

Comparing *Hybrid-Pop* and *FIFO*, we can see the performance difference when *AutoTune* is implemented with and without the popularity-based cache management strategy. We see that *Hybrid-Pop* is effective in reducing cloud bandwidth consumption and increasing users' average video bitrate, user satisfaction and playback continuity.

6 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 provider 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. The client can then 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. To enhance the performance of AutoTune, we further proposed the reputation-based incentive scheme and the popularity-based cache management scheme to further increase the bandwidth contribution from peers and reduce the cloud bandwidth consumption.

We conducted extensive experiments in 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, users' average bitrate, user satisfaction and playback continuity. The experimental results also show the effectiveness of the proposed enhancement schemes in improving the performance of AutoTune. 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. An early version of this work was presented in the Proceedings of P2P 2015 [53].

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.

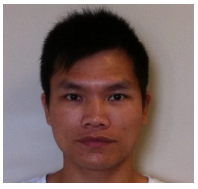
[4] W. Xiao, W. Bao, X. Zhu, C. Wang, L. Chen, and L. Yang. Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment. *TPDS*, 1(99):1–14, 2015.

[5] Z. Wang, B. Li, L. Sun, W. Zhu, and S. Yang. Dispersing instant social video service across multiple clouds. *TPDS*, 27(3):735–747, 2015.

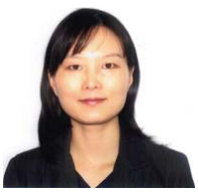
[6] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li. Livesky: Enhancing CDN with P2P. *TOMCCAP*, 6(3):16, 2010.

[7] 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.

[8] 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.
[9] 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.
[10] 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.
[11] AWS on-demand bandwidth pricing. http://aws.amazon.com /cloudfront/pricing/, [Accessed in May, 2015].
[12] 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.
[13] Y. Lin, H. Shen, and L. Chen. Ecoflow: An economical and deadline-driven inter-datacenter video flow scheduling system. In *Proc. of ACM Multimedia*, 2015.
[14] A. Mansy and M. Ammar. Analysis of adaptive streaming for hybrid CDN/P2P live video systems. In *Proc. of ICNP*, 2011.
[15] S. Akhshabi, A. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proc. of MMSys*, 2011.
[16] R. Mok, X. Luo, E. Chan, and R. Chang. Qdash: a qoe-aware dash system. In *Proc. of MMSys*, 2012.
[17] L. Cicco, S. Mascolo, and V. Palmisano. Feedback control for adaptive live video streaming. In *Proc. of MMSys*, 2011.
[18] X. Li and B. Veeravalli. A differentiated quality adaptation approach for scalable streaming services. *TPDS*, 26(8):2089–2099, 2015.
[19] R. Pantos and W. May. Http live streaming. *IETF Draft, June*, 2010.
[20] D. Hassoun. Dynamic streaming in flash media server 3.5. http://www.adobe.com/devnet/flashmediaserver/, [Accessed in May, 2015].
[21] 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.
[22] T. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. of SIGCOMM*, 2014.
[23] G. Tian and Y. Liu. Towards agile and smooth video adaptation in dynamic http streaming. In *Proc. of CoNEXT*, 2012.
[24] K. Binmore. *Mathematical Analysis: a straightforward approach*. Cambridge University Press, 1982.
[25] 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.
[26] 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.
[27] S. Mostafavi and M. Dehghan. Game theoretic bandwidth procurement mechanisms in live P2P streaming systems. *Multimedia Tools and Applications*, pages 1–24, 2015.
[28] W. Wu, R. Ma, and J. Lui. Distributed caching via rewarding: An incentive scheme design in p2p-vod systems. *TPDS*, 25(3):612–621, 2014.
[29] J. Mol, J. Pouwelse, M. Meulpolder, D. Epema, and H. Sips. Give-to-get: free-riding resilient video-on-demand in p2p systems. In *Proc. of Electronic imaging*, 2008.
[30] F. Pianese, D. Perino, J. Keller, and E. Biersack. Pulse: an adaptive, incentive-based, unstructured p2p live streaming system. *IEEE Transactions on Multimedia*, 9(8):1645–1660, 2007.
[31] A. Li, Y. Liang, and D. Wu. Utilizing layered taxation to provide incentives in p2p streaming systems. *Journal of Systems and Software*, 85(8):1749–1756, 2012.
[32] S. Podlipnig and L. Böszörmenyi. A survey of web cache replacement strategies. *CSUR*, 35(4):374–398, 2003.
[33] D. Krishnappa, S. Khemmarat, L. Gao, and M. Zink. On the feasibility of prefetching and caching for online tv services: a measurement study on hulu. In *Passive and Active Measurement*, pages 72–80, 2011.
[34] S. Das, Z. Naor, and M. Raj. Popularity-based caching for iptv services over p2p networks. *Peer-to-Peer Networking and Applications*, pages 1–14, 2015.
[35] D. Wu, Y. Liu, and K. W. Ross. Modeling and Analysis of Multichannel P2P Live Video Systems. *TON*, 18(4):1248–1260, 2010.
[36] 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.
[37] B. Fan, D. Andersen, M. Kaminsky, and K. Papagiannaki. Balancing throughput, robustness, and in-order delivery in P2P VoD. In *Proc. of CoNEXT*, 2010.
[38] W. Tushar, W. Saad, H. Poor, and D. Smith. Economics of electric vehicle charging: A game theoretic approach. *TSG*, 3(4):1767–1778, 2012.
[39] 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.
[40] J. Lucas and M. Saccucci. Exponentially weighted moving average control schemes: Properties and enhancements. *Technometrics*, 32(1):1–29, 1990.
[41] X. Cheng and J. Liu. NetTube: Exploring social networks for peer-to-peer short video sharing. In *Proc. of INFOCOM*, 2009.
[42] The PeerSim simulator. http://peersim.sf.net, [Accessed in May, 2015].
[43] PlanetLab. http://www.planet-lab.org/, [Accessed in May, 2015].
[44] 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.
[45] 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.
[46] C. Huang, J. Li, and K. W. Ross. Can internet video-on-demand be profitable? In *Proc. of SIGCOMM*, 2007.
[47] Xu Cheng and Jiangchuan Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *Proc. of INFOCOM*, 2009.
[48] D. Wu, Y. Liang, J. He, and X. Hei. Balancing performance and fairness in p2p live video systems. *TCSVT*, 23(6):1029–1039, 2013.
[49] 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.
[50] N. Bansal and M. Harchol-Balter. Analysis of srpt scheduling: investigating unfairness. In *Proc. of SIGMETRICS/Performance*, 2001.
[51] X. Zhang, Y. Qu, and L. Xiao. Improving distributed workload performance by sharing both cpu and memory resources. In *Proc. of ICDCS*, 2000.
[52] R. Subrata and A. Y. Zomaya. Game-theoretic approach for load balancing in computational grids. *TPDS*, 19(1):66–76, 2008.
[53] Y. Lin and H. Shen. Autotune: Game-based adaptive bitrate streaming in p2p-assisted cloud-based vod systems. In *Proc. of P2P*, 2015.



Yuhua Lin Yuhua Lin received both his BS degree in Software Engineering and MS degree in Computer science from Sun Yat-sen University, China in 2009 and 2012 respectively. He is currently a Ph.D student in the Department of Electrical and Computer Engineering of Clemson University. His research interests focus on effective and economical content delivery strategy on the cloud.



Haiying Shen Haiying Shen received the BS degree in Computer Science and Engineering from Tongji University, China in 2000, and the MS and Ph.D. degrees in Computer Engineering from Wayne State University in 2004 and 2006, respectively. She is currently an Associate Professor in the Department of Electrical and Computer Engineering at Clemson University. Her research interests include distributed computer systems and computer networks, with an emphasis on P2P and content delivery networks, mobile computing, wireless sensor networks, and grid and cloud computing. She was the Program Co-Chair for a number of international conferences and member of the Program Committees of many leading conferences. She is a Microsoft Faculty Fellow of 2010 and a member of the IEEE and ACM.